HP Prime Graphing Calculator

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

HP Company shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual or the examples contained herein.

Product Regulatory and Environment Information is provided on the CD shipped with this product.

Second Edition: September 2016

First Edition: July 2015

Document Part Number: 813269-002

Table of contents

1 Preface

Manual conventions

The following conventions are used in this manual to represent the keys that you press and the menu options that you choose to perform operations.

- A key that initiates an unshifted function is represented by an image of that key:

  SIN, EEX, Settings, and so on.

- A key combination that initiates a shifted function (or inserts a character) is represented by the appropriate shift key (Shift or ALPHA) followed by the key for that function or character:

  Shift LN initiates the natural exponential function and ALPHA 3 inserts the pound character (#).

  The name of the shifted function may also be given in parentheses after the key combination:

  Shift Esc (Clear), Shift Symb (Setup)

- A key pressed to insert a digit is represented by that digit:

  5, 7, 8, and so on.

- All fixed on-screen text—such as screen and field names—appear in bold:

  CAS Settings, **xstep**, **Decimal Mark**, and so on.

- A menu item selected by touching the screen is represented by an image of that item:

  Sto ▶, OK, Cancel, and so on.

NOTE: You must use your finger to select a menu item. Using a stylus or something similar will not select whatever is touched.

- Characters on the entry line are set in a nonproportional font, as follows:

  `Function`, `Polar`, `Parametric`, `Ans`, and so on.

- Cursor keys are represented by ▲, ▼, ▶, and ◀. You use these keys to move from field to field on a screen or from one option to another in a list of options.

- Error messages are enclosed in quotation marks:

  “Syntax Error”

2 Getting started

The HP Prime Graphing Calculator is an easy-to-use yet powerful graphing calculator designed for secondary mathematics education and beyond. It offers hundreds of functions and commands, and includes a computer algebra system (CAS) for symbolic calculations.

In addition to an extensive library of functions and commands, the calculator comes with a set of HP apps. An HP app is a special application designed to help you explore a particular branch of mathematics or to solve a problem of a particular type. For example, there is a HP app that will help you explore geometry and another to help you explore parametric equations. There are also apps to help you solve systems of linear equations and to solve time-value-of-money problems.

The HP Prime calculator also has its own programming language you can use to explore and solve mathematical problems.

Functions, commands, apps, and programming are covered in detail later in this guide. In this chapter, the general features of the calculator are explained, along with common interactions and basic mathematical operations.

Before starting

Charge the battery fully before using the calculator for the first time. To charge the battery, do one of the following:

- Connect the calculator to a computer by using the USB cable that came in the package with your HP Prime calculator. (The computer must be on for charging to occur.)
- Connect the calculator to a wall outlet using the HP-provided wall adapter.

When the calculator is on, a battery symbol appears in the title bar of the screen. Its appearance indicates the charge level of the battery. A discharged battery takes approximately 4 hours to fully charge.

⚠ **WARNING!**

Battery warning

- To reduce the risk of fire or burns, do not disassemble, crush, or puncture the battery; do not short the external contacts; and do not dispose of the battery in fire or water.
- To reduce potential safety risks, only use the battery provided with the calculator, a replacement battery provided by HP, or a compatible battery recommended by HP.
- Keep the battery away from children.
- If you encounter problems when charging the calculator, stop charging and contact HP immediately.

Adapter warning

- To reduce the risk of electric shock or damage to equipment, only plug the AC adapter into an AC outlet that is easily accessible at all times.
- To reduce potential safety risks, only use the AC adapter provided with the calculator, a replacement AC adapter provided by HP, or an AC adapter purchased as an accessory from HP.

On/off, cancel operations

To turn on

Press On Off to turn on the calculator.

To cancel

When the calculator is on, pressing the Esc Clear key cancels the current operation. For example, it will clear whatever you have entered on the entry line. It will also close a menu and a screen.

To turn off

Press Shift On Off (Off) to turn the calculator off.

To save power, the calculator turns itself off after several minutes of inactivity. All stored and displayed information is saved.

The Home View

Home view is the starting point for many calculations. Most mathematical functions are available in the Home view. Some additional functions are available in the computer algebra system (CAS). A history of your previous calculations is retained and you can reuse a previous calculation or its result.

To display Home view, press ⌂ Settings .

The CAS View

CAS view enables you to perform symbolic calculations. It is largely identical to Home view—it even has its own history of past calculations—but the CAS view offers some additional functions.

To display CAS view, press CAS Settings .

Protective cover

The calculator is provided with a slide cover to protect the display and keyboard. Remove the cover by grasping both sides of it and pulling down.

You can reverse the slide cover and slide it onto the back of the calculator. This will ensure that you do not misplace the cover while you are using the calculator.

To prolong the life of the calculator, always place the cover over the display and keyboard when you are not using the calculator.

The display

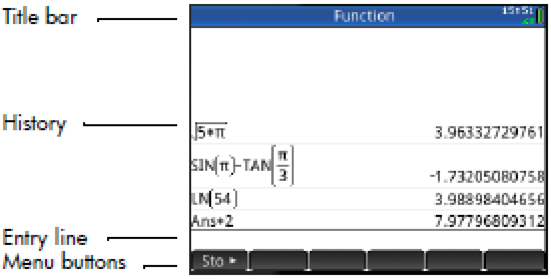
Adjusting the brightness

To adjust the brightness of the display, press and hold On Off, and then press the + Ans ; or − Base ± key to increase or decrease the brightness. The brightness will change with each press of the + Ans ; or On Off key.

Clearing the display

- Press Esc Clear or On Off to clear the entry line
- Press Shift Esc Clear (Clear) to clear the entry line and the history.

Sections of the display



Function

√5•π	3.96332729761
SIN(π)-TAN(π/3)	-1.73205080758
LN(54)	3.98898404656
Ans•2	7.97796809312

Sto ▸

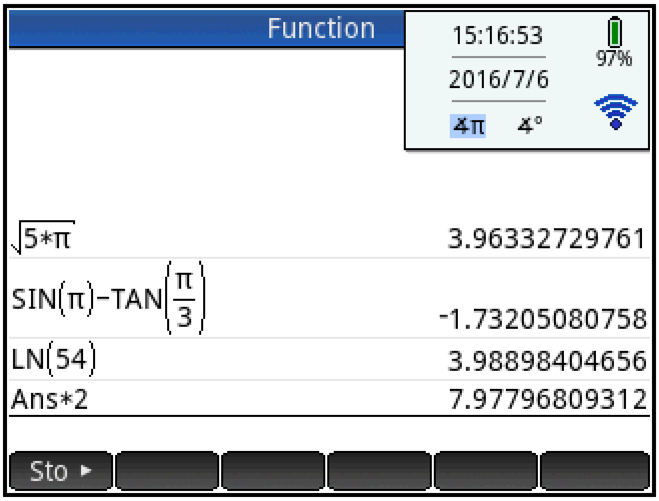
Home view has four sections (shown above). The title bar shows either the screen name or the name of the app you are currently using—**Function** in the example above. It also shows the time, a battery power indicator, and a number of symbols that indicate various calculator settings. These are explained below. The history displays a record of your past calculations. The entry line displays the object you are currently entering or modifying. The menu buttons are options that are relevant to the current display. Select an option by tapping the corresponding menu button. You close a menu, without making a selection from it, by pressing Esc Clear .

Annunciators are symbols or characters that appear in the title bar. They indicate current settings, and also provide time and battery power information.

Annunciator	**Meaning**
∡° [Lime green]	The angle mode setting is currently degrees.
∡π [Lime green]	The angle mode setting is currently radians.
⬆S [Cyan]	The Shift key is active. The function shown in blue on a key will be activated when a key is pressed. Press Shift to cancel shift mode.

Annunciator	Meaning
CAS [White]	You are working in CAS view, not Home view.
A...Z [Orange]	In Home View, this means the Alpha key is active. The character shown in orange on a key will be entered in uppercase when a key is pressed. See Shift keys on page 10 for more information. In CAS view, this means he Alpha–Shift key combination is active. The character shown in orange on a key will be entered in uppercase when a key is pressed. See Shift keys on page 10 for more information.
a...z [Orange]	In Home view, this means the Alpha–Shift key combination is active. The character shown in orange on a key will be entered in lowercase when a key is pressed. See Shift keys on page 10 for more information. In CAS view, this means the Alpha key is active. The character shown in orange on a key will be entered in lowercase when a key is pressed. See Shift keys on page 10 for more information.
⬆U [Yellow]	The user keyboard is active. All the following key presses will enter the customized objects associated with the key. You can customize the user keyboard presses.
1U [Yellow]	The user keyboard is active. The next key press will enter the customized object associated with the key. You can customize the user keyboard presses.
[Time]	Displays the current time. The default is 24-hour format, but you can choose am–pm format. See Home settings on page 20 for more information.
🔋 [Green with gray border]	Indicates the battery charge.

The Quick Settings menu



Function

15:16:53

2016/7/6

∡π ∡°

97%

$\sqrt{5*\pi}$	3.96332729761
$\mathrm{SIN}(\pi)-\mathrm{TAN}\left(\frac{\pi}{3}\right)$	-1.73205080758
LN(54)	3.98898404656
Ans*2	7.97796809312

Tap the right side of the title bar (where the time, battery, and angle measure mode are shown) to open the Quick Settings menu. Actions you can take in this menu include the following:

- Tap one of the angle icons to change the angle measure mode (radians or degrees).
- Tap the date/time to open a monthly calendar. You can navigate between months to find dates of interest.
- Tap the wireless icon to connect to the nearest HP Classroom Network or to disconnect from the current HP Classroom Network.

Navigation

The HP Prime offers two modes of navigation: touch and keys. In many cases, you can tap on an icon, field, menu, or object to select (or deselect) it. For example, you can open the Function app by tapping once on its icon in the Application Library. However, to open the Application Library, you will need to press a key: Apps Info.

Instead of tapping an icon in the Application Library, you can also press the cursor keys—▲, ▼, ◀, ▶—until the app you want to open is highlighted, and then press Enter ≈. In the Application Library, you can also type the first one or two letters of an app's name to highlight the app. Then either tap the app's icon or press Enter ≈ to open it.

Sometimes a touch or key–touch combination is available. For example, you can deselect a toggle option either by tapping twice on it, or by using the arrow keys to move to the field and then tapping a touch button along the bottom of the screen (in this case √).

NOTE: You must use your finger or a capacitive stylus to select an item by touch.

Touch gestures

The HP Prime calculator recognizes the following touch gestures:

- Tap—Point to an item on the screen, and then tap one finger to select the item.
- Tap and hold—Place your finger on the screen and hold it there for a moment.
- Scrolling—Place a finger on the screen and then drag it up, down, left, right, or diagonally to move up, down, sideways, or diagonally on a page or image.
- One-finger slide—To scroll across the screen, lightly slide one finger across the screen in the direction you want to move. To drag, in the Plot view of the Geometry app only, press and hold an object, and then drag the object to move it. To select multiple cells in the Numeric view of the Spreadsheet, Statistics 1Var, and Statistics 2Var apps and in the List and Matrix Editors, tap and hold a cell, and then drag your finger to select subsequent cells. This selection can be copied and pasted like a single value.
- 2-finger pinch zoom—Zoom out by placing two fingers apart on the screen and then moving your fingers together. Zoom in by placing two fingers together on the screen and then moving your fingers apart. In the Spreadsheet app, this gesture controls the width of columns and the height of rows.

Touch gestures may not be supported in all apps, editors, and input forms, and their function might vary. Keep the following guidelines in mind:

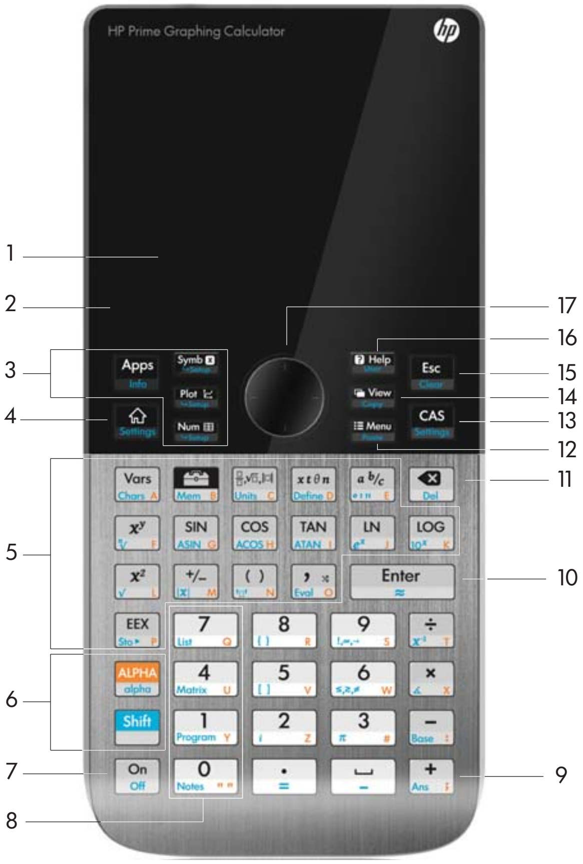
- In Plot view, if a 2-finger pinch zoom gesture is performed horizontally, the zoom is performed on the x-axis only. If a 2-finger pinch zoom is performed vertically, the zoom is performed on the y-axis only. If a 2-finger pinch zoom is performed diagonally, a square zoom is performed (that is, the zoom is performed on both axes). In the Geometry app, only the diagonal zoom is supported.
- In Numeric view, if a 2-finger pinch zoom is performed vertically, the zoom is performed on the currently selected row of the table. A zoom in decreases the common difference in the x-values and a zoom out increases the common difference in the x-values. If a 2-finger pinch zoom is performed horizontally, the column width changes.

The keyboard

The numbers in the legend below refer to the parts of the keyboard described in the illustration on the next page.

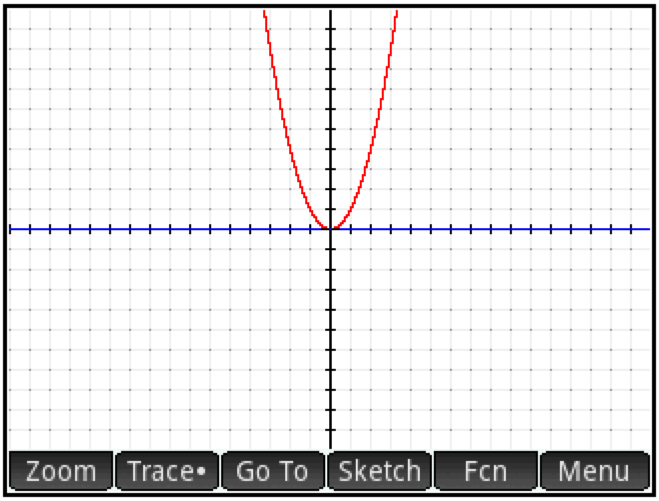
Number	Feature
1	LCD and touchscreen: 320 × 240 pixels
2	Context-sensitive touch-button menu
3	HP Apps keys
4	Home view and preference settings
5	Common math and science functions
6	Alpha and Shift keys
7	On, Cancel, and Off key
8	List, matrix, program, and note catalogs
9	Last Answer key (Ans)
10	Enter key
11	Backspace and Delete key

Number	Feature
12	Menu (and Paste) key
13	CAS (and CAS preferences) key
14	View (and Copy) key
15	Escape (and Clear) key
16	Help key
17	Rocker wheel (for cursor movement)



Context-sensitive menu

A context-sensitive menu occupies the bottom line of the screen.



The options available depend on the context, that is, the view you are in. Note that the menu items are activated by touch.

There are two types of buttons on the context-sensitive menu:

- Menu button—tap to display a pop-up menu. These buttons have square corners along their top (such as Zoom in the illustration above).
- Command button—tap to initiate a command. These buttons have rounded corners (such as Go To in the illustration above).

Entry and edit keys

Keys	Purpose
0 to 9	Enter numbers.
On or Esc	Cancels the current operation or clears the entry line.
Enter	Enters an input or executes an operation. In calculations, Enter acts like "=". When OK or Start is present as a menu key, Enter acts the same as pressing OK or Start.
+/–	For entering a negative number. For example, to enter –25, press +/– 25. **NOTE:** This is not the same operation that is performed by the subtraction key (–).
Template key	Displays a palette of preformatted templates representing common arithmetic expressions.
$x\,t\,\theta\,n$	Enters the independent variable (that is, either X, T, θ, or N, depending on the app that is currently active).

Keys	Purpose
Shift 6	Displays a palette of comparison operators and Boolean operators.
Shift 9	Displays a palette of common math and Greek characters.
Shift a b/c	Automatically inserts the degree, minute, or second symbol according to the context.
Del	Deletes the character to the left of the cursor. It returns the highlighted field to its default value, if it has one.
Shift Del	Deletes the character to the right of the cursor.
Shift Esc (Clear)	Clears all data on the screen (including the history). On a settings screen—for example Plot Setup—returns all settings to their default values.
▲ ▼ ◀ ▶	Move the cursor around the display. Press Shift ▼ to move to the end of a menu or screen, or Shift ▲ to move to the start. These keys represent the directions of the rocker wheel. Diagonal movements are also supported by the rocker wheel.
Shift Vars	Displays all the available characters. To enter a character, use the cursor keys to highlight it, and then tap OK. To select multiple characters, select one, tap Echo, and continue likewise before pressing OK. There are many pages of characters. You can jump to a particular Unicode block by tapping More and selecting the block. You can also flick from page to page.

Shift keys

There are two shift keys that you use to access the operations and characters printed on the bottom of the keys: Shift and ALPHA.

Key	Purpose
Shift	Press Shift to access the operations printed in blue on a key. For instance, to access the settings for Home view, press Shift Settings.
ALPHA	Press the ALPHA key to access the characters printed in orange on a key. For instance, to type Z in Home view, press ALPHA and then press 2. For a lowercase letter, press ALPHA Shift and then the letter. In CAS view, ALPHA

Key	Purpose
	and another key gives a lowercase letter, and ALPHA Shift and another letter gives an uppercase letter.

Adding text

The text you can enter directly is shown by the orange characters on the keys. These characters can only be entered in conjunction with the ALPHA and Shift keys. Both uppercase and lowercase characters can be entered, and the method is exactly the opposite in CAS view than in Home view.

Keys	Effect in Home view	Effect in CAS view
ALPHA	Makes the next character uppercase.	Makes the next character lowercase.
ALPHA ALPHA	Lock mode: makes all characters uppercase until the mode is reset.	Lock mode: makes all characters lowercase until the mode is reset.
Shift	With uppercase locked, makes the next character lowercase.	With lowercase locked, makes the next character uppercase.
ALPHA Shift	Makes the next character lowercase.	Makes the next character uppercase.
ALPHA Shift ALPHA	Lock mode: makes all characters lowercase until the mode is reset.	Lock mode: makes all characters lowercase until the mode is reset.
Shift	With lowercase locked, makes the next character uppercase.	With uppercase locked, makes the next character lowercase.
Shift ALPHA	With lowercase locked, makes all characters uppercase until the mode is reset.	With uppercase locked, makes all characters lowercase until the mode is reset.
ALPHA	Reset uppercase lock mode.	Replace lowercase lock mode.
ALPHA ALPHA ALPHA ALPHA	Replace lowercase lock mode.	Reset uppercase lock mode.

You can also enter text (and other characters) by displaying the characters palette: Shift Vars.

Math keys

The most common math functions have their own keys on the keyboard (or a key in combination with the Shift key).

Example 1: To calculate SIN(10), press SIN 10, and then press Enter. The answer displayed is –0.544… (if your angle measure setting is radians).

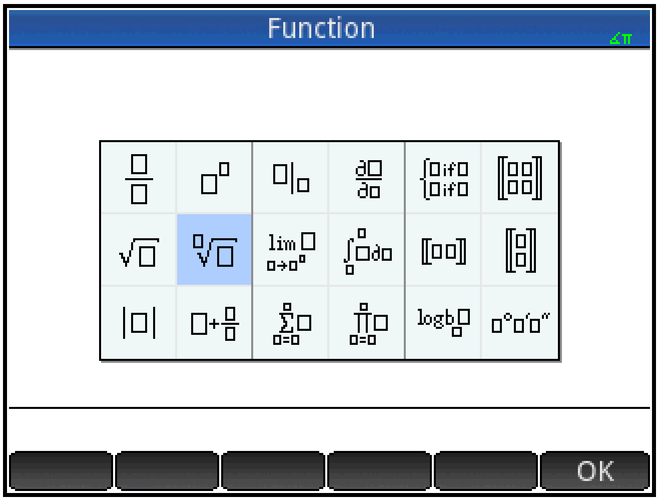
Example 2: To find the square root of 256, press Shift x^2 256 and press Enter. The answer displayed is 16. Notice that the Shift key initiates the operator represented in blue on the next key pressed (in this case √ on the x^2 key).

The mathematical functions not represented on the keyboard are on the **Math**, **CAS**, and **Catlg** menus.

NOTE: The order in which you enter operands and operators is determined by the entry mode. By default, the entry mode is *textbook*, which means that you enter operands and operators just as you would if you were writing the expression on paper. If your preferred entry mode is Reverse Polish Notation, the order of entry is different.

Math template

The math template key (Units) helps you insert the framework for common calculations (and for vectors, matrices, and hexagesimal numbers). It displays a palette of preformatted outlines to which you add the constants, variables, and so on. Just tap on the template you want (or use the arrow keys to highlight it and press Enter). Then enter the components needed to complete the calculation.



Example: Suppose you want to find the cube root of 945:

1. In Home view, press Units .

2. Select $\sqrt[\square]{\square}$.

   The skeleton or framework for your calculation now appears on the entry line: $\sqrt[\square]{\blacksquare}$.

3. Each filled box on the template must be completed. Any hollow boxes are optional.

   3 ▶ 945

4. Press Enter to display the result: 9.813...

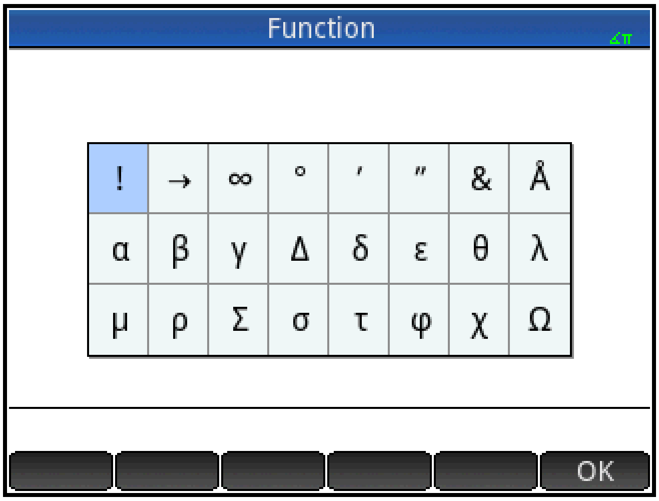
The template palette can save you a lot of time, especially with calculus calculations.

You can display the palette at any stage in defining an expression. In other words, you don’t need to start out with a template. Rather, you can embed one or more templates at any point in the definition of an expression.

Math shortcuts

As well as the math template, there are other similar screens that offer a palette of special characters. For example, pressing Shift 9 displays the special symbols palette, shown in the following figure.

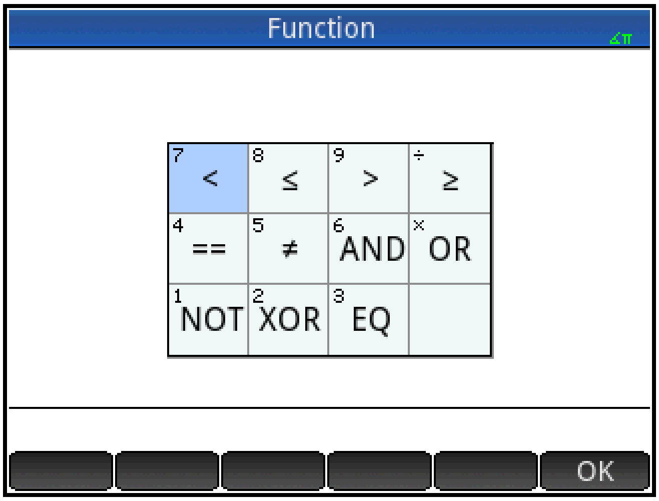
Select a character by tapping it (or scrolling to it and pressing Enter).



Function

!	→	∞	°	′	″	&	Å
α	β	γ	Δ	δ	ε	θ	λ
μ	ρ	Σ	σ	τ	φ	χ	Ω

A similar palette—the relations palette—is displayed if you press Shift 6. The palette displays operators useful in math and programming. Again, just tap the character you want.



Function

7 <	8 ≤	9 >	÷ ≥
4 ==	5 ≠	6 AND	× OR
1 NOT	2 XOR	3 EQ	

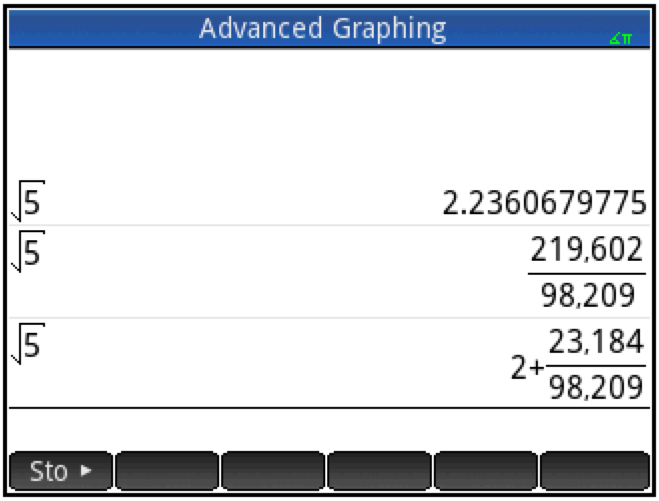
Other math shortcut keys include xtθn. Pressing this key inserts an X, T, θ, or N depending on what app you are using. (This is explained further in the chapters describing the apps.)

Similarly, pressing Shift a b/c enters a degree, minute, or second character. It enters ° if no degree symbol is part of your expression; enters ′ if the previous entry is a value in degrees; and enters ″ if the previous entry is a value in minutes.

Thus, entering 36 Shift a b/c 40 Shift a b/c 20 Shift a b/c yields 36°40′ 20 ″. See Hexagesimal numbers on page 15 for more information.

Fractions

The fraction key (a b/c) cycles through three varieties of fractional display. If the current answer is the decimal fraction 5.25, pressing a b/c converts the answer to the common fraction 21/4. If you press a b/c again, the answer is converted to a mixed number (5 + 1/4). If pressed again, the display returns to the decimal fraction (5.25).



Advanced Graphing

$\sqrt{5}$ 2.2360679775

$\sqrt{5}$ $\frac{219,602}{98,209}$

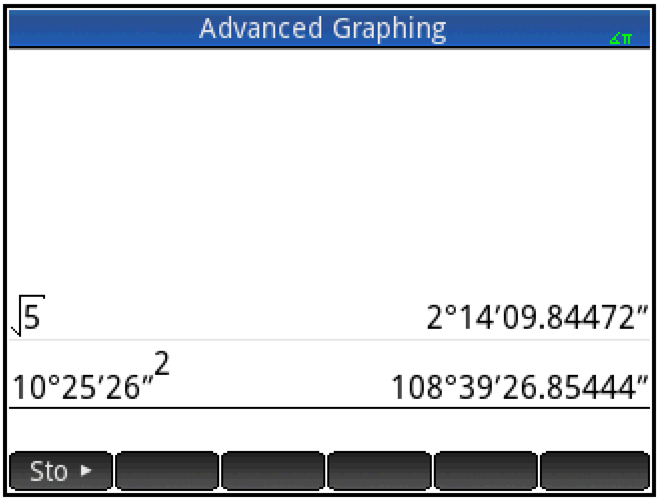
$\sqrt{5}$ $2+\frac{23,184}{98,209}$

The HP Prime will approximate fraction and mixed number representations in cases where it cannot find exact ones. For example, enter √5 to see the decimal approximation: 2.236.... Press a b/c once to see $\frac{219602}{98209}$ and again to see $2+\frac{23184}{98209}$. Pressing a b/c a third time will cycle back to the original decimal representation.

Hexagesimal numbers

Any decimal result can be displayed in hexagesimal format; that is, in units subdivided into groups of 60. This includes degrees, minutes, and seconds as well as hours, minutes, and seconds. For example, enter $\frac{11}{8}$ to see the decimal result: 1.375. Now press Shift a b/c to see 1°22′30. Press Shift a b/c again to return to the decimal representation.

The HP Prime calculator will produce the best approximation in cases where an exact result is not possible. Enter √5 to see the decimal approximation: 2.236... Press Shift a b/c to see 2°14′9.84472 .



Advanced Graphing

√5	2°14’09.84472”
$10°25’26”^2$	108°39’26.85444”

NOTE: The degree and minute entries must be integers, and the minute and second entries must be positive. Decimals are not allowed, except in the seconds.

Note too that the HP Prime calculator treats a value in hexgesimal format as a single entity. Hence, any operation performed on a hexagesimal value is performed on the entire value. For example, if you enter 10°25’26”^2, the whole value is squared, not just the seconds component. The result in this case is 108°39’26.8544”.

EEX key (powers of 10)

Numbers like 5×10^4 and 3.21×10^{-7} are expressed in *scientific notation*, that is, in terms of powers of ten. This is simpler to work with than 50 000 or 0.000 000 321. To enter numbers like these, use the EEX functionality. This is easier than using × 10 x^y .

Example: Suppose you want to calculate

$$\frac{(4 \times 10^{-13})(6 \times 10^{23})}{3 \times 10^{-5}}$$

1. Open the **Home Settings** window.

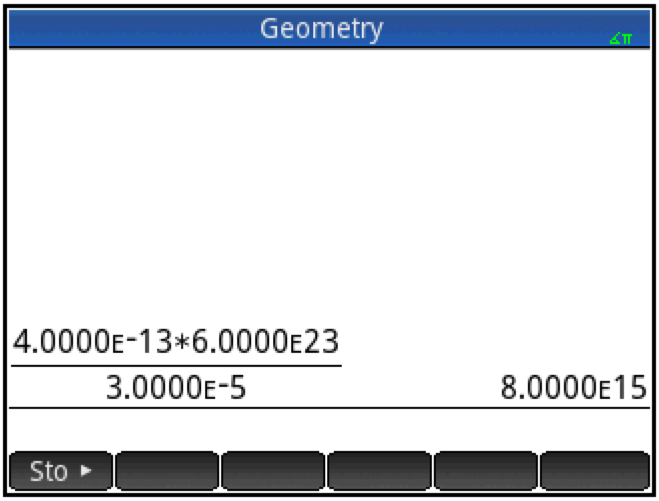
   Shift Settings

2. Select `Scientific` from the **Number Format** menu.

3. Return home by pressing Settings .

4. Enter 4 EEX +/− 13 × 6 EEX 23 ÷ 3 EEX +/− 5.

5. Press Enter.



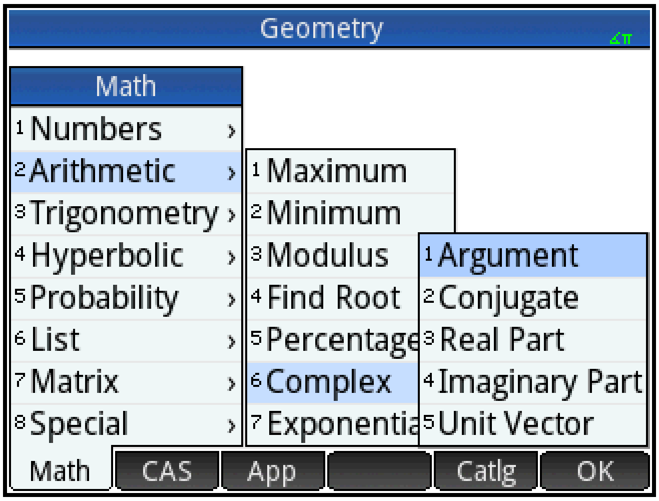
Geometry

```
4.0000E-13*6.0000E23
    3.0000E-5                 8.0000E15
```

The result is 8.0000E15. This is equivalent to 8 × 10^{15}.

Menus

A menu offers you a choice of items. As in the following example, some menus have sub-menus and sub-sub-menus.



Selecting from a menu

There are two techniques for selecting an item from a menu, as follows:

- Direct tapping
- Using the arrow keys to highlight the item you want and then either tapping OK or pressing Enter.

NOTE: The menu of buttons along the bottom of the screen can only be activated by tapping.

Shortcuts

- Press ▲ when you are at the top of the menu to immediately display the last item in the menu.
- Press ▼ when you are at the bottom of the menu to immediately display the first item in the menu.
- Press Shift ▼ to jump straight to the bottom of the menu.
- Press Shift ▲ to jump straight to the top of the menu.
- Enter the first few characters of the item's name to jump straight to that item.
- Enter the number of the item shown in the menu to jump straight to that item.

Closing a menu

A menu will close automatically when you select an item from it. If you want to close a menu without selecting anything from it, press On Off or Esc Clear .

Toolbox menus

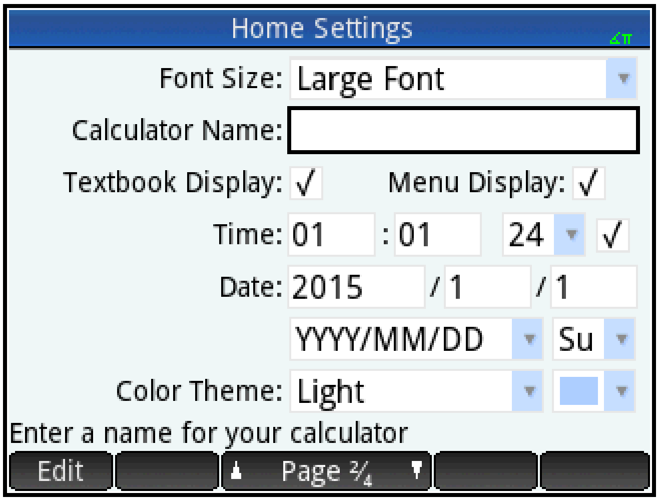
The Toolbox menus (Mem B) are a collection of menus offering functions and commands useful in mathematics and programming. The Math, CAS, and Catlg menus offer over 400 functions and commands.

Input forms

An input form is a screen that provides one or more fields for you to enter data or select an option. It is another name for a dialog box.

- If a field allows you to enter data of your choice, you can select it, add your data, and tap OK. (There is no need to tap Edit first.)
- If a field allows you to choose an item from a menu, you can tap on it (the field or the label for the field), tap on it again to display the options, and tap on the item you want. (You can also choose an item from an open list by pressing the cursor keys and pressing Enter ≈ when the option you want is highlighted.)
- If a field is a toggle field—one that is either selected or not selected—tap on it to select the field and tap on it again to select the alternate option. (Alternatively, select the field and tap √.)

The following illustration shows an input form with all three types of field.



Home Settings

Font Size: Large Font

Calculator Name:

Textbook Display: ✓ Menu Display: ✓

Time: 01 : 01 24 ✓

Date: 2015 / 1 / 1

YYYY/MM/DD Su

Color Theme: Light

Enter a name for your calculator

Calculator Name is a free-form data-entry field, **Font Size** provides a menu of options, and **Textbook Display** is a toggle field.

Resetting input form fields

To reset a field to its default value, highlight the field and press Del. To reset all fields to their default values, press Shift Esc Clear (Clear).

System-wide settings

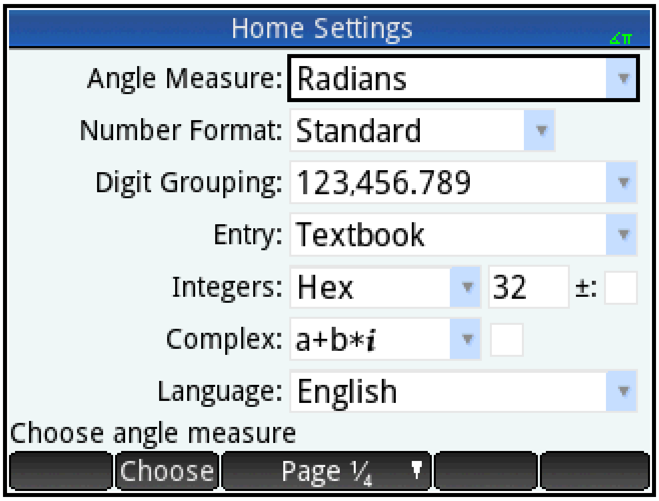
System-wide settings are values that determine the appearance of windows, the format of numbers, the scale of plots, the units used by default in calculations, and much more.

There are two system-wide settings: Home settings and CAS settings. Home settings control Home view and the apps. CAS settings control how calculations are done in the computer algebra system. CAS settings are discussed in chapter 3.

Although Home settings control the apps, you can override certain Home settings once inside an app. For example, you can set the angle measure to radians in the Home settings but choose degrees as the angle measure once inside the Polar app. Degrees then remains the angle measure until you open another app that has a different angle measure.

Home settings

You use the Home Settings input form to specify the settings for Home view (and the default settings for the apps). Press Shift Settings (Settings) to open the Home Settings input form. There are four pages of settings.



Home Settings

Angle Measure: Radians

Number Format: Standard

Digit Grouping: 123,456.789

Entry: Textbook

Integers: Hex 32 ±: □

Complex: a+b*i □

Language: English

Choose angle measure

Page 1

Setting	Options
Angle Measure	**Degrees**—360 degrees in a circle. **Radians**—2π radians in a circle. The angle mode you set is the angle setting used in both Home view and the current app. This is to ensure that trigonometric calculations done in the current app and Home view give the same result.
Number Format	The number format you set is the format used in all Home view calculations. • **Standard**—Full-precision display. • **Fixed**—Displays results rounded to a number of decimal places. If you choose this option, a new field appears for you to enter the number of decimal places. For example, 123.456789 becomes 123.46 in **Fixed 2** format. • **Scientific**—Displays results with an one-digit exponent to the left of the decimal point, and the specified number of decimal places. For example, 123.456789 becomes 1.23E2 in **Scientific 2** format. • **Engineering**—Displays results with an exponent that is a multiple of 3, and the specified number of significant digits beyond the first one. Example: 123.456E7 becomes 1.23E9 in **Engineering 2** format.
Entry	• **Textbook**—An expression is entered in much the same way as if you were writing it on paper (with some arguments above or below others). In other words, your entry could be two-dimensional. • **Algebraic**—An expression is entered on a single line. Your entry is always one-dimensional. • **RPN**—Reverse Polish Notation. The arguments of the expression are entered first followed by the operator. The entry of an operator automatically evaluates what has already been entered.

Setting	Options
Integers	Sets the default base for integer arithmetic: binary, octal, decimal, or hex. You can also set the number of bits per integer and whether integers are to be signed.
Complex	Choose one of two formats for displaying complex numbers: **(a,b)** or **a+b*i**. To the right of this field is an unnamed check box. Select it if you want to allow complex output from real input.
Language	Choose the language you want for menus, input forms, and the online help.
Decimal Mark	Select **Dot** or **Comma**. Displays a number as 12456.98 (dot mode) or as 12456,98 (comma mode). Dot mode uses commas to separate elements in lists and matrices, and to separate function arguments. Comma mode uses semicolons as separators in these contexts.

Page 2

Setting	Options
Font Size	Choose between small, medium, and large font for general display.
Calculator Name	Enter a name for the calculator.
Textbook Display	If selected, expressions and results are displayed in textbook format (that is, much as you would see in textbooks). If not selected, expressions and results are displayed in algebraic format (that is, in one-dimensional format). For example, $\begin{bmatrix}4 & 5\\6 & 2\end{bmatrix}$ is displayed as `[[4,5],[6,2]]` in algebraic format.
Menu Display	This setting determines whether the commands on the **Math** and **CAS** menus are presented descriptively or in common mathematical shorthand. The default is to provide the descriptive names for the functions. If you prefer the functions to be presented in mathematical shorthand, deselect this option.
Time	Set the time and choose a format: 24-hour or am–pm format. The check box at the far right lets you choose whether to show or hide the time on the title bar of screens.
Date	Set the date and choose a format: **YYYY/MM/DD**, **DD/MM/YYYY**, or **MM/DD/YYYY**.
Color Theme	**Light**—black text on a light background. **Dark**—white text on a dark background At the far right is an option for you to choose a color for the shading (such as the color of the highlight).

Page 3

Page 3 of the **Home Settings** input form is for setting Exam mode. This mode enables certain functions of the calculator to be disabled for a set period, with the disabling controlled by a password. This feature will primarily be of interest to those who supervise examinations and who need to ensure that the calculator is used appropriately by students sitting an examination.

Page 4

If your HP Prime calculator supports wireless connectivity, you will see a fourth page of Home Settings. Page 4 of the **Home Settings** input form is for configuring your HP Prime calculator to work with the HP Prime Wireless Kit to establish an HP Wireless Classroom Network. Visit http://www.hp.com/support for further information.

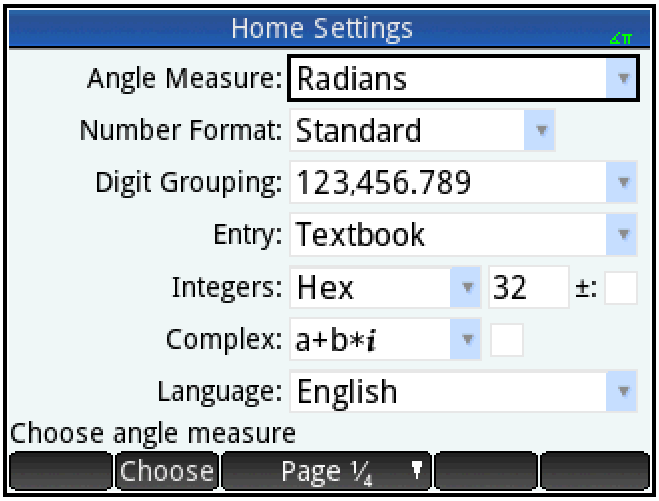
Option	Settings
Network Name	● No network available ● Network 1 ● Network 2 (and so on)
Status	● No adapter found ● Disconnected ● Connected
RF Version	● No adapter found ● Adapter firmware version

Specifying a Home setting

This example demonstrates how to change the number format from the default setting—Standard—to Scientific with two decimal places.

1. Press Shift Settings (Settings) to open the Home Settings input form.

   The **Angle Measure** field is highlighted.



Home Settings

Angle Measure: Radians

Number Format: Standard

Digit Grouping: 123,456.789

Entry: Textbook

Integers: Hex 32 ±:

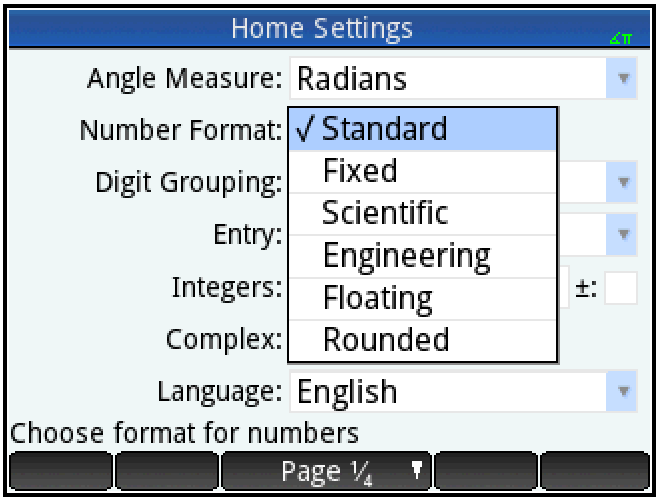
Complex: a+b*i

Language: English

Choose angle measure

2. Tap **Number Format** (either the field label or the field). This selects the field. (You can also press ▼ to select it.)

3. Tap **Number Format** again. A menu of number format options appears.



Home Settings

Angle Measure: Radians

Number Format: √Standard

Fixed

Scientific

Engineering

Floating

Rounded

Digit Grouping:

Entry:

Integers: ±:

Complex:

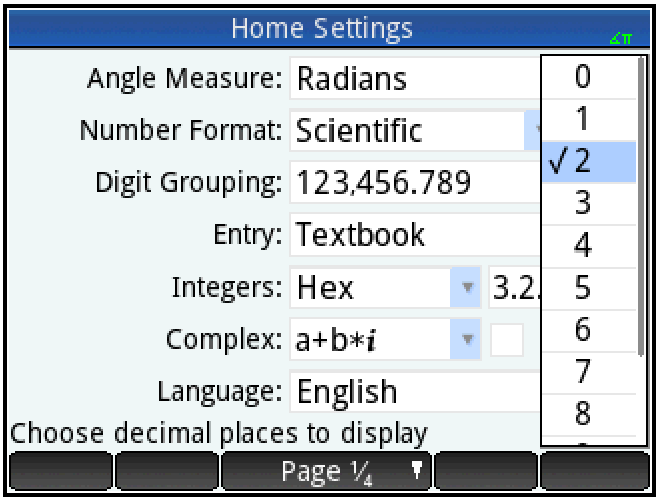
Language: English

Choose format for numbers

Page 1/4

4. Tap **Scientific**. The option is chosen and the menu closes. (You can also choose an item by pressing the cursor keys and pressing Enter ≈ when the option you want is highlighted.)

5. Notice that a number appears to the right of the **Number Format** field. This is the number of decimal places currently set. To change the number to **2**, tap the current number twice, and then tap **2** in the menu that appears.



Home Settings

Angle Measure: Radians

Number Format: Scientific

Digit Grouping: 123,456.789

Entry: Textbook

Integers: Hex 3.2

Complex: a+b*i

Language: English

Choose decimal places to display

Page 1/4

0

1

√2

3

4

5

6

7

8

6. Press Settings to return to Home view.

Mathematical calculations

The most commonly used math operations are available from the keyboard (see Math keys on page 11). Access to the rest of the math functions is via various menus (see Menus on page 17).

Note that the HP Prime represents all numbers smaller than 1 × 10^{-499} as zero. The largest number displayed is 9.99999999999 × 10^{499}. A greater result is displayed as this number.

Where to start

The home base for the calculator is the Home view (⌂ Settings). You can do all your nonsymbolic calculations here. You can also do calculations in CAS view, which uses the computer algebra system. In fact, you can use functions from the **CAS** menu (one of the Toolbox menus) in an expression you are entering in Home view, and use functions from the **Math** menu (another of the Toolbox menus) in an expression you are entering in CAS view.

Choosing an entry type

The first choice you need to make is the style of entry. The three types are as follows:

- Textbook

  $$\frac{\text{LN}(5)}{\pi}$$

  An expression is entered in much the same way as if you were writing it on paper (with some arguments above or below others). In other words, your entry could be two-dimensional, as in the example above.

- Algebraic

  LN(5)/π

  An expression is entered on a single line. Your entry is always one-dimensional.

- Reverse Polish Notation (RPN) [Not available in CAS view.]

  The arguments of the expression are entered first followed by the operator. The entry of an operator automatically evaluates what has already been entered. Thus you will need to enter a two-operator expression (as in the example above) in two steps, one for each operator:

  Step 1: 5 LN —the natural logarithm of 5 is calculated and displayed in history.

  Step 2: Shift 3 ÷ – π is entered as a divisor and applied to the previous result.

NOTE: On page 2 of the **Home Settings** screen, you can specify whether you want to display your calculations in **Textbook** format. This refers to the appearance of your calculations in the history section of both Home view and CAS view. This is a different setting from the **Entry** setting discussed above.

Entering expressions

The examples that follow assume that the entry mode is **Textbook**.

- An expression can contain numbers, functions, and variables.
- To enter a function, press the appropriate key, or open a Toolbox menu and select the function. You can also enter a function by using the alpha keys to spell out its name.
- When you have finished entering the expression, press Enter to evaluate it.

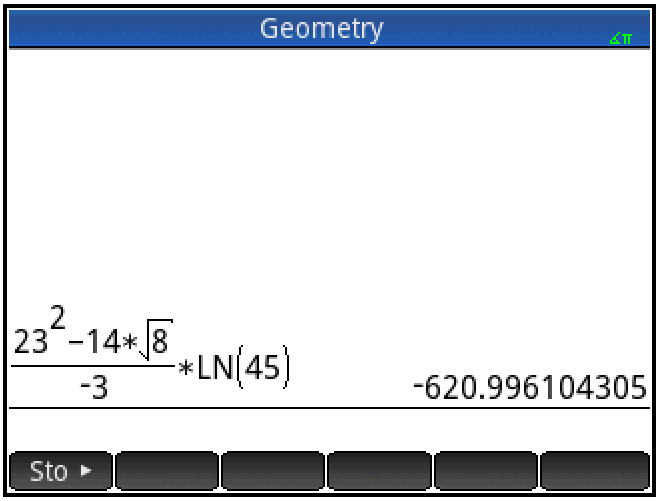
If you make a mistake while entering an expression, you can do any of the following:

- Delete the character to the left of the cursor by pressing [⌫ Del] .
- Delete the character to the right of the cursor by pressing [Shift] [⌫ Del] .
- Clear the entire entry line by pressing [On Off] or [Esc Clear] .

Example

To calculate $\frac{23^2-14\sqrt{8}}{-3}\ln(45)$:

▲ Enter [( )] 23 [x^2] [−] 14 [Shift] [x^2] 8 [▶] [▶] [÷] [+/−] 3 [▶] [LN] 45 [Enter ≈] .


Geometry

$\frac{23^2-14*\sqrt{8}}{-3}*LN(45)$ -620.996104305

This example illustrates a number of important points to be aware of, as follows:

- The importance of delimiters (such as parentheses)
- How to enter negative numbers
- The use of implied versus explicit multiplication.

Parentheses

As the example above shows, parentheses are automatically added to enclose the arguments of functions, as in `LN()`. However, you will need to manually add parentheses—by pressing [( )]—to enclose a group of objects you want operated on as a single unit. Parentheses provide a way of avoiding arithmetic ambiguity. In the example above we wanted the entire numerator divided by –3, thus the entire numerator was enclosed in parentheses. Without the parentheses, only 14√8 would have been divided by –3.

The following examples show the use of parentheses, and the use of the cursor keys to move outside a group of objects enclosed within parentheses.

Entering…	Calculates…
SIN 45 + Shift 3	$\sin(45+\pi)$
SIN 45 ▶ + Shift 3	$\sin(45)+\pi$
Shift x^2 85 ▶ × 9	$\sqrt{85}\times 9$
Shift x^2 85 × 9	$\sqrt{85\times 9}$

Algebraic precedence

The HP Prime calculator calculates according to the following order of precedence. Functions at the same level of precedence are evaluated in order from left to right.

1. Expressions within parentheses. Nested parentheses are evaluated from inner to outer.
2. !, √, reciprocal, square
3. nth root
4. Power, 10^{n}
5. Negation, multiplication, division, and modulo
6. Addition and subtraction
7. Relational operators (<, >, ≤, ≥, ==, ≠, =)
8. AND and NOT
9. OR and XOR
10. Left argument of | (where)
11. Assign to variable (:=)

Negative numbers

It is best to press +/– to start a negative number or to insert a negative sign. Pressing – instead will, in some situations, be interpreted as an operation to subtract the next number you enter from the last result. (This is explained in Reusing the last result on page 28.)

To raise a negative number to a power, enclose it in parentheses. For example, $(-5)^2 = 25$, whereas $-5^2 = -25$.

Explicit and implied multiplication

Implied multiplication takes place when two operands appear with no operator between them. If you enter AB, for example, the result is A*B. You can enter 14 Shift x^y 8 without the multiplication operator after 14. For the sake of clarity, the calculator adds the operator to the expression in history, but it is not strictly necessary when you are entering the expression. You can, though, enter the operator if you wish. The result will be the same.

Large results

If the result is too long or too tall to be seen in its entirety—for example, a many-rowed matrix—highlight it and then press Show. The result is displayed in full-screen view. You can now press ▲ and ▼ (as well as ▶ and ◀) to bring hidden parts of the result into view. Tap OK to return to the previous view.

Reusing previous expressions and results

Being able to retrieve and reuse an expression provides a quick way of repeating a calculation that requires only a few minor changes to its parameters. You can retrieve and reuse any expression that is in history. You can also retrieve and reuse any result that is in history.

To retrieve an expression and place it on the entry line for editing, do one of the following:

- Tap it twice.
- Use the cursor keys to highlight the expression and then either tap it or tap Copy.

To retrieve a result and place it on the entry line, use the cursor keys to highlight it and then tap Copy.

If the expression or result you want is not showing, press ▲ repeatedly to step through the entries and reveal those that are not showing. You can also swipe the screen to quickly scroll through history.

TIP: Pressing Shift ▲ takes you straight to the very first entry in history, and pressing Shift ▼ takes you straight to the most recent entry.

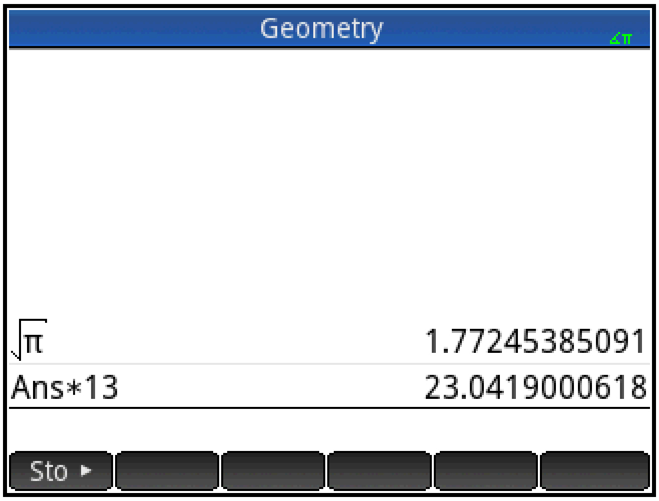
Using the clipboard

Your last four expressions are always copied to the clipboard and can easily be retrieved by pressing Shift Menu Paste. This opens the clipboard from where you can quickly choose the one you want.

NOTE: expressions and not results are available from the clipboard. Note too that the last four expressions remain on the clipboard even if you have cleared history.

Reusing the last result

Press Shift + (Ans) to retrieve your last answer for use in another calculation. Ans appears on the entry line. This is a shorthand for your last answer and it can be part of a new expression. You could now enter other components of a calculation—such as operators, number, variables, etc.—and create a new calculation.



```
√π                1.77245385091
Ans*13            23.0419000618
```

TIP: You don't need to first select `Ans` before it can be part of a new calculation. If you press a binary operator key to begin a new calculation, Ans is automatically added to the entry line as the first component of the new calculation. For example, to multiply the last answer by 13, you could enter Shift + × 13 Enter. But the first two keystrokes are unnecessary. All you need to enter is × 13 Enter.

The variable Ans is always stored with full precision whereas the results in history will only have the precision determined by the current Number Format setting (see Page 1 on page 20). In other words, when you retrieve the number assigned to Ans, you get the result to its full precision; but when you retrieve a number from history, you get exactly what was displayed.

You can repeat the previous calculation simply by pressing Enter. This can be useful if the previous calculation involved Ans. For example, suppose you want to calculate the nth root of 2 when n is 2, 4, 8, 16, 32, and so on.

1. Calculate the square root of 2.

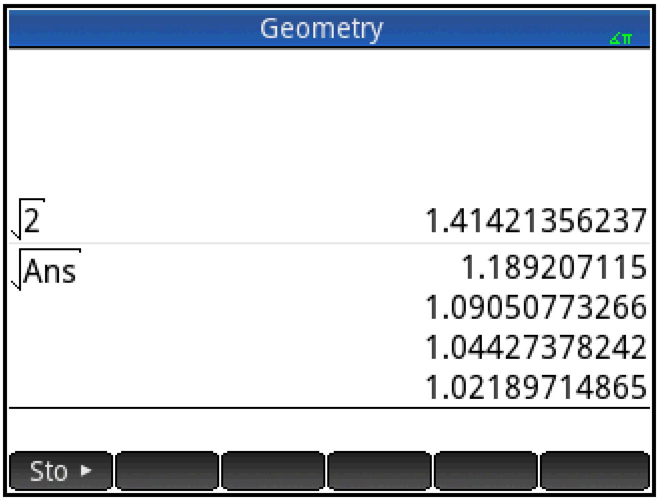
   Shift x^2 2 Enter

2. Enter √Ans.

   Shift x^2 Shift + Enter

   This calculates the fourth root of 2.

3. Press Enter repeatedly. Each time you press, the root is twice the previous root. The last answer shown in the following illustration is $\sqrt[32]{2}$.



Geometry

$\sqrt{2}$	1.41421356237
$\sqrt{Ans}$	1.189207115
	1.09050773266
	1.04427378242
	1.02189714865

Reusing an expression or result from the CAS

When you are working in Home view, you can retrieve an expression or result from the CAS by tapping Menu and selecting **Get from CAS**. The CAS opens. Press ▲ or ▼ until the item you want to retrieve is highlighted, and then press Enter. The highlighted item is copied to the cursor point in Home view.

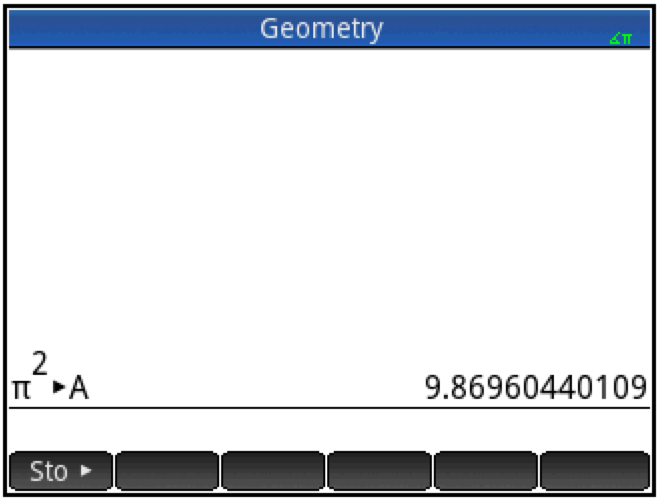
Storing value in a variable

You can store a value in a variable (that is, assign a value to a variable). Then when you want to use that value in a calculation, you can refer to it by the variable's name. You can create your own variables, or you can take advantage of the built-in variables in Home view (named A to Z and θ) and in the CAS (named a to z, and a few others). CAS variables can be used in calculations in Home view, and Home variables can be used in calculations in the CAS. There are also built-in app variables and geometry variables. These can also be used in calculations.

Example: To assign π^2 to the variable A:

Shift 3 x^2 Sto ▸ ALPHA Vars Enter

Your stored value appears as shown in the following figure. If you then wanted to multiply your stored value by 5, you could enter: ALPHA Vars × 5 Enter.



```
Geometry

π²▸A                9.86960440109

Sto ▸
```

You can also create your own variables in Home view. For example, suppose you wanted to create a variable called ME and assign π^2 to it. You would enter:

Shift 3 x^2 Sto ▸ ALPHA +/– ALPHA a b/c Enter

A message appears asking if you want to create a variable called `ME`. Tap OK or press Enter to confirm your intention. You can now use that variable in subsequent calculations: `ME*3` will yield 29.6088132033, for example.

You can also create variables in CAS view in the same way. However, the built-in CAS variables must be entered in lowercase. However, the variables you create yourself can be uppercase or lowercase.

As well as built-in Home and CAS variables, and the variables you create yourself, each app has variables that you can access and use in calculations.

Complex numbers

You can perform arithmetic operations using complex numbers. Complex numbers can be entered in the following forms in Textbook mode, where *x* is the real part, *y* is the imaginary part, and *i* is the imaginary constant, √-1.

- *(x, y)*
- *x + yi* (except in RPN mode)
- *x – yi* (except in RPN mode)
- *x + iy* (except in RPN mode)
- *x – iy* (except in RPN mode)

In RPN mode, complex numbers must be entered enclosed in single quotes and require explicit multiplication. For example, ‘3 – 2 * *i*’.

To enter *i*:

▲ Press ALPHA Shift TAN.

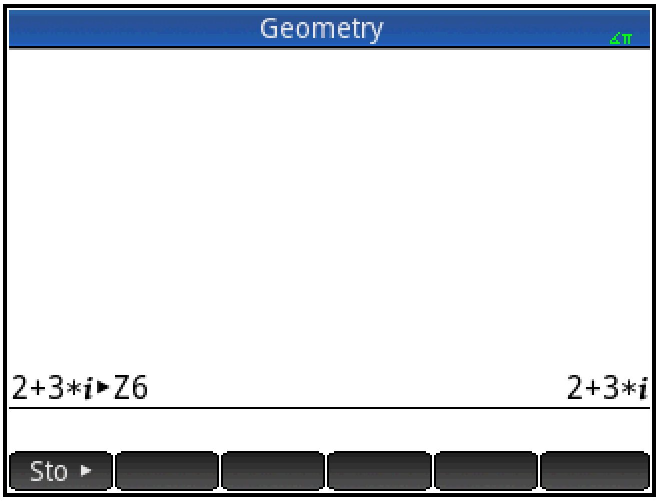
– or –

Press Shift 2.

There are 10 built-in variables available for storing complex numbers. These are labeled `Z0` to `Z9`. You can also assign a complex number to a variable you create yourself.

To store a complex number in a variable, enter the complex number, press Sto ▸, enter the variable that you want to assign the complex number to, and then press Enter. For example, to store 2 + 3*i* in variable `Z6`:

( ) 2 , 3 ▶ Sto ▸ ALPHA 2 6 Enter



Geometry

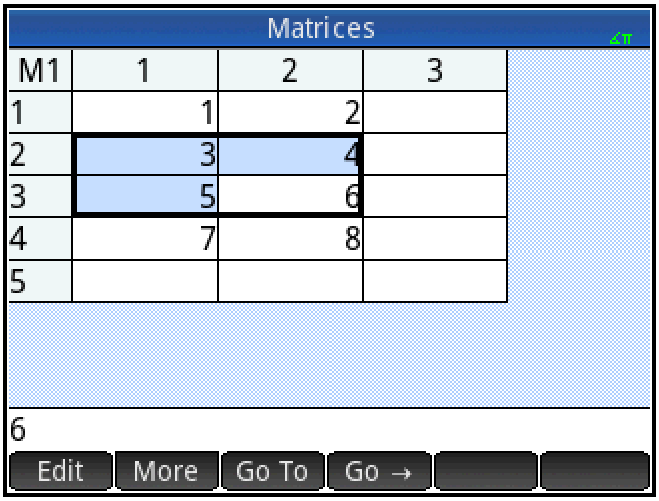
```
2+3*i▸Z6                2+3*i
```

Copy and paste

Shift View Copy copies the selected item to the HP Prime clipboard. Shift Menu Paste opens the clipboard and enables you to select a clipboard item and paste it to the current cursor location.

In the List Editor, you can select part of a list, an entire list, or a rectangular array of elements from multiple lists. This selection can then be copied and pasted in either the Matrix Editor or the Numeric view of the Spreadsheet, Statistics 1Var, or Statistics 2Var apps. Similarly, in the Matrix Editor, you can select one or more rows, one or more columns, a sub-matrix, or the entire matrix. This selection can then be copied and pasted in either the List Editor or the Numeric view of the three previously listed apps.

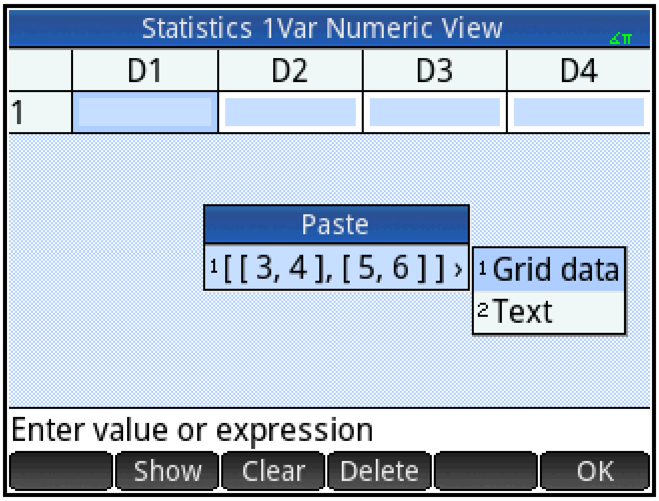
For example, in the following figure, a 2x2 array has been selected in the Matrix editor and copied to the clipboard.



M1	1	2	3
1	1	2	
2	3	4	
3	5	6	
4	7	8	
5			

6

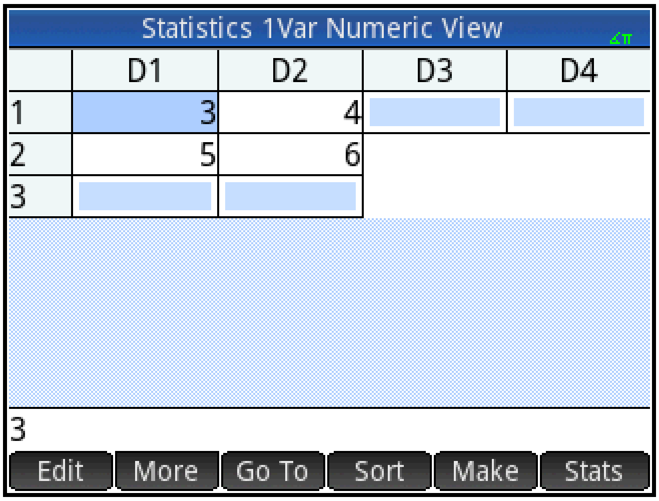
In the following figure, that array is being pasted as grid data into the Numeric view of the Statistics 1Var app.



	D1	D2	D3	D4
1				

Enter value or expression

In the following figure, that array is pasted into the Numeric view of the Statistics 1Var app.



	D1	D2	D3	D4
1	3	4		
2	5	6		
3				

3

Generally, the copy-and-paste function allows you to transfer numbers and expressions throughout the calculator software.

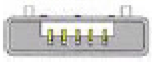
To continue the previous example, tap Calc to calculate summary statistics for the two data points in column D1. Tap the sample standard deviation, and then press Shift View Copy to copy it to the clipboard. Press ⌂ Settings to enter Home view and press Shift Menu Paste to copy the sample standard deviation to the command line. Press x^2 to square it and press Enter ≈ to see the result.

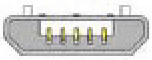
Using this same copy-and-paste technique, you can perform other operations such as copying values and pasting them into the Xmin and Xtick boxes in the Plot Setup view.

Sharing data

As well as giving you access to many types of mathematical calculations, the HP Prime calculator enables you to create various objects that can be saved and used over and over again. For example, you can create apps, lists, matrices, programs, and notes. You can also send these objects to other HP Prime calculators. Whenever you encounter a screen with Send as a menu item, you can select an item on that screen to send it to another HP Prime calculator.

You use one of the supplied USB cables to send objects from one HP Prime to another. This is the micro-A–micro B USB cable. Note that the connectors on the ends of the USB cable are slightly different. The micro-A connector has a rectangular end and the micro-B connector has a trapezoidal end. To share objects with another HP Prime, the micro-A connector must be inserted into the USB port on the sending calculator, with the micro-B connector inserted into the USB port on the receiving calculator.


Micro-A: sender


Micro-B: receiver

General procedure

The general procedure for sharing objects is as follows:

1. Navigate to the screen that lists the object you want to send.

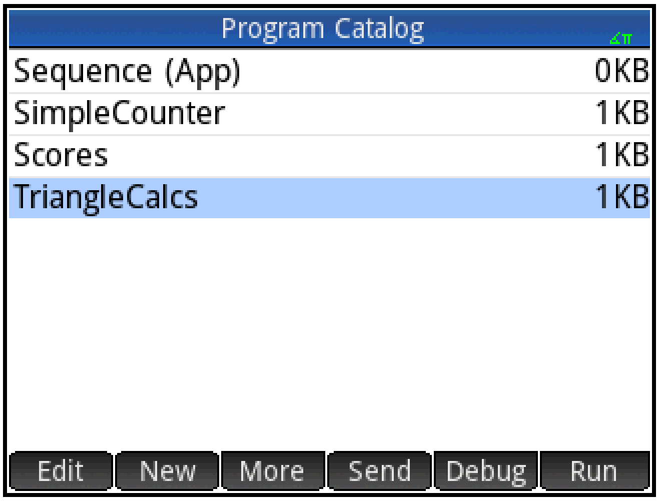
   This will be the Application Library for apps, the List Catalog for lists, the Matrix Catalog for matrices, the Program Catalog for programs, and the Notes Catalog for notes.

2. Connect the USB cable between the two calculators.

   The micro-A connector—with the rectangular end—must be inserted into the USB port on the sending calculator.

3. On the sending calculator, highlight the object you want to send and tap Send.

In the following illustration, a program named **TriangleCalcs** has been selected in the Program Catalog and will be sent to the connected calculator when Send is tapped.



Program Catalog

Sequence (App)	0KB
SimpleCounter	1KB
Scores	1KB
TriangleCalcs	1KB

Edit | New | More | Send | Debug | Run

Using Memory Manager

Memory Manager contains a list of catalogs, the Home and CAS histories, the user variables, and backups.

▲ To open Memory Manager, press Shift Mem.

To use Memory Manager:

▲ Select one of the following menu buttons:

- Info—Displays the available memory and storage space.
- Clone—Clones the HP Prime calculator to an attached HP Prime calculator.
- Send—Sends all data in a selected category (such as Lists or Matrices) to an attached HP Prime calculator.
- View—Opens the selected catalog. You can also open a catalog by pressing Enter ≈. In the catalog, you can delete unnecessary objects.

Backups catalog

The Backups catalog can be used to back up or restore your HP Prime calculator without connecting it to a computer.

To open the Backups catalog:

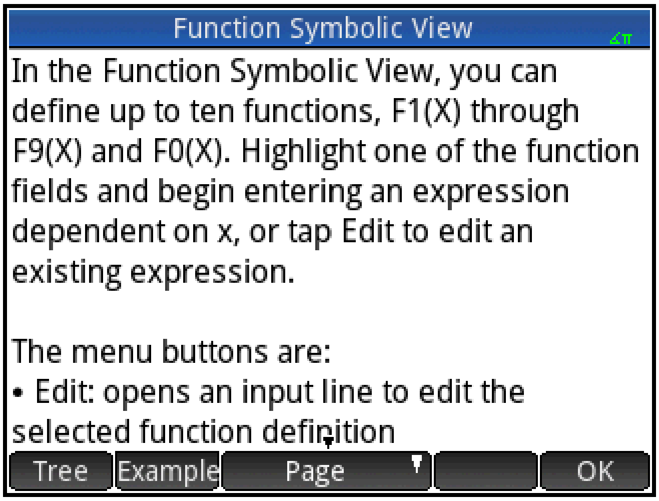
1. Open Memory Manager.
2. Tap **Backups**, and then tap View.

The following options are available:

- Restore—Restores the HP Prime calculator using the selected backup file.
- Delete—Deletes the selected backup file. You can also press ⌫ Del to delete the selected backup file.
- New—Creates a new backup file using the current state of the HP Prime calculator. By default, the name of the backup file includes the date.

Online help

The HP Prime calculator has an extensive online help system that is context sensitive. Generally, you can view context-sensitive help for each app, each app view, each dedicated editor (List, Matrix, and so on), and each function or command. Press Help User to open the online help that relates to the current context. For example, if you open the Symbolic view in the Function app and press Help User, the following help page is displayed.



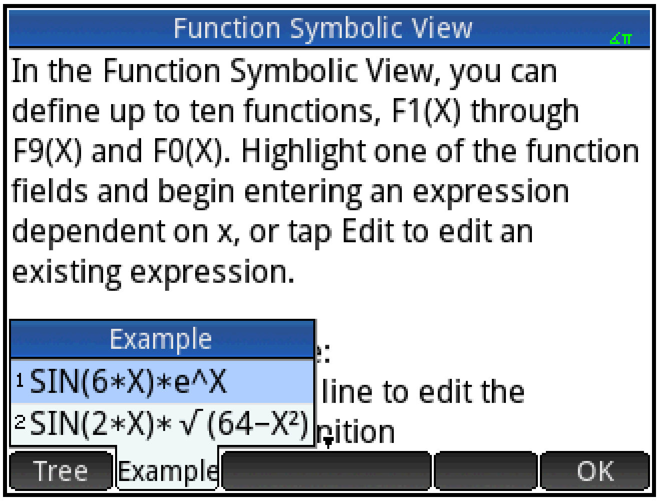
Function Symbolic View

In the Function Symbolic View, you can define up to ten functions, F1(X) through F9(X) and F0(X). Highlight one of the function fields and begin entering an expression dependent on x, or tap Edit to edit an existing expression.

The menu buttons are:

• Edit: opens an input line to edit the selected function definition

Many of the menu pages have the Example menu key available. Tap this key to paste an example into the current cursor location. For example, tap Example and then tap the first example in the list: SIN(6*X)*e^X.



Function Symbolic View

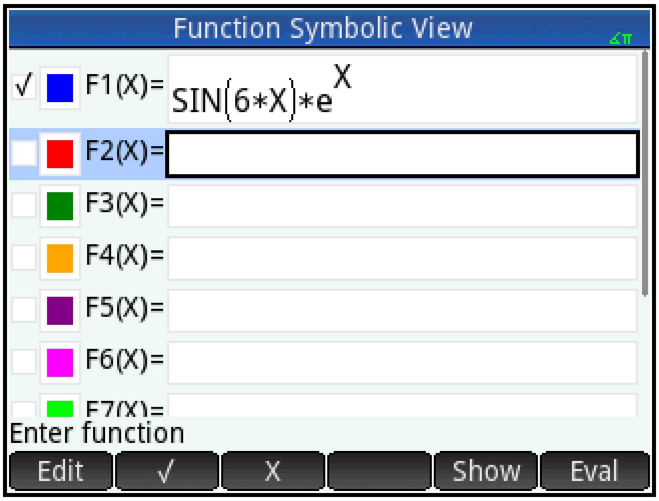
In the Function Symbolic View, you can define up to ten functions, F1(X) through F9(X) and F0(X). Highlight one of the function fields and begin entering an expression dependent on x, or tap Edit to edit an existing expression.

Example

1 SIN(6*X)*e^X

2 SIN(2*X)*√(64–X²)

The function is pasted into the command line in the Symbolic view of the Function app. Press Enter ≈ to paste this function into F1(X).



Function Symbolic View

√ 🟦 F1(X)= $SIN(6*X)*e^{X}$

🟥 F2(X)=

🟩 F3(X)=

🟧 F4(X)=

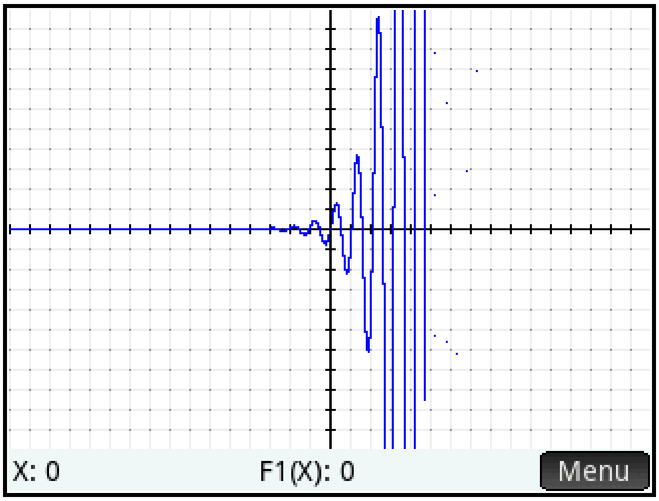
🟪 F5(X)=

🟪 F6(X)=

🟩 F7(X)=

Enter function

Press Plot Setup to see the graph.



When a help page is displayed, you can tap Tree to display a hierarchial tree display of the entire help system. Tap an entry and then tap OK to view the page. Tap the + sign to expand any entry to view its subentries. Tap Keys and then press any key (or any shifted key combination) to display the help for that key.

Extensive help is available for each command. The help provides the syntax of each command, a description of the command, and an example. If you enter a command but need the syntax, press Help User to display its syntax. For example, if you have entered `int( )` in CAS view, pressing Help User displays help on the integral command.

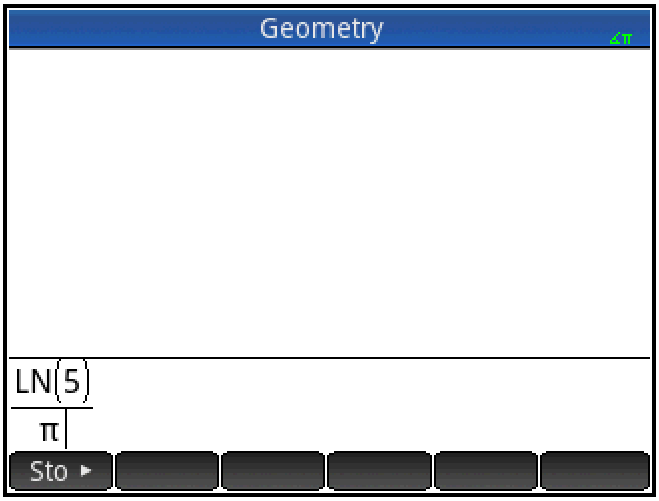
Finally, if you have online help open, you can tap Search and enter a keyword to search help for that keyword.

3 Reverse Polish Notation (RPN)

The HP Prime calculator provides you with three ways of entering objects in Home view, as follows:

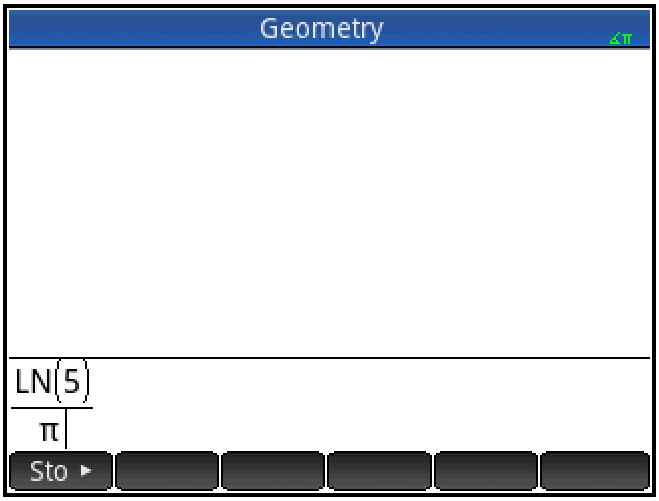
- Textbook

  An expression is entered in much the same way was if you were writing it on paper (with some arguments above or below others). In other words, your entry could be two-dimensional, as in the following example:

- Algebraic

  An expression is entered on a single line. Your entry is always one-dimensional. The same calculation as above would appear like this is algebraic entry mode:

- Reverse Polish Notation (RPN)

  The arguments of the expression are entered first followed by the operator. The entry of an operator automatically evaluates what has already been entered. Thus you will need to enter a two-operator expression (as in the example above) in two steps, one for each operator:

Step 1: 5 LN —the natural logarithm of 5 is calculated and displayed in history.

Step 2: Shift 3 ÷ – π is entered as a divisor and applied to the previous result.

You select your preferred entry method from page 1 of the **Home Settings** screen (Shift Settings). Select settings as normal.

RPN is available in Home view, but not in CAS view.

The same entry-line editing tools are available in RPN mode as in algebraic and textbook mode. You can edit an expression on the entry line using the following keys:

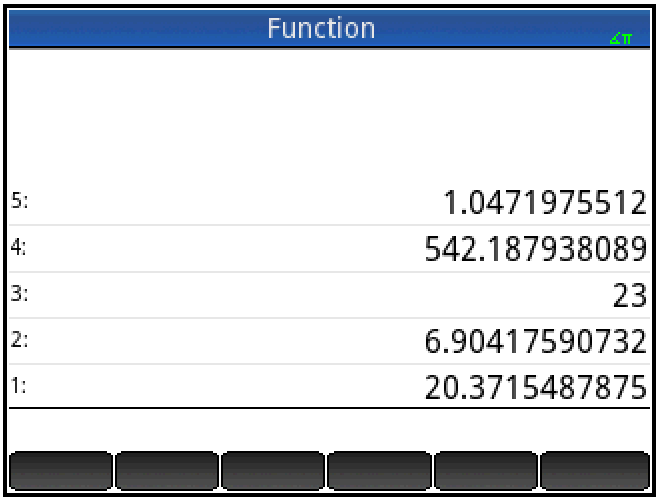
- Press Del to delete the character to the left of the cursor.
- Press Shift Del to delete the character to the right of the cursor.
- Press Shift Esc to clear the entire entry line.

If there is no expression on the entry line, you can press Shift Esc to clear the entire history.

History in RPN mode

The results of your calculations are kept in history. This history is displayed above the entry line (and by scrolling up to calculations that are no longer immediately visible). The calculator offers three histories: one for the CAS view and two for Home view. The two histories in Home view are:

- non-RPN—visible if you have chosen algebraic or textbook as your preferred entry technique
- RPN—visible only if you have chosen RPN as your preferred entry technique. The RPN history is also called the stack. As shown in the following illustration, each entry in the stack is given a number. This is the stack level number.



Function

```
5:                 1.0471975512
4:                 542.187938089
3:                            23
2:                 6.90417590732
1:                 20.3715487875
```

As more calculations are added, an entry’s stack level number increases.

If you switch from RPN to algebraic or textbook entry, your history is not lost. It is just not visible. If you switch back to RPN, your RPN history is redisplayed. Likewise, if you switch to RPN, your non-RPN history is not lost.

When you are not in RPN mode, your history is ordered chronologically: oldest calculations at the top, most recent at the bottom. In RPN mode, your history is ordered chronologically by default, but you can change the order of the items in history. (This is explained in Manipulating the stack on page 42.)

Reusing results

There are two ways to reuse a result in history. Method 1 deselects the copied result after copying; Method 2 keeps the copied item selected.

Method 1

1. Select the result to be copied. You can do this by pressing ▲ or ▼ until the result is highlighted, or by tapping on it.
2. Press Enter. The result is copied to the entry line and is deselected.

Method 2

1. Select the result to be copied. You can do this by pressing the up arrow or the down arrow until the result is highlighted, or by tapping on it.
2. Tap Stack and select **ECHO**. The result is copied to the entry line and remains selected.

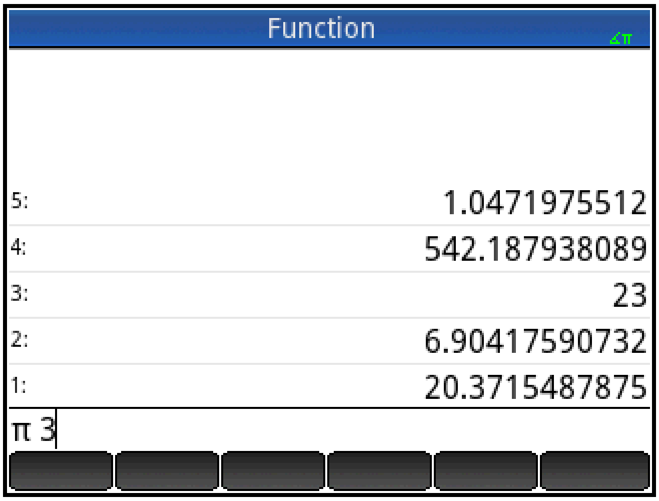
Note that while you can copy an item from the CAS history to use in a Home calculation (and copy an item from the Home history to use in a CAS calculation), you cannot copy items from or to the RPN history. You can, however, use CAS commands and functions when working in RPN mode.

Sample calculations

The general philosophy behind RPN is that arguments are placed before operators. The arguments can be on the entry line (each separated by a space) or they can be in history. For example, to multiply π by 3, you could enter the following on the entry line:

Shift 3 ␣ 3

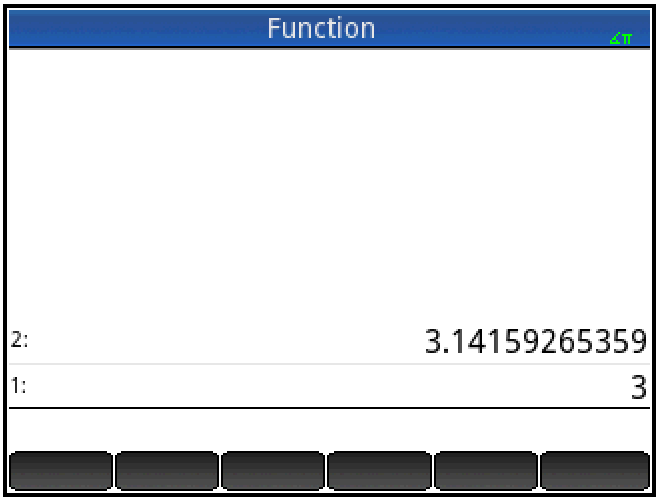
Then, enter the operator (×). Thus, your entry line would look like this before entering the operator:



Function

```
5:                1.0471975512
4:               542.187938089
3:                          23
2:               6.90417590732
1:               20.3715487875
π 3
```

However, you could also have entered the arguments separately and then, with a blank entry line, entered the operator (×). Your history would look like the following figure before entering the operator:



Function

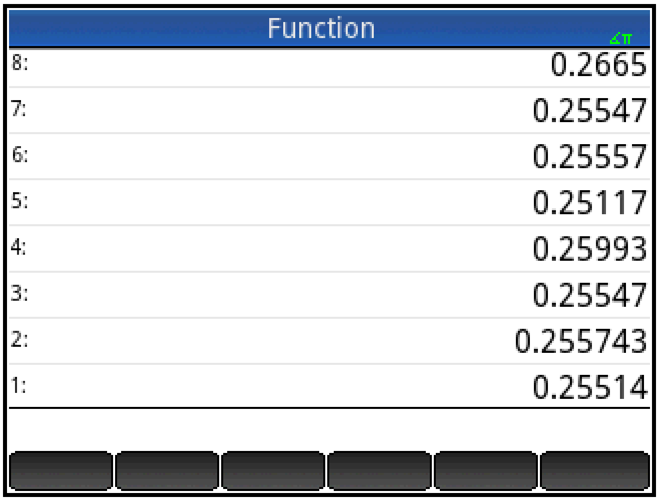
```
2:               3.14159265359
1:                           3
```

To get the same result, you could also press Shift 3 Enter to enter the value π on stack level one, and then press 3 × .

If there are no entries in history and you enter an operator or function, an error message appears. An error message will also appear if there is an entry on a stack level that an operator needs but it is not an appropriate argument for that operator. For example, pressing COS when there is a string on level 1 displays an error message.

An operator or function will work only on the minimum number of arguments necessary to produce a result. Thus if you enter on the entry line 2 4 6 8 and press ×, stack level 1 shows 48. Multiplication needs only two arguments, so the two arguments last entered are the ones that get multiplied. The entries 2 and 4 are not ignored: 2 is placed on stack level 3 and 4 on stack level 2.

Where a function can accept a variable number of arguments, you need to specify how many arguments you want it to include in its operation. You do this by specifying the number in parentheses straight after the function name. You then press Enter to evaluate the function. For example, suppose your stack looks like following:

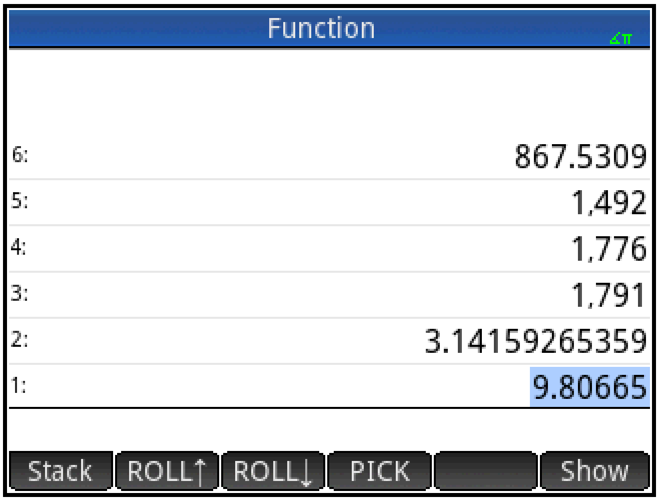


```
Function
8:                 0.2665
7:                0.25547
6:                0.25557
5:                0.25117
4:                0.25993
3:                0.25547
2:               0.255743
1:                0.25514
```

Suppose further that you want to determine the minimum of just the numbers on stack levels 1, 2, and 3. You choose the **MIN** function from the **MATH** menu and complete the entry as **MIN(3)**. When you press Enter, the minimum of just the last three items on the stack is displayed.

Manipulating the stack

A number of stack-manipulation options are available. Most appear as menu items across the bottom the screen. To see these items, you must first select an item in history:



```
6:                 867.5309
5:                    1,492
4:                    1,776
3:                    1,791
2:            3.14159265359
1:                  9.80665
```

PICK

Copies the selected item to stack level 1. The item below the one that is copied is then highlighted. Thus if you tapped PICK four times, four consecutive items will be moved to the bottom four stack levels (levels 1–4).

ROLL

There are two roll commands:

- Tap ROLL↑ to move the selected item to stack level 1. This is similar to pick, but pick duplicates the item, with the duplicate being placed on stack level 1. However, roll does not duplicate an item. It simply moves it.
- Tap ROLL↓ to move the item on stack level 1 to the currently highlighted level.

Swap

You can swap the position of the objects on stack level 1 with those on stack level 2. Just press , . The level of other objects remains unchanged. Note that the entry line must not be active at the time, otherwise a comma will be entered.

Stack

Tapping Stack displays further stack-manipulation tools.

DROPN

Deletes all items in the stack from the highlighted item down to and including the item on stack level 1. Items above the highlighted item drop down to fill the levels of the deleted items.

If you just want to delete a single item from the stack, see Delete an item on page 45.

DUPN

Duplicates all items between (and including) the highlighted item and the item on stack level 1. If, for example, you have selected the item on stack level 3, selecting **DUPN** duplicates it and the two items below it, places them on stack levels 1 to 3, and moves the items that were duplicated up to stack levels 4 to 6.

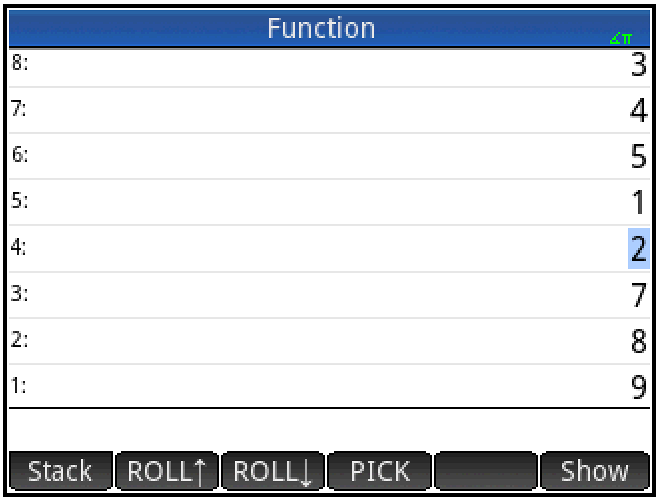
Echo

Places a copy of the selected result on the entry line and leaves the source result highlighted.

→LIST

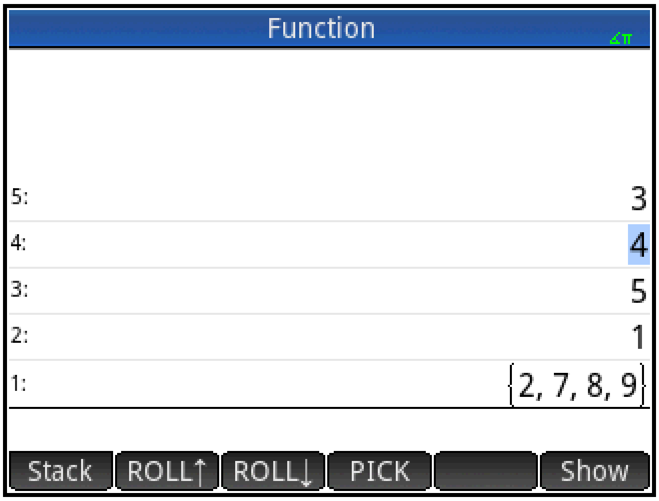
Creates a list of results, with the highlighted result the first element in the list and the item on stack level 1 the last.

Figure 3-1 Before



```
8:                  3
7:                  4
6:                  5
5:                  1
4:                  2
3:                  7
2:                  8
1:                  9
```

Figure 3-2 After



```
5:                  3
4:                  4
3:                  5
2:                  1
1:       {2, 7, 8, 9}
```

Show an item

To show a result in full-screen textbook format, tap Show.

Tap OK to return to the history.

Delete an item

To delete an item from the stack:

1. Select it. You can do this by pressing ▲ or ▼ until the item is highlighted, or by tapping on it.

2. Press ⌫ Del .

Delete all items

To delete all items, thereby clearing the history, press Shift Esc Clear .

4 Computer algebra system (CAS)

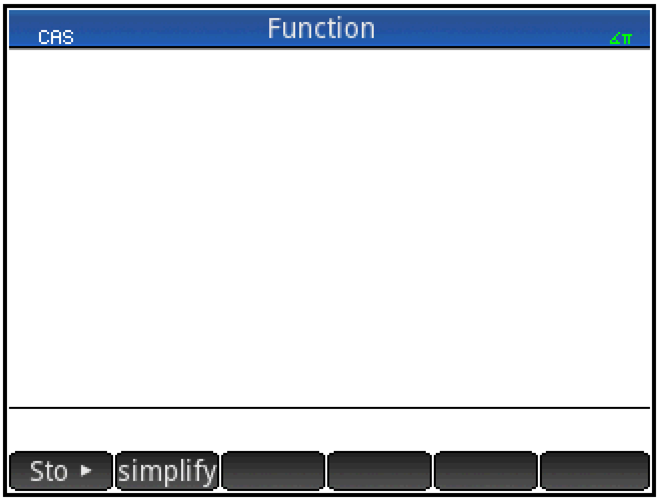
A computer algebra system (CAS) enables you to perform symbolic calculations. By default, CAS works in exact mode, giving you infinite precision. On the other hand, non-CAS calculations, such as those performed in HOME view or by an app, are numerical calculations and are often approximations limited by the precision of the calculator (to 12 significant digits in the case of the HP Prime). For example, 1/3+2/7 yields the approximate answer .619047619047 in Home view (with Standard numerical format), but yields the exact answer 13/21 in the CAS.

The CAS offers many hundreds of functions, covering algebra, calculus, equation solving, polynomials, and more. You select a function from the **CAS** menu, one of the Toolbox menus. For more information on the CAS commands, see *CAS menu* in the *Functions and commands* chapter.

CAS view

CAS calculations are done in CAS view. CAS view is almost identical to Home view. A history of calculations is built and you can select and copy previous calculations just as you can in Home view, as well as store objects in variables.

To open CAS view, press CAS Settings . **CAS** appears in white at the left of the title bar to indicate that you are in CAS view rather than Home view.



The menu buttons in CAS view are:

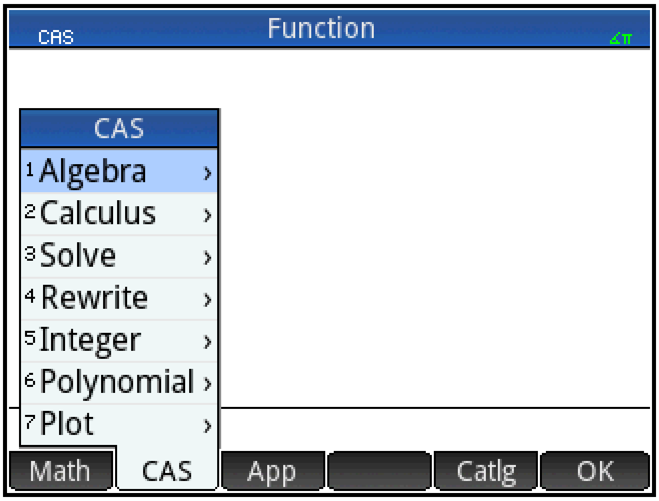
- Sto ▸—assigns an object to a variable.
- simplify—applies common simplification rules to reduce an expression to its simplest form. For example, `simplify(e^(a + LN(b*e^c)))` yields `b*EXP(a)*EXP(c)`.
- Copy—copies a selected entry in history to the entry line.
- Show—displays the selected entry in full-screen mode, with horizontal and vertical scrolling enabled. The entry is also presented in textbook format.

CAS calculations

With one exception, you perform calculations in CAS view just as you do in Home view. (The exception is that there is no RPN entry mode in CAS view, just algebraic and textbook modes). All the operator and function keys work in the same way in CAS view as Home view (although all the alpha characters are lowercase rather than uppercase). But the primary difference is that the default display of answers is symbolic rather than numeric.

You can also use the template key (Units C) to help you insert the framework for common calculations (and for vectors and matrices).

The most commonly used CAS functions are available from the CAS menu. To display the menu, press the Mem B button. (If the CAS menu is not open by default, tap CAS.) Other CAS commands are available from the Catlg menu (also a Toolbox menu).



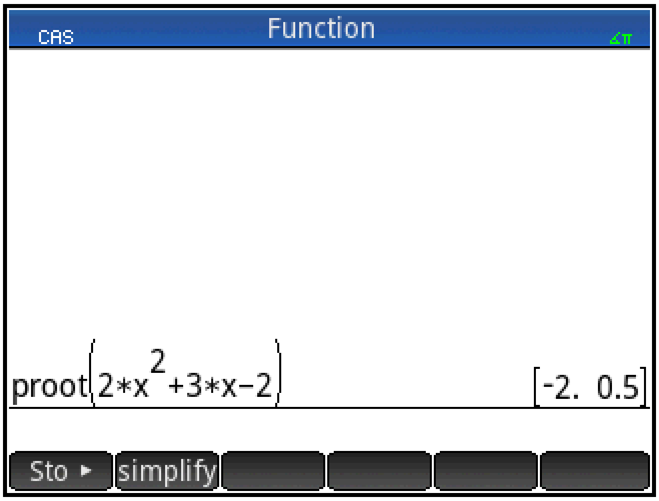
To choose a function, select a category and then a command.

Example 1

To find the roots of $2x^2 + 3x - 2$:

1. With the CAS menu open, select **Polynomial**, and then select **Find Roots**.

   The function `proot()` appears on the entry line.

   ```
   proot(2*x^2+3*x-2)                [-2.  0.5]
   ```

2. Between the parentheses, enter: 2 ALPHA × x^2 + 3 ALPHA × − 2.

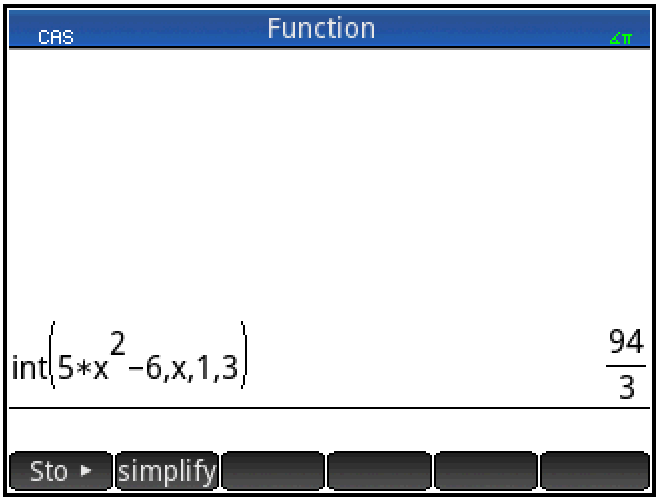
3. Press Enter.

Example 2

To find the area under the graph of $5x^2 - 6$ between x =1 and x = 3:

1. With the CAS menu open, select **Calculus** and then **Integrate**.

   The function `int()` appears on the entry line.



CAS Function

int(5*x^2 −6,x,1,3) $\frac{94}{3}$

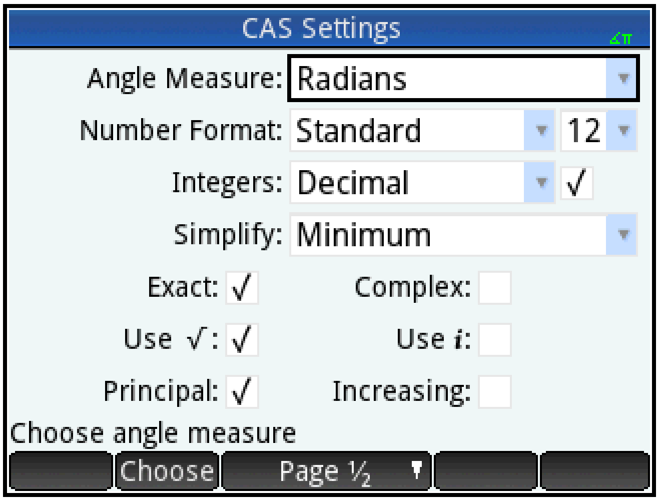
2. Between the parentheses, enter: 5 ALPHA × x² − 6 , ALPHA × , 1 , 3.

3. Press Enter.

Settings

Various settings allow you to configure how the CAS works. To display the settings, press Shift CAS.

The modes are spread across two pages.



CAS Settings

Angle Measure: Radians

Number Format: Standard 12

Integers: Decimal ☑

Simplify: Minimum

Exact: ☑ Complex: □

Use √: ☑ Use *i*: □

Principal: ☑ Increasing: □

Choose angle measure

Page 1

Setting	Purpose
Angle Measure	Select the units for angle measurements: **Radians** or **Degrees**.
Number Format (first drop-down list)	Select the number format for displayed solutions: **Standard** or **Scientific** or **Engineering**.
Number Format (second drop-down list)	Select the number of digits to display in approximate mode (mantissa + exponent).
Integers (drop-down list)	Select the integer base: **Decimal** (base 10) **Hex** (base 16) **Octal** (base 8)
Integers (check box)	If checked, any real number equivalent to an integer in a non-CAS environment will be converted to an integer in the CAS. (Real numbers not equivalent to integers are treated as real numbers in CAS whether or not this option is selected.)
Simplify	Select the level of automatic simplification: **None**—do not simplify automatically (use simplify for manual simplification) **Minimum**—do basic simplifications (default) **Maximum**—always try to simplify
Exact	If checked, the calculator is in exact mode and solutions will be symbolic. If not checked, the calculator is in approximate mode and solutions will be approximate. For example, 26 ÷ 5 yields 26/5 in exact mode and 5.2 in approximate mode.
Complex	Select this to allow complex results in variables.
Use √	If checked, second order polynomials are factorized in complex mode or in real mode if the discriminant is positive.
Use *i*	If checked, the calculator is in complex mode and complex solutions will be displayed when they exist. If not checked, the calculator is in real mode and only real solutions will be displayed. For example, factors(x^4–1) yields (x–1),(x+1),(x+i),(x–i) in complex mode and (x–1),(x+1),(x^2+1) in real mode.
Principal	If checked, the principal solutions to trigonometric functions will be displayed. If not checked, the general solutions to trigonometric functions will be displayed.
Increasing	If checked, polynomials will be displayed with increasing powers (for example, –4+x+3x^2+x^3). If not checked, polynomials will be displayed with decreasing powers (for example, x^3+3x^2+x–4).

Page 2

Setting	Purpose
Recursive Evaluation	Specify the maximum number of embedded variables allowed in an interactive evaluation. See also Recursive Replacement.
Recursive Replacement	Specify the maximum number of embedded variables allowed in a single evaluation in a program. See also Recursive Evaluation.
Recursive Function	Specify the maximum number of embedded function calls allowed.
Epsilon	Any number smaller than the value specified for epsilon will be shown as zero.
Probability	Specify the maximum probability of an answer being wrong for non-deterministic algorithms. Set this to zero for deterministic algorithms.
Newton	Specify the maximum number of iterations when using the Newtonian method to find the roots of a quadratic.

Setting the form of menu items

One setting that affects the CAS is made outside the **CAS Settings** screen. This setting determines whether the commands on the CAS menu are presented descriptively or by their command name. Here are some examples of identical functions that are presented differently depending on what presentation mode you select:

Descriptive Name	Command Name
Factor List	ifactors
Complex Zeroes	cZeros
Groebner Basis	gbasis
Factor by Degree	factor_xn
Find Roots	proot

The default menu presentation mode provides the descriptive names for the CAS functions. If you prefer the functions to be presented by their command name, deselect the **Menu Display** option on the second page of the **Home Settings** screen.

Using an expression or result from Home view

When you are working in CAS, you can retrieve an expression or result from Home view by tapping Menu Paste and selecting **Get from Home**. Home view opens. Press ▲ or ▼ until the item you want to retrieve is highlighted, and then press Enter ≈. The highlighted item is copied to the cursor point in CAS. You can also use a copy (Shift View Copy) and paste (Shift Menu Paste) operation.

Using a Home variable in CAS

You can access Home variables from within the CAS. Home variables are assigned uppercase letters; CAS variables are assigned lowercase letters. Thus SIN(x) and SIN(X) will yield different results.

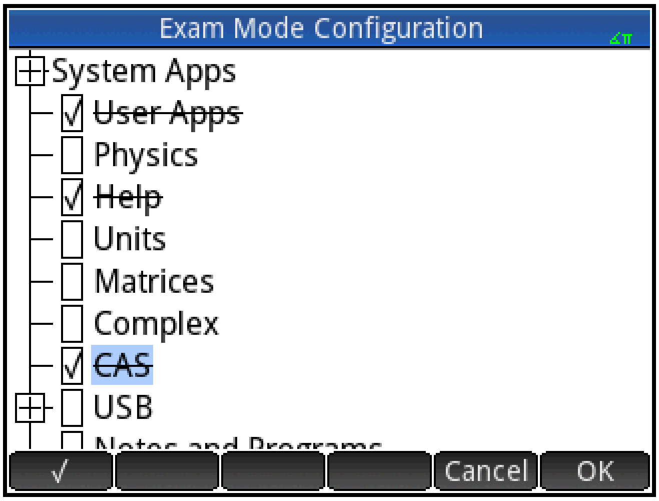
To use a Home variable in the CAS, simply include its name in a calculation. For example, suppose in Home view you have assigned variable Q to 100. Suppose too that you have assigned variable q to 1000 in the CAS. If you are in the CAS and enter 5*q, the result is 5000. If you enter 5*Q, the result is 500.

In a similar way, CAS variables can be used in calculations in Home view. Thus you can enter 5*q in Home view and get 5000, even though q is a CAS variable.

5 Exam Mode

The HP Prime calculator can be precisely configured for an examination, with any number of features or functions disabled for a set period of time. Configuring a HP Prime calculator for an examination is called exam mode configuration. You can create and save multiple exam mode configurations, each with its own subset of functionality disabled. You can set each configuration for its own time period, with or without a password. An exam mode configuration can be activated from an HP Prime calculator, sent from one HP Prime calculator to another via USB cable, or sent to one or more HP Prime calculators via the Connectivity Kit.

Exam mode configuration will be of interest primarily to teachers, proctors, and invigilators who want to ensure that the calculator is used appropriately by students sitting for an examination. In the following figure, user-customized apps, the help system, and the computer algebra system have been selected for disabling.


Exam Mode Configuration

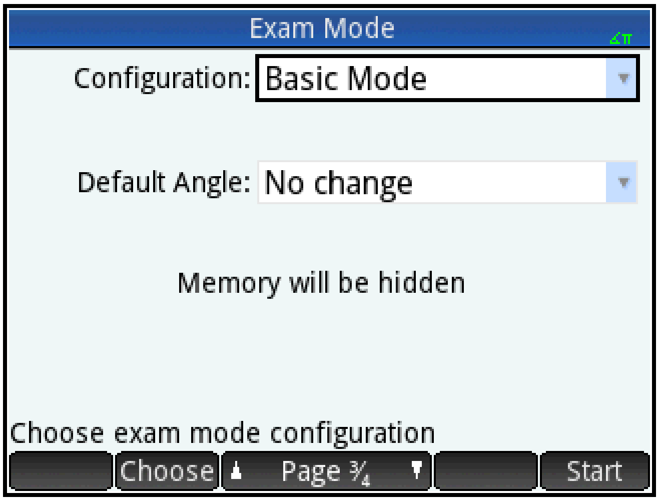
⊞ System Apps
- ☑ ~~User Apps~~
- □ Physics
- ☑ ~~Help~~
- □ Units
- □ Matrices
- □ Complex
- ☑ ~~CAS~~
- ⊞ □ USB
- □ Notes and Programs

As part of an exam mode configuration, you can choose to activate three lights on the calculator that blink periodically during exam mode. The lights are on the top edge of the calculator. The lights help the supervisor of the examination detect if any particular calculator has dropped out of exam mode. The flashing of lights on all calculators placed in exam mode are synchronized so that all blink the same pattern at the same time.

Using Basic Mode

When you first access the Exam Mode view, the Configuration field displays Basic Mode by default. Basic Mode cannot be changed by the user. If you wish to define your own exam mode configuration, change the configuration to **Custom Mode**. For more information about designing your own configuration, see Modifying the default configuration on page 55. In Basic Mode, the following settings are configured:

- The HP Prime calculator memory is hidden while exam mode is enabled.
- The green light at the top of the calculator blinks.



Exam Mode

Configuration: Basic Mode

Default Angle: No change

Memory will be hidden

Choose exam mode configuration

There is no time limit setting how long the calculator stays in Basic Mode. To exit this mode, connect the calculator to either a computer or another HP Prime calculator via the included micro USB cable.

Modifying the default configuration

You can define your own exam mode configurations after selecting **Custom Mode** in the Configuration box. If only one configuration is needed, you can simply modify the Custom Mode configuration. If you foresee the need for a number of configurations—for example, different configurations for different examinations—modify the Custom Mode configuration so that it matches the settings you will need most often, and then create other configurations for the settings you will need less often. You can access the screen for configuring and activating Custom Mode in two ways:

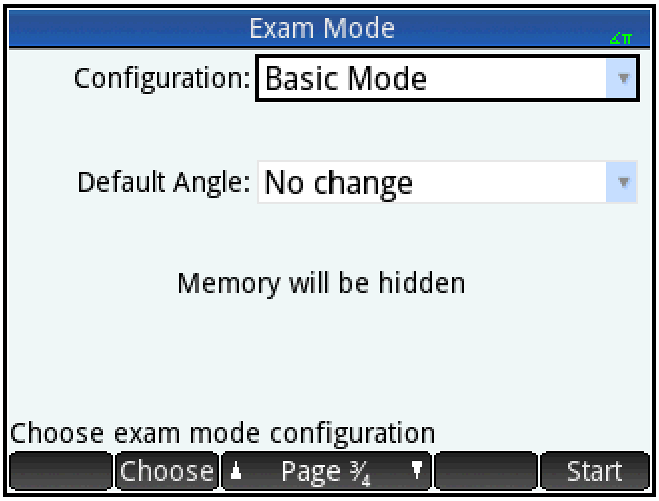
- Press On Off + a b/c or On Off + Esc Clear .
- Choose the third page of the **Home Settings** screen.

The following procedure illustrates the second method.

1. Press Shift ⌂ Settings . The **Home Settings** screen appears.
2. Tap the right side of Page 1/4 ▼.

3. Tap the right side of Page 2/4.

The **Exam Mode** screen appears. You use this screen to activate a particular configuration (just before an examination begins, for example).



Exam Mode

Configuration: Basic Mode

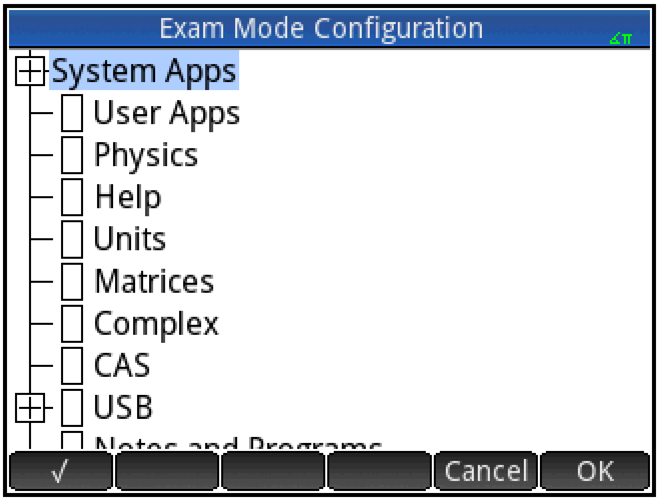
Default Angle: No change

Memory will be hidden

Choose exam mode configuration

4. Tap Choose and select **Custom Mode**.

5. Tap Config. The **Exam Mode Configuration** screen appears.



Exam Mode Configuration

⊞ System Apps
- □ User Apps
- □ Physics
- □ Help
- □ Units
- □ Matrices
- □ Complex
- □ CAS
- ⊞ □ USB
- □ Notes and Programs

6. Select those features you want disabled, and make sure that those features you don't want disabled are not selected.

An expand box at the left of a feature indicates that it is a category with sub-items that you can individually disable. (Notice that there is an expand box beside **System Apps** in the example shown above.) Tap on the expand box to see the sub-items. You can then select the sub-items individually. If you want to disable all the sub-item, just select the category.

You can select (or deselect) an option either by tapping on the check box beside it, or by using the cursor keys to scroll to it and tapping √.

7. When you have finished selecting the features to be disabled, tap OK.

If you want to activate exam mode now, continue with Activating exam mode on page 58.

Creating a new configuration

You can modify the default exam configuration when new circumstances require a different set of disabled functions. Alternatively, you can retain the default configuration and create a new configuration. When you create a new configuration, you choose an existing configuration on which to base it.

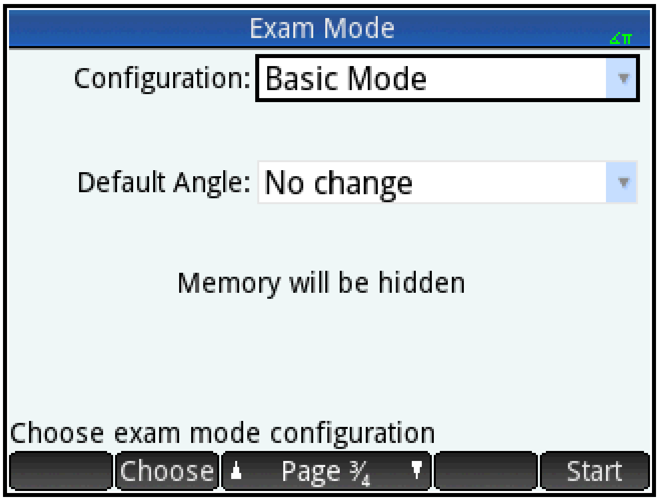
TIP: You cannot modify Basic Mode.

1. Press Shift Settings. The **Home Settings** screen appears.

2. Tap Page ¼ ▼.

3. Tap ▲ Page ²⁄₄ ▼.

   The **Exam Mode** screen appears.



Exam Mode

Configuration: Basic Mode

Default Angle: No change

Memory will be hidden

Choose exam mode configuration

4. Choose a base configuration, except for Basic Mode, from the **Configuration** list. If you have not created any exam mode configurations before, the only base configuration available is Custom Mode.

5. Tap More, select **Copy** from the menu and enter a name for the new configuration.

6. Tap OK twice.

7. Tap Config. The **Exam Mode Configuration** screen appears.

8. Select those features you want disabled, and make sure that those features you don't want disabled are not selected.

9. When you have finished selecting the features to be disabled, tap OK.

   Note that you can create exam mode configurations using the Connectivity Kit in much the same way you create them on an HP Prime. You can then activate them on multiple HP Primes, either via USB or by broadcasting them to a class using the wireless modules. For more information, install and launch the HP Connectivity Kit that came on your product CD. From the Connectivity Kit menu, click **Help** and select **HP Connectivity Kit User Guide**.

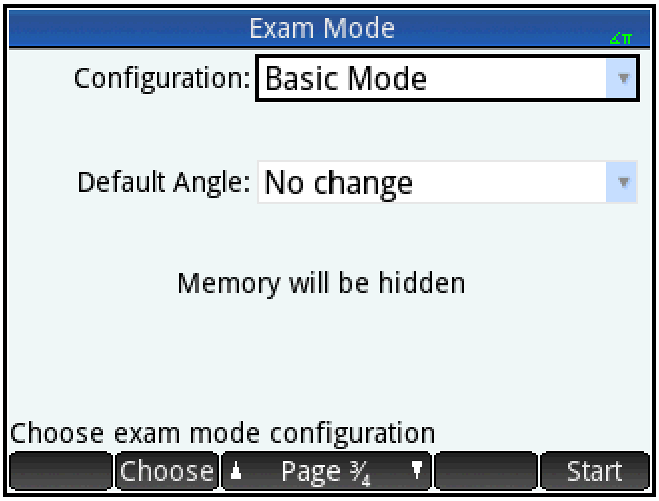
   If you want to activate exam mode now, continue with Activating exam mode on page 58.

Activating exam mode

When you activate exam mode you prevent users of the calculator from accessing those features you have disabled. The features become accessible again at the end of the specified time-out period or on entry of the exam-mode password, whichever occurs sooner.

To activate exam mode:

1. If the **Exam Mode** screen is not showing, press Shift Settings, tap Page ¼, and then tap Page ²⁄₄.



Exam Mode

Configuration: Basic Mode

Default Angle: No change

Memory will be hidden

Choose exam mode configuration

2. If a configuration other than Basic Mode is required, choose it from the **Configuration** list.

3. If you are using a configuration other than Basic Mode, select a time-out period from the **Timeout** list.

   Note that 8 hours is the maximum period. If you are preparing to supervise a student examination, make sure that the time-out period chosen is greater than the duration of the examination.

4. If you are using a configuration other than Basic Mode, either choose a default angle mode or leave the default empty (or, **No change**).

5. If you are using a configuration other than Basic Mode, enter a password of between 1 and 10 characters. The password must be entered if you—or another user—wants to cancel exam mode before the time-out period has elapsed.

6. Select one of the following calculator memory options:

TIP: Basic Mode automatically hides the calculator memory while exam mode is enabled.

- **Keep**—Allows the student to have full access to the current calculator memory, including programs and notes.

- **Erase**—Erases the calculator memory completely.

  NOTE: This action cannot be undone.

- **Hide**—Hides the calculator memory while exam mode is enabled.

- **Keep and restore**—Hides the calculator memory while exam mode is enabled. After the exam mode is disabled, the calculator memory is restored to its state prior to exam mode.

7. If you want the exam mode indicator to blink periodically while the calculator is in exam mode, select **Blink LED**. The green light on top of the calculator automatically blinks in Basic Mode.

8. If you want to increase exam-mode security, select **Security code**, and then give the students the security code to enter to start exam mode.

9. If you are using Basic Mode, tap Start on the student's calculator. Otherwise, use the supplied USB cable to connect a student's calculator.

   Insert the micro-A connector—the one with the rectangular end—into the USB port on the sending calculator, and the other connector into the USB port on the receiving calculator.

10. To activate the configuration on an attached calculator, tap Send, and then select one of the following options:

    - **Send and start**—Automatically starts exam mode on the connected calculator, with the specified disabled features not accessible to the user of that calculator.

    - **Send file**—Starts exam mode on the connected calculator after you disconnect the calculator and tap Start. The calculator is now in exam mode, with the specified disabled features not accessible to the user of that calculator.

11. Repeat from step 9 for each calculator that needs to have its functionality limited.

Cancelling exam mode

If you want to cancel exam mode before the set time period has elapsed, do one of the following:

- Connect the calculator to a computer or another HP Prime calculator using the appropriate cable.
- Enter the password using the following procedure, if you configured a password for the exam-mode configuration.

To enter the exam-mode password:

1. If the **Exam Mode** screen is not showing, press Shift Settings, tap Page 1/4 ▼ and tap ▲ Page 2/4 ▼.
2. Enter the password for the current exam mode activation and tap OK twice.

You can also cancel exam mode using the Connectivity Kit. See the *HP Connectivity Kit User Guide* for more details.

Modifying configurations

Exam mode configurations can be changed. You can also delete a configuration and restore the default configuration.

Changing a configuration

1. If the **Exam Mode** screen is not showing, press Shift Settings, tap Page 1/4 ▼ and tap ▲ Page 2/4 ▼.
2. Select the configuration you want to change from the **Configuration** list.
3. Tap Config.
4. Make whatever changes are necessary and then tap OK.

Returning to the default configuration

1. Press Shift Settings. The **Home Settings** screen appears.
2. Tap Page 1/4 ▼.
3. Tap ▲ Page 2/4 ▼.

   The **Exam Mode** screen appears.
4. Choose **Custom Mode** from the **Configuration** list.
5. Tap More, select **Reset** from the menu and tap OK to confirm your intention to return the configuration to its default settings.

Deleting configurations

1. If the **Exam Mode** screen is not showing, press Shift Settings, tap Page 1/4 ▼ and tap ▲ Page 2/4 ▼.
2. Select the configuration you want to delete from the **Configuration** list.

NOTE: You cannot delete Basic Mode or Custom Mode.

3. Tap More, and then select **Delete**.
4. When asked to confirm the deletion, tap OK or press Enter.

6 Introduction to HP apps

Much of the function of the HP Prime calculator is provided in packages called HP apps. The HP Prime calculator comes with 18 HP apps: 10 dedicated to mathematical topics or tasks, three specialized Solvers, three function Explorers, a spreadsheet, and an app for recording data streamed to the calculator from an external sensing device. You launch an app by first pressing Apps Info (which displays the Application Library screen) and tapping on the icon for the app you want.

What each app enables you to do is outlined in the following table, where the apps are listed in alphabetical order.

App name	Use this app to:
Advanced Graphing	Explore the graphs of symbolic open sentences in x and y. Example: $x^2 + y^2 = 64$
DataStreamer	Collect real-world data from scientific sensors and export it to a statistics app for analysis.
Finance	Solve time-value-of-money (TVM) problems and amortization problems.
Function	Explore real-valued, rectangular functions of y in terms of x. $y = 2x^2 + 3x + 5$
Geometry	Explore geometric constructions and perform geometric calculations.
Inference	Explore confidence intervals and hypothesis tests based on the Normal and Student's t-distributions.
Linear Explorer	Explore the properties of linear equations and test your knowledge.
Linear Solver	Find solutions to sets of two or three linear equations.
Parametric	Explore parametric functions of x and y in terms of t. Example: x = cos(t) and y = sin(t).
Polar	Explore polar functions of r in terms of an angle θ. Example: r = 2cos(4θ)
Quadratic Explorer	Explore the properties of quadratic equations and test your knowledge.
Sequence	Explore sequence functions, where U is defined in terms of n, or in terms of previous terms in the same or another sequence, such as U_{n-1} and U_{n-2}. Example: $U_1 = 0$, $U_2 = 1$, and $U_n = U_{n-2} + U_{n-1}$
Solve	Explore equations in one or more real-valued variables, and systems of equations. Example: $x + 1 = x^2 - x - 2$
Spreadsheet	To solve problems or represent data best suited to a spreadsheet.
Statistics 1Var	Calculate one-variable statistical data (x).
Statistics 2Var	Calculate two-variable statistical data (x and y).
Triangle Solver	Find the unknown values for the lengths and angles of triangles.
Trig Explorer	Explore the properties of sinusoidal equations and test your knowledge.

As you use an app to explore a lesson or solve a problem, you add data and definitions in one or more of the app's views. All this information is automatically saved in the app. You can come back to the app at any time and all the information is still there. You can also save a version of the app with a name you give it and then use the original app for another problem or purpose. See Creating an app on page 105 for more information about customizing and saving apps.

With one exception, all the apps mentioned above are described in detail in this user guide. The exception is the DataStreamer app. A brief introduction to this app is given in the *HP Prime Graphing Calculator Quick Start Guide*. Full details can be found in the *HP StreamSmart 410 User Guide*.

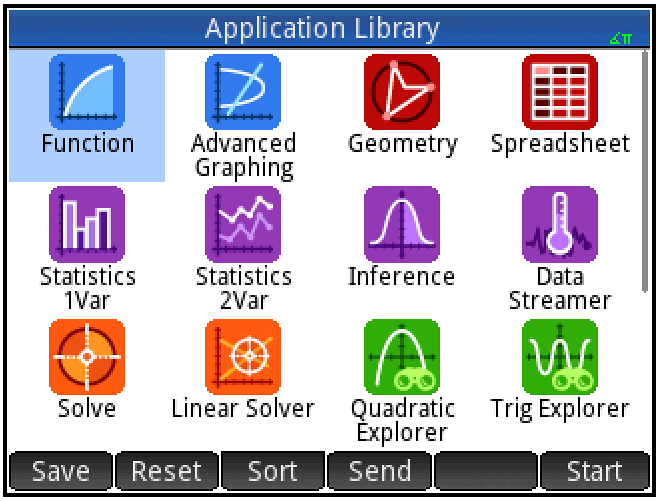
Application library

Apps are stored in the Application Library, displayed by pressing Apps Info .

Opening an app

1. Open the Application Library.
2. Find the desired app icon and tap it.

   You can also use the cursor keys to scroll to the app and, when it is highlighted, either tap Start or press Enter ≈ .



Resetting an app

You can leave an app at any time and all the data and settings in it are retained. When you return to the app, you can continue as you left off.

However, if you don't want to use the previous data and settings, you can return the app to its default state, that is, the state it was in when you opened it for the first time.

To reset the app:

1. Open the Application Library.
2. Use the cursor keys to highlight the app.

3. Tap Reset.

4. Tap OK to confirm your intention.

You can also reset an app from within the app. From the main view of the app—which is usually, but not always, the Symbolic view—press Shift Esc Clear and tap OK to confirm your intention.

Sorting apps

By default, the built-in apps in the Application Library are sorted chronologically, with the most recently used app shown first. (Customized apps always appear after the built-in apps.)

You can change the sort order of the built-in apps to the following:

- **Alphabetically**—The app icons are sorted alphabetically by name, and in ascending order: A to Z.
- **Fixed**—Apps are displayed in their default order: Function, Advanced Graphing, Geometry ... Polar, and Sequence. Customized apps are placed at the end, after all the built-in apps. They appear in chronological order: oldest to most recent.

To change the sort order:

1. Open the Application Library.
2. Tap Sort.
3. From the **Sort Apps** list, choose the option you want.

Deleting an app

The apps that come with the HP Prime calculator are built-in and cannot be deleted, but you can delete an app you have created.

To delete an app:

1. Open the Application Library.
2. Use the cursor keys to highlight the app.
3. Tap Delete.
4. Tap OK to confirm your intention.

Other options

The other options available in the Application Library are as follows:

- Save—Enables you to save a copy of an app under a new name. See Creating an app on page 105.
- Send—Enables you to send an app to another HP Prime calculator.

App views

Most apps have three major views: Symbolic, Plot, and Numeric. These views are based on the symbolic, graphic, and numeric representations of mathematical objects. They are accessed through the Symb Setup , Plot Setup, and Num Setup keys near the top left of the keyboard. Typically these views enable you to define a mathematical object—such as an expression or an open sentence—plot it, and see the values generated by it.

Each of these views has an accompanying setup view, a view that enables you to configure the appearance of the data in the accompanying major view. These views are called Symbolic Setup, Plot Setup, and Numeric Setup. They are accessed by pressing Shift Symb Setup, Shift Plot Setup, and Shift Num Setup.

Not all apps have all the six views outlined above. The scope and complexity of each app determines its particular set of views. For example, the Spreadsheet app has no Plot view or Plot Setup view, and the Quadratic Explorer has only a Plot view. What views are available in each app is outlined in the next six sections.

Note that the DataStreamer app is not covered in this chapter. See *HP StreamSmart 410 User Guide* for information about this app.

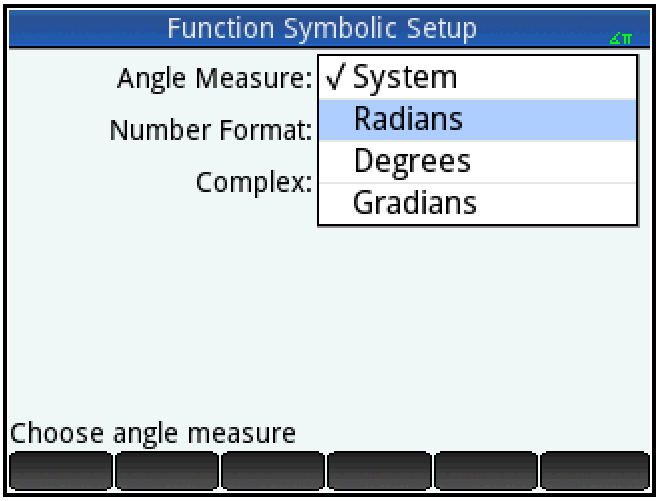
Symbolic view

The following table outlines what is done in the Symbolic view of each app.

App	Use the Symbolic view to do the following:
Advanced Graphing	Specify up to 10 open sentences.
Finance	N/A
Function	Specify up to 10 real-valued, rectangular functions of y in terms of x.
Geometry	View the symbolic definition of geometric constructions.
Inference	Choose to conduct a hypothesis test or test a confidence level, and select a type of test.
Linear Explorer	N/A
Linear Solver	N/A
Parametric	Specify up to 10 parametric functions of x and y in terms of t.
Polar	Specify up to 10 polar functions of r in terms of an angle θ.
Quadratics Explorer	N/A
Sequence	Specify up to 10 sequence functions.
Solve	Specify up to 10 equations.
Spreadsheet	N/A
Statistics 1Var	Specify up to 5 univariate analyses.
Statistics 2Var	Specify up to 5 multivariate analyses.
Triangle Solver	N/A
Trig Explorer	N/A

Symbolic Setup view

The Symbolic Setup view is the same for each app. It enables you to override the system-wide settings for angle measure, number format, and complex-number entry. The override applies only to the current app.



Function Symbolic Setup

Angle Measure:

Number Format:

Complex:

Choose angle measure

You can change the settings for all apps using the Home settings and CAS settings.

Plot view

The following table outlines what is done in the Plot view of each app.

App	Use the Plot view to do the following:
Advanced Graphing	Plot and explore the open sentences selected in Symbolic view.
Finance	Display an amortization graph.
Function	Plot and explore the functions selected in Symbolic view.
Geometry	Create and manipulate geometric constructions.
Inference	View a plot of the test results.
Linear Explorer	Explore linear equations and test your knowledge of them.
Linear Solver	N/A
Parametric	Plot and explore the functions selected in Symbolic view.
Polar	Plot and explore the functions selected in Symbolic view.
Quadratics Explorer	Explore quadratic equations and test your knowledge of them.
Sequence	Plot and explore the sequences selected in Symbolic view.
Solve	Plot and explore a single function selected in Symbolic view.
Spreadsheet	N/A
Statistics 1Var	Plot and explore the analyses selected in Symbolic view.
Statistics 2Var	Plot and explore the analyses selected in Symbolic view.
Triangle Solver	N/A
Trig Explorer	Explore sinusoidal equations and test your knowledge of them.

Plot Setup view

The following table outlines what is done in the Plot Setup view of each app.

App	Use the Plot Setup view to do the following:
Advanced Graphing	Modify the appearance of plots and the plot environment.
Finance	N/A
Function	Modify the appearance of plots and the plot environment.
Geometry	Modify the appearance of the drawing environment.
Inference	N/A
Linear Explorer	N/A
Linear Solver	N/A
Parametric	Modify the appearance of plots and the plot environment.
Polar	Modify the appearance of plots and the plot environment.
Quadratics Explorer	N/A
Sequence	Modify the appearance of plots and the plot environment.
Solve	Modify the appearance of plots and the plot environment.
Spreadsheet	N/A
Statistics 1Var	Modify the appearance of plots and the plot environment.
Statistics 2Var	Modify the appearance of plots and the plot environment.
Triangle Solver	N/A
Trig Explorer	N/A

Numeric view

The following table outlines what is done in the Numeric view of each app.

App	Use the Numeric view to do the following:
Advanced Graphing	View a table of numbers generated by the open sentences selected in Symbolic view.
Finance	Enter values for time-value-of-money calculations.
Function	View a table of numbers generated by the functions selected in Symbolic view.
Geometry	Perform calculations on the geometric objects drawn in Plot view.
Inference	Specify the statistics needed to perform the test selected in Symbolic view.
Linear Explorer	N/A
Linear Solver	Specify the coefficients of the linear equations to be solved.
Parametric	View a table of numbers generated by the functions selected in Symbolic view.
Polar	View a table of numbers generated by the functions selected in Symbolic view.
Quadratics Explorer	N/A

App	Use the Numeric view to do the following:
Sequence	View a table of numbers generated by the sequences selected in Symbolic view.
Solve	Enter the known values and solve for the unknown value.
Spreadsheet	Enter numbers, text, formulas, etc. The Numeric view is the primary view for this app.
Statistics 1Var	Enter data for analysis.
Statistics 2Var	Enter data for analysis.
Triangle Solver	Enter known data about a triangle and solve for the unknown data.
Trig Explorer	N/A

Numeric Setup view

The following table outlines what is done in the Numeric Setup view of each app.

App	Use the Numeric Setup view to do the following:
Advanced Graphing	Specify the numbers to be calculated according to the open sentences specified in Symbolic view, and set the zoom factor.
Finance	N/A
Function	Specify the numbers to be calculated according to the functions specified in Symbolic view, and set the zoom factor.
Geometry	N/A
Inference	N/A
Linear Explorer	N/A
Linear Solver	N/A
Parametric	Specify the numbers to be calculated according to the functions specified in Symbolic view, and set the zoom factor.
Polar	Specify the numbers to be calculated according to the functions specified in Symbolic view, and set the zoom factor.
Quadratics Explorer	N/A
Sequence	Specify the numbers to be calculated according to the functions specified in Symbolic view, and set the zoom factor.
Solve	N/A
Spreadsheet	N/A
Statistics 1Var	N/A
Statistics 2Var	N/A
Triangle Solver	N/A
Trig Explorer	N/A

Quick example

The following example uses all six app views and should give you an idea of the typical workflow involved in working with an app. The Polar app is used as the sample app.

Opening the app

1. Press Apps Info to open the Application Library.

2. Tap the Polar app icon.

The Polar app opens in Symbolic view.

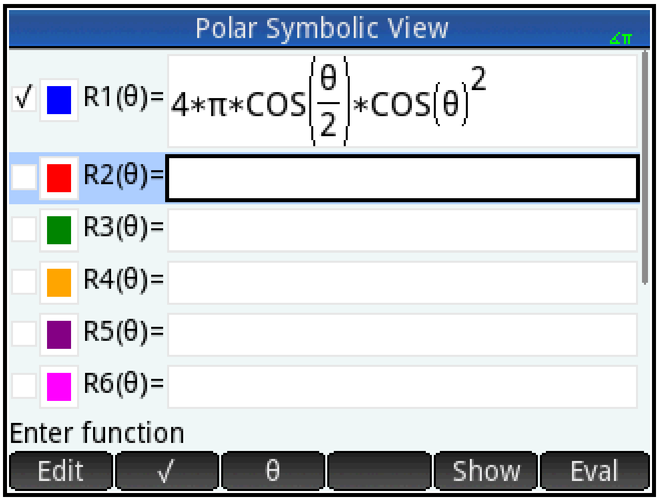
Symbolic view

The Symbolic view of the Polar app is where you define or specify the polar equation you want to plot and explore. In this example we will plot and explore the equation r = 5πcos(θ/2)cos(θ)^2.

▲ Define the equation r = 5πcos(θ/2)cos(θ)^2 as follows:

5 Shift 3 COS θ ÷ 2 ▶ ▶ COS θ ▶ x^2 Enter

(If you are using algebraic entry mode, enter 5 Shift 3 COS θ ÷ 2 ▶ COS θ ▶ x^2 Enter.)



Polar Symbolic View

☑ 🟦 R1(θ)= $4*\pi*COS\left(\frac{\theta}{2}\right)*COS(\theta)^2$

□ 🟥 R2(θ)=

□ 🟩 R3(θ)=

□ 🟧 R4(θ)=

□ 🟪 R5(θ)=

□ 🟪 R6(θ)=

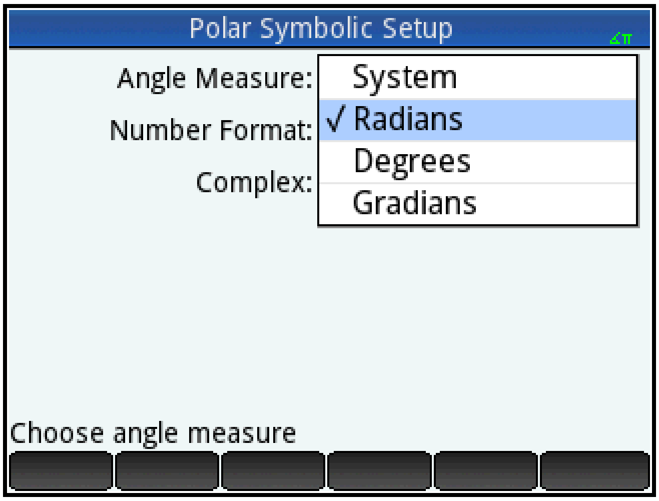
Enter function

This equation draws symmetrical petals provided that the angle measure is set to radians. The angle measure for this app is set in the Symbolic Setup view.

Symbolic Setup view

1. Press Shift Symb (Setup).

2. Select **Radians** from the Angle Measure menu.



Polar Symbolic Setup

Angle Measure: System

Number Format: √ Radians

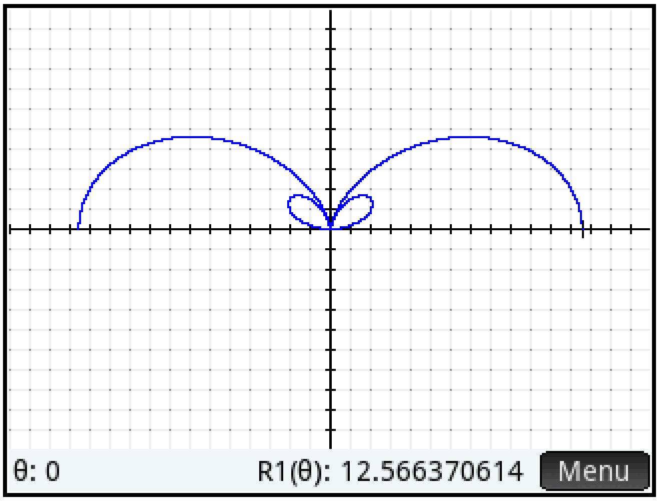
Complex: Degrees

Gradians

Choose angle measure

Plot view

▲ Press Plot (Setup).



θ: 0 R1(θ): 12.566370614

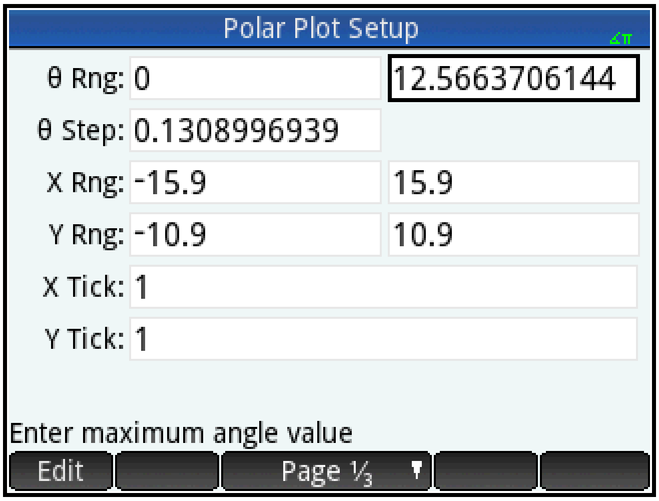
A graph of the equation is plotted. However, as the previous figure shows, only a part of the petals is visible. To see the rest you will need to change the plot setup parameters.

Plot Setup view

1. Press Shift Plot ↳Setup .

2. Set the second **θ Rng** field to 4π by entering:

   ▶ 4 Shift 3 (π) OK

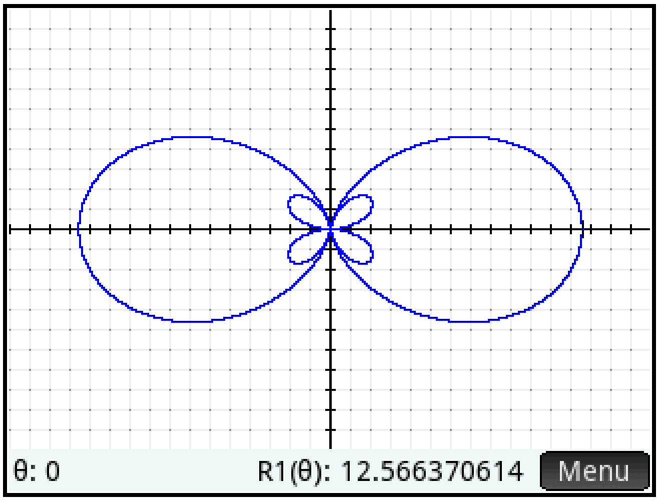


Polar Plot Setup

θ Rng:	0	12.5663706144
θ Step:	0.1308996939	
X Rng:	-15.9	15.9
Y Rng:	-10.9	10.9
X Tick:	1	
Y Tick:	1	

Enter maximum angle value

3. Press Plot ↳Setup to return to Plot view and see the complete plot.



θ: 0 R1(θ): 12.566370614

Numeric view

The values generated by the equation can be seen in Numeric view.

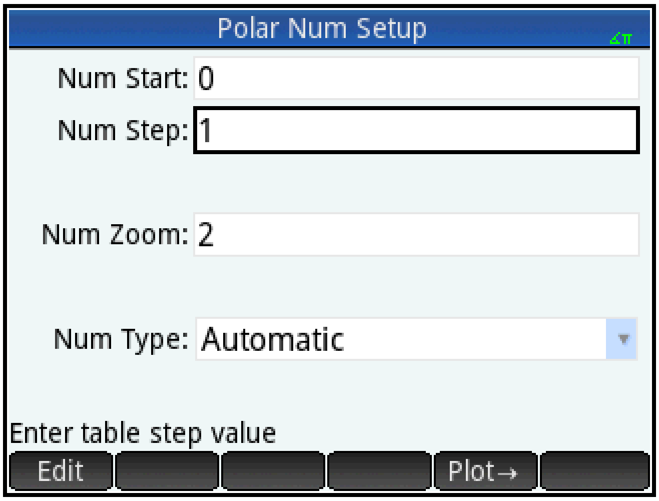
▲ Press Num ↳Setup .

Suppose you want to see just whole numbers for θ; in other words, you want the increment between consecutive values in the θ column to be 1. You set this up in the Numeric Setup view.

Numeric Setup view

1. Press Shift Num ↳Setup.

2. Change the **Num Step** field to 1.



Polar Num Setup

Num Start: 0

Num Step: 1

Num Zoom: 2

Num Type: Automatic

Enter table step value

3. Press Num ↳Setup to return to Numeric view.

You will see that the θ column now contains consecutive integers starting from zero, and the corresponding values calculated by the equation specified in Symbolic view are listed in the R1 column.

Common operations in Symbolic view

This section discusses Advanced Graphing, Function, Parametric, Polar, Sequence, Solve. See the dedicated app chapters for information about the other apps.

Symbolic view is typically used to define a function or open sentence that you want to explore (by plotting and/or evaluating). In this section, the term “definition” will be used to cover both functions and open sentences.

Press Symb ↳Setup to open Symbolic view.

Adding a definition

With the exception of the Parametric app, there are 10 fields for entering definitions. In the Parametric app there are 20 fields, two for each paired definition.

1. Highlight an empty field you want to use, either by tapping on it or scrolling to it.

2. Enter your definition.

   NOTE: The variables used in definitions must be in uppercase. A variable entered in lowercase causes an error message to appear.

   If you need help, see Definitional building blocks on page 73.

3. Tap OK or press Enter ≈ when you have finished.

Your new definition is added to the list of definitions.

Modifying a definition

1. Highlight the definition you want to modify, either by tapping on it or scrolling to it.
2. Tap Edit.

   The definition is copied to the entry line.
3. Modify the definition.
4. Tap OK or press Enter when you have finished.

Definitional building blocks

The components that make up a symbolic definition can come from a number of sources.

- From the keyboard

  You can enter components directly from the keyboard. To enter 2X^2 – 3, just press 2 ALPHA X x^2 − 3.

- From user variables

  If, for example, you have created a variable called COST, you could incorporate that into a definition either by typing it or choosing it from the **User** menu (one of the sub-menus of the Variables menu). Thus you could have a definition that reads F1(X) = X^2 + COST.

  To select a user variable, press Vars , tap User, select **User Variables**, and then select the variable of interest.

- From Home variables

  Some Home variables can be incorporated into a symbolic definition. To access a Home variable, press Vars , tap Home, select a category of variable, and select the variable of interest. Thus you could have a definition that reads F1(X) = X^2 + Q. (Q is on the **Real** sub-menu of the **Home** menu.)

- From app variables

  All settings, definitions, and results, for all apps, are stored as variables. Many of these variables can be incorporated into a symbolic definition. To access app variables, press Vars , tap App, select the app, select the category of variable, and then select the variable of interest. You could, for instance, have a definition that reads F2(X) = X^2 + X – Root. The value of the last root calculated in the Function app is substituted for Root when this definition is evaluated.

- From math functions

  Some of the functions on the **Math** menu can be incorporated into a definition. The **Math** menu is one of the Toolbox menus (🧰). The following definition combines a math function (**Size**) with a Home variable (L1): F4(X) = X^2 – SIZE(L1). It is equivalent to x^2 – n where n is the number of elements in the list named L1. (**Size** is an option on the **List** menu, which is a sub-menu of the **Math** menu.)

- From CAS functions

  Some of the functions on the **CAS** menu can be incorporated into a definition. The **CAS** menu is one of the Toolbox menus (Mem). The following definition incorporates the CAS function irem: F5(X) = X^2+ CAS.irem(45,7). (irem is entered by selecting **Remainder**, an option on the **Division** menu, which is a sub-menu of the **Integer** menu. Note that any CAS command or function selected to operate outside the CAS is given the CAS. prefix.)

- From app functions

  Some of the functions on the **App** menu can be incorporated into a definition. The **App** menu is one of the Toolbox menus (Mem). The following definition incorporates the app function PredY:

  F9(X) = X^2 + Statistics_2Var.PredY(6).

- From the **Catlg** menu

  Some of the functions on the **Catlg** menu can be incorporated into a definition. The **Catlg** menu is one of the Toolbox menus (Mem). The following definition incorporates a command from that menu and an app variable: F6(X) = X^2 + INT(Root). The integer value of the last root calculated in the Function app is substituted for INT(Root) when this definition is evaluated.

- From other definitions

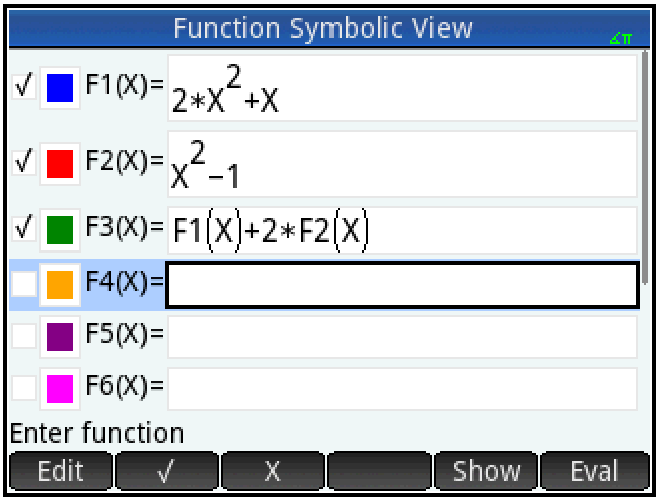
  For example, you can define F3(X) as F1(X) * F2(X).

Evaluating a dependent definition

If you have a dependent definition—that is, one defined in terms of another definition—you can combine all the definitions into one by evaluating the dependent definition.

1. Select the dependent expression.
2. Tap Eval.

Consider the following example. Notice that F3(X) is defined in terms of two other functions. It is a dependent definition and can be evaluated. If you highlight F3(X) and tap Eval, F3(X) becomes 2 * X^2 + X+ 2 * (X^2 – 1).


Function Symbolic View

✓ 🟦 F1(X)= $2*X^2+X$

✓ 🟥 F2(X)= X^2-1

✓ 🟩 F3(X)= F1(X)+2*F2(X)

□ 🟧 F4(X)=

□ 🟪 F5(X)=

□ 🟪 F6(X)=

Enter function

Selecting or deselecting a definition to explore

In the Advanced Graphing, Function, Parametric, Polar, Sequence, and Solve apps you can enter up to 10 definitions. However, only those definitions that are selected in Symbolic view will be plotted in Plot view and evaluated in Numeric view.

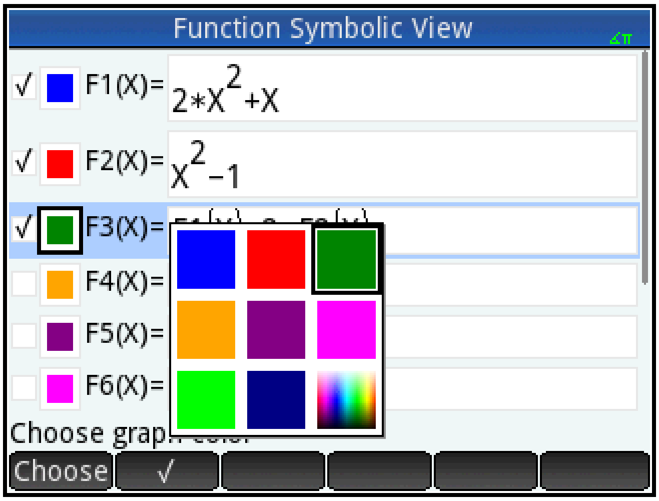
You can tell if a definition is selected by the tick (or checkmark) beside it. A checkmark is added by default as soon as you create a definition. So if you don't want to plot or evaluate a particular definition, highlight it and tap √. (Do likewise if you want to reselect a deselected function.)

Choosing a color for plots

Each function and open sentence can be plotted in a different color. If you want to change the default color of a plot:

1. Tap the colored square to the left of the function's definition.

   You can also select the square by pressing Enter while the definition is selected. Pressing Enter moves the selection from the definition to the colored square and from the colored square to the definition.



Function Symbolic View

√ 🟦 F1(X)= $2*X^2+X$

√ 🟥 F2(X)= X^2-1

√ 🟩 F3(X)=

🟧 F4(X)=

🟪 F5(X)=

🟪 F6(X)=

Choose grap

2. Tap Choose.
3. Select the desired color from the color-picker.

Deleting a definition

To delete a single definition:

1. Tap the definition once (or highlight it using the cursor keys).
2. Press Del.

To delete all the definitions:

1. Press Shift Esc Clear .

2. Tap OK or press Enter ≈ to confirm your intention.

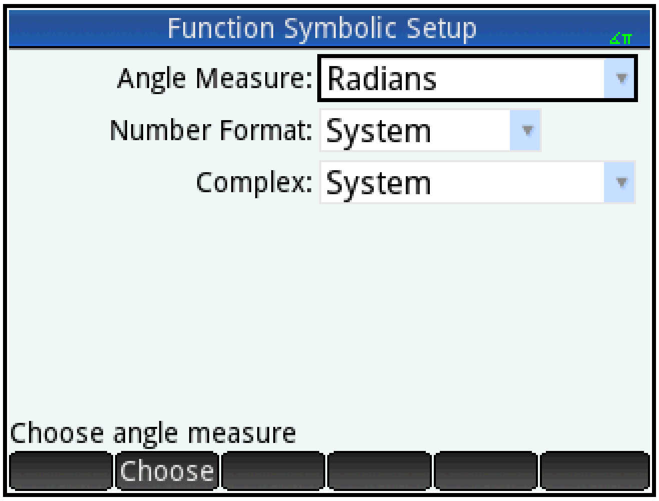
Symbolic view: Summary of menu buttons

Button	Purpose
Edit	Copies the highlighted definition to the entry line for editing. Tap OK when done. To add a new definition—even one that is replacing an existing one—highlight the field and just start entering your new definition.
√	Selects (or deselects) a definition.
X [Function only]	Enters the independent variable in the Function app. You can also press xtθn Define D .
X [Advanced Graphing only]	Enters an X in the Advanced Graphing app. You can also press xtθn Define D .
Y [Advanced Graphing only]	Enters an Y in the Advanced Graphing app.
T [Parametric only]	Enters the independent variable in the Parametric app. You can also press xtθn Define D .
θ [Polar only]	Enters the independent variable in the Polar app. You can also press xtθn Define D .
N [Sequence only]	Enters the independent variable in the Sequence app. You can also press xtθn Define D .
= [Solve only]	Enters the equal sign in the Solve app. A shortcut equivalent to pressing Shift ≐ .
Show	Displays the selected definition in full-screen mode.
Eval	Evaluates dependent definitions. See Evaluating a dependent definition on page 74.

Common operations in Symbolic Setup view

The Symbolic Setup view is the same for all apps. Its primary purpose is to allow you to override three of the system-wide settings specified on the **Home Settings** window.

Press Shift Symb ↳Setup to open Symbolic Setup view.



Function Symbolic Setup

Angle Measure: Radians

Number Format: System

Complex: System

Choose angle measure

Overriding system-wide settings

1. Tap the setting you want to change.

   You can tap the field name or the field.

2. Tap the setting again.

   A menu of options appears.

3. Select the new setting.

NOTE: Selecting the **Fixed**, **Scientific**, or **Engineering** options on the **Number Format** menu displays a second field for you to enter the required number of significant digits.

You can also select a field, tap Choose, and select the new setting.

Restoring default settings

To restore default settings is to return precedence to the settings on the **Home Settings** screen.

To restore one field to its default setting:

1. Select the field.
2. Press Del.

To restore all default settings, press Shift Esc Clear.

Common operations in Plot view

Plot view functionality that is common to many apps is described in detail in this section. Functionality that is available only in a particular app is described in the chapter dedicated to that app.

Press Plot Setup to open Plot view.

Zoom

To easily zoom in Plot view, use a 2-finger pinch zoom. If a 2-finger pinch zoom gesture is performed horizontally, the zoom is performed on the x-axis only. If a 2-finger pinch zoom is performed vertically, the zoom is performed on the y-axis only. If a 2-finger pinch zoom is performed diagonally, a square zoom is performed (that is, the zoom is performed on both axes).

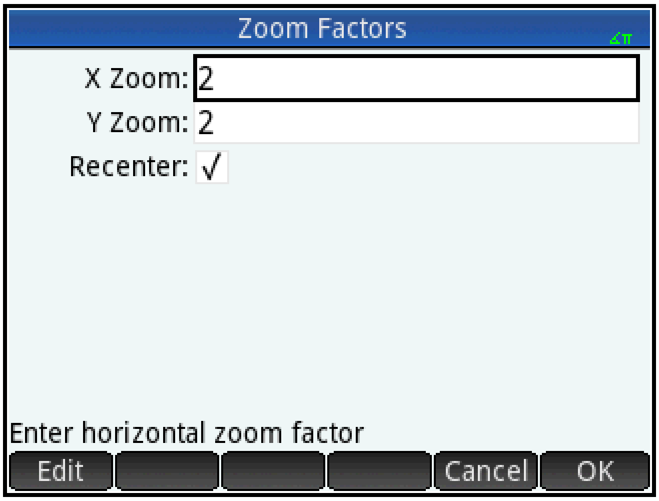
For more concise control, use the options in the Zoom menu. These options use either a horizontal or vertical factor, or both. By default, these factors are both set to the number 2. Zooming out multiplies the scale by the factor, so that a greater scale distance appears on the screen. Zooming in divides the scale by the factor, so that a shorter scale distance appears on the screen.

Zoom factors

To change the default zoom factors:

1. Open the Plot view of the app (Plot Setup).
2. Tap Menu to open the Plot view menu.
3. Tap Zoom to open the Zoom menu.
4. Scroll and select **Set Factors**.

   The **Zoom Factors** screen appears.



Zoom Factors

X Zoom: 2

Y Zoom: 2

Recenter: ✓

Enter horizontal zoom factor

5. Change one or both zoom factors.

6. If you want the plot to be centered around the current position of the cursor in Plot view, select Recenter.

7. Tap OK or press Enter.

Zoom options

Zoom options are available from the following sources:

- Touchscreen
- Keyboard
- Zoom menu in Plot view
- **View** menu (View Copy)

Zoom gestures

In Plot view, a 2-finger pinch zoom performed diagonally zooms by the same scale factor in both vertical and horizontal directions. A 2-finger pinch zoom performed vertically zooms on the y-axis only. A 2-finger pinch zoom performed horizontally zooms on the x-axis only.

In Numeric view, a 2-finger pinch zoom performed vertically zooms the selected row. A zoom in decreases the common difference in the x-values, and a zoom out increases the common difference in the x-values.

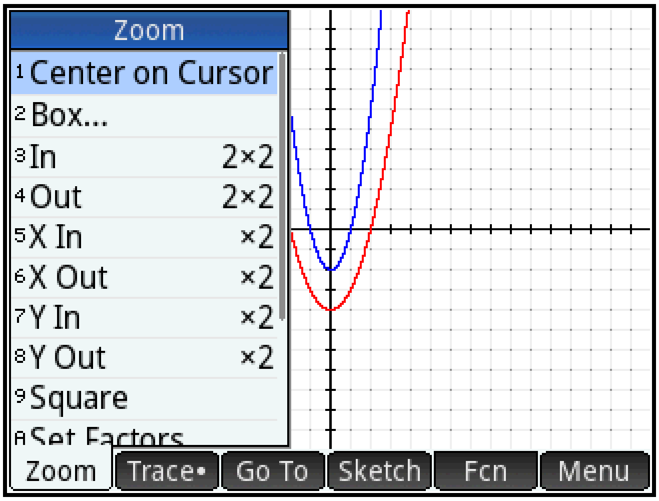
Zoom keys

There are two zoom keys: pressing + zooms in and pressing − zooms out. The extent of the scaling is determined by the **Zoom Factors** settings.

Zoom menu

In Plot view, tap Zoom and tap an option. (If Zoom is not displayed, tap Menu)

The zoom options are explained in the following table. Examples are provided on Zoom examples on page 82.



Option	Result
Center on Cursor	Redraws the plot so that the cursor is in the center of the screen. No scaling occurs.
Box	See Box zoom on page 80.
In	Divides the horizontal and vertical scales by **X Zoom** and **Y Zoom** (values set with the **Set Factors** option). For instance, if both zoom factors are 4, then zooming in results in 1/4 as many units depicted per pixel. (Shortcut: press + .)
Out	Multiplies the horizontal and vertical scales by the **X Zoom** and **Y Zoom** settings. (Shortcut: press − .)
X In	Divides the horizontal scale only, using the **X Zoom** setting.
X Out	Multiplies the horizontal scale only, using the **X Zoom** setting.
Y In	Divides the vertical scale only, using the **Y Zoom** setting.
Y Out	Multiplies the vertical scale only, using the **Y Zoom** setting.
Square	Changes the vertical scale to match the horizontal scale. This is useful after you have done a box zoom, X zoom or Y zoom.
Autoscale	Rescales the vertical axis so that the display shows a representative piece of the plot given the supplied x axis settings. (For Sequence, Polar, parametric, and Statistics apps, autoscaling rescales both axes.) The autoscale process uses the first selected function to determine the best scale to use.
Decimal	Rescales both axes so each pixel is 0.1 units. This is equivalent to resetting the default values for **xrng** and **yrng**.
Integer	Rescales the horizontal axis only, making each pixel equal to 1 unit.
Trig	Rescales the horizontal axis so that 1 pixel equals π/24 radians or 7.5 degrees; rescales the vertical axis so that 1 pixel equals 0.1 units.
Undo Zoom	Returns the display to the previous zoom. **NOTE:** This option is only available after a zoom operation has been performed.

Box zoom

A box zoom enables you to zoom in on an area of the screen that you specify.

1. With the Plot view menu open, tap Zoom and select **Box**.

2. Tap one corner of the area you want to zoom in on and then tap OK.

3. Tap the diagonally opposite corner of the area you want to zoom in on and then tap OK.

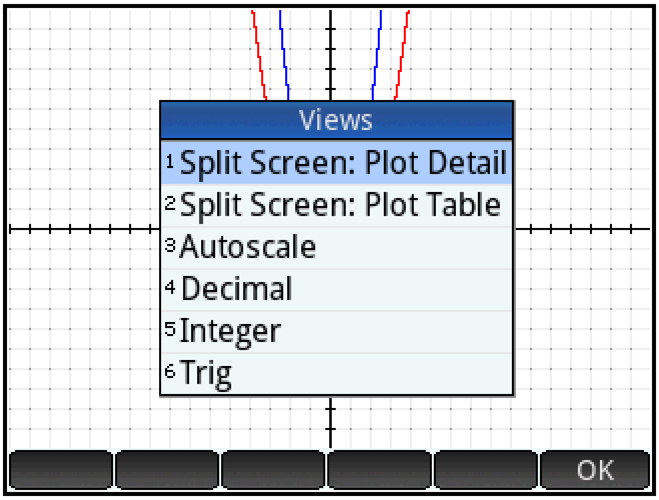
   The screen fills with the area you specified. To return to the default view, tap Zoom and select **Decimal**.

You can also use the cursor keys to specify the area you want to zoom in on.

Views menu

The most commonly used zoom options are also available on the Views menu. These are as follows:

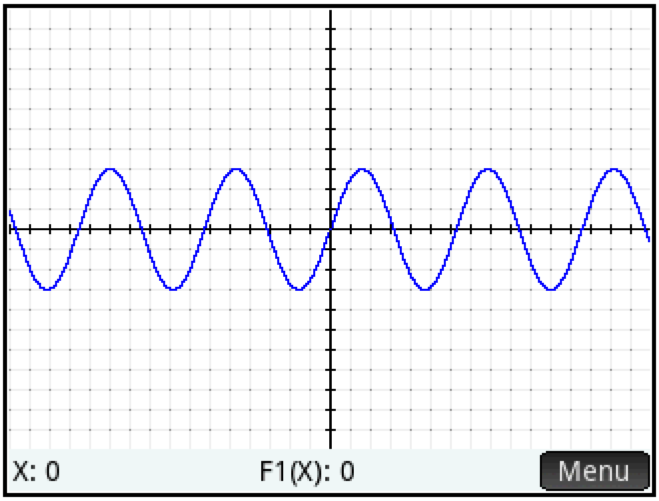
- Autoscale
- Decimal
- Integer
- Trig



These options can be applied whatever view you are currently working in.

Testing a zoom with split-screen viewing

A useful way of testing a zoom is to divide the screen into two halves, with each half showing the plot, and then to apply a zoom only to one side of the screen. The following figure is a plot of y = 3sinx.



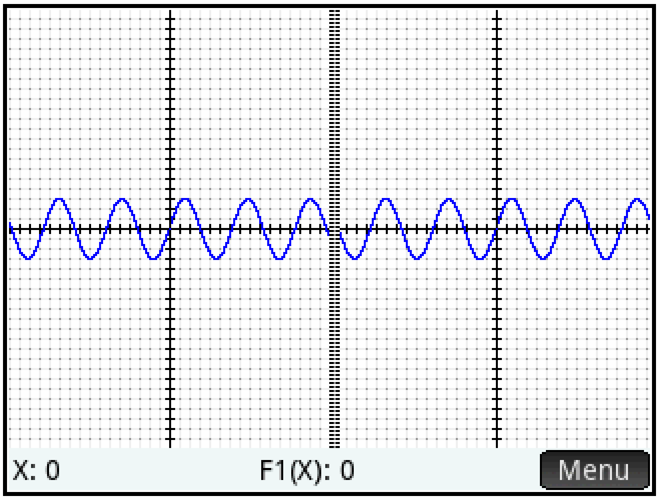
To split the screen into two halves:

1. Open the Views menu.

   Press View Copy .

2. Select **Split Screen: Plot Detail**.

The result is shown in the following figure. Any zoom operation you undertake will be applied only to the copy of the plot in the right-hand half of the screen. This will help you test and then choose an appropriate zoom.


X: 0 F1(X): 0

NOTE: You can replace the original plot on the left with the zoomed plot on the right by tapping ←Plot.

To unsplit the screen, press Plot ↳Setup.

Zoom examples

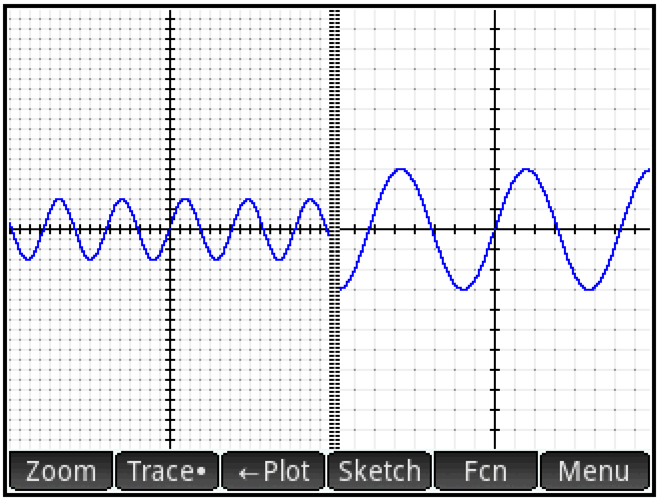
The following examples show the effects of the zooming options on a plot of 3sinx using the default zoom factors (2 × 2). Split-screen mode (described previously) has been used to help you see the effect of zooming.

NOTE: There is an **Unzoom** option on the **Zoom** menu. Use this to return a plot to its pre-zoom state. If the **Zoom** menu is not shown, tap Menu.

Zoom In

Zoom In

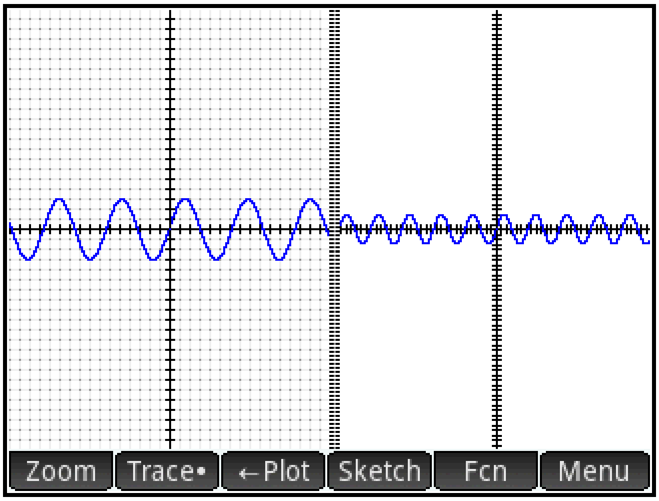
Shortcut: press +



Zoom Out

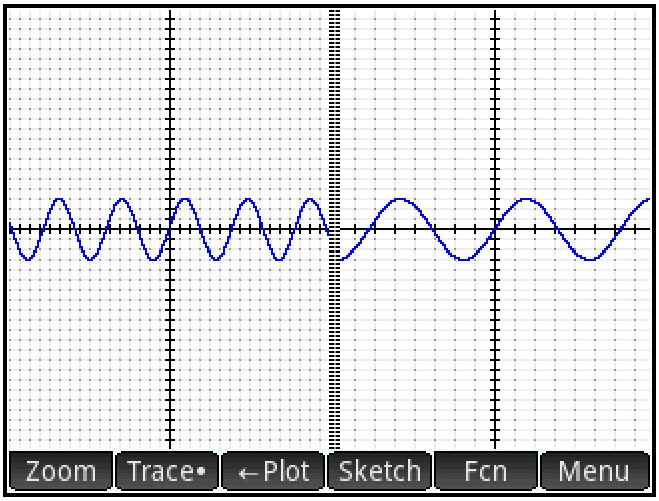
Menu Zoom Out

Shortcut: press −



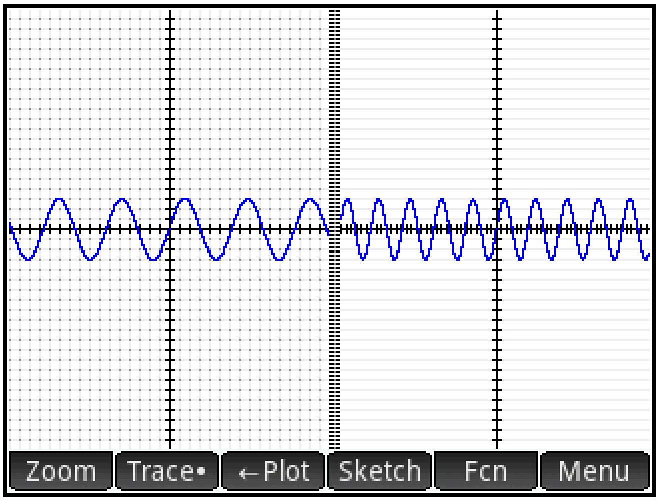
X In

Zoom X In



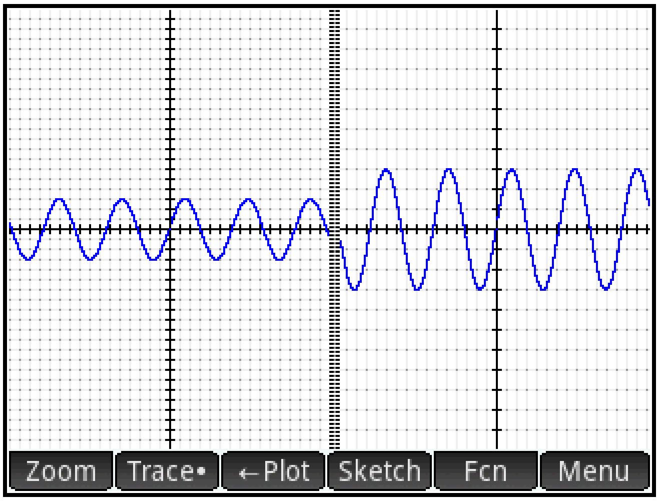
X Out

Zoom X Out



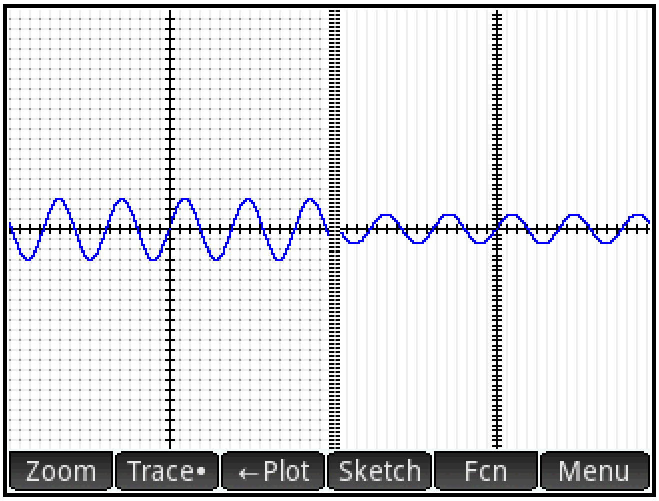
Y In

Zoom **Y In**



Y Out

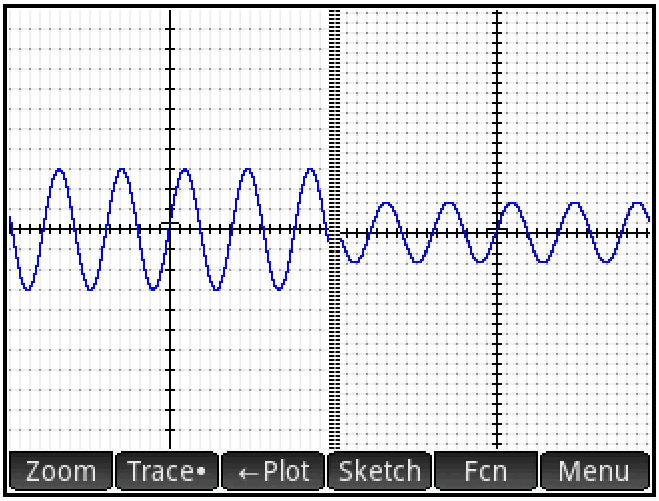
Zoom **Y Out**



Square

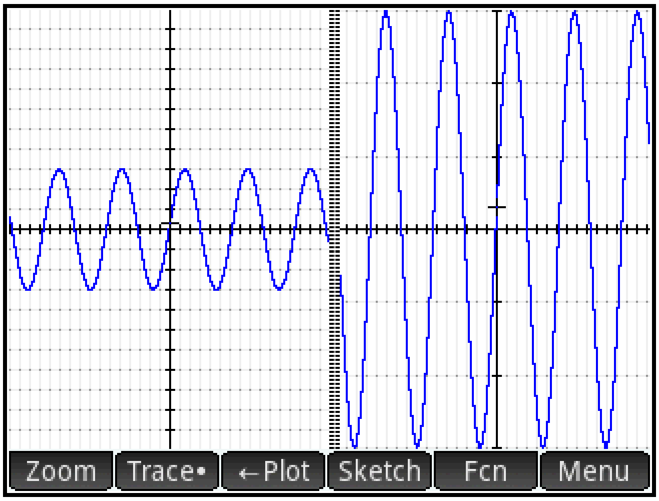
Zoom **Square**

NOTE: In this example, the plot on left has had a **Y In** zoom applied to it. The **Square** zoom has returned the plot to its default state where the X and Y scales are equal.



Autoscale

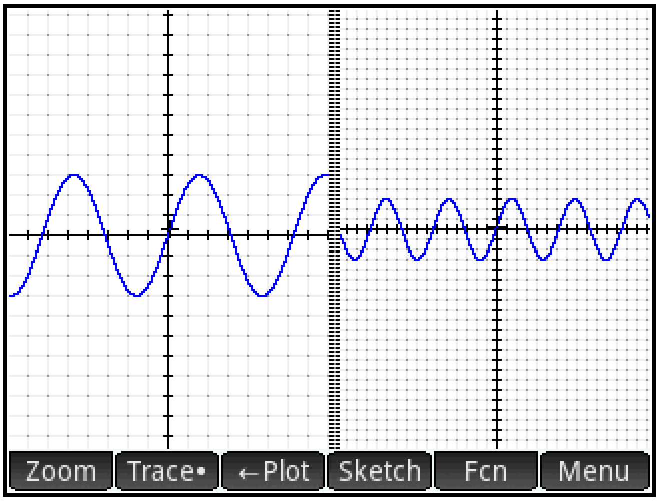
Zoom **Autoscale**



Decimal

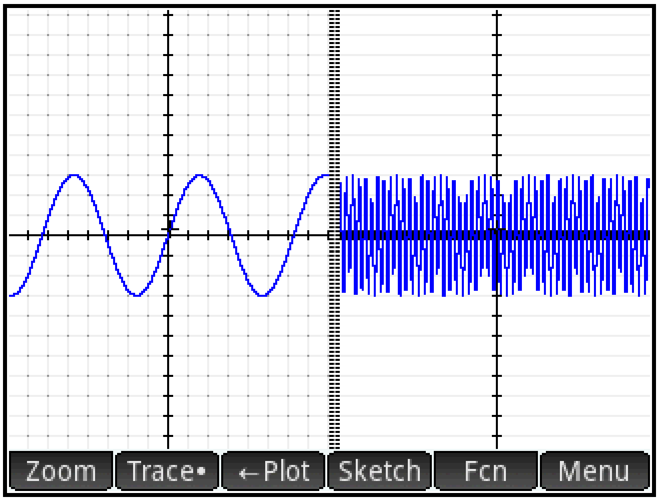
Zoom **Decimal**

NOTE: In this example, the plot on left has had a **X In** zoom applied to it. The **Decimal** zoom has returned the plot to its default state where the X and Y scales are equal.



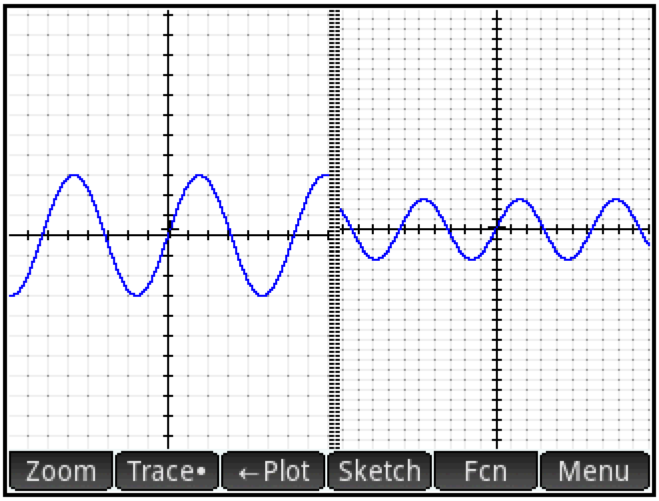
Integer

Zoom **Integer**



Trig

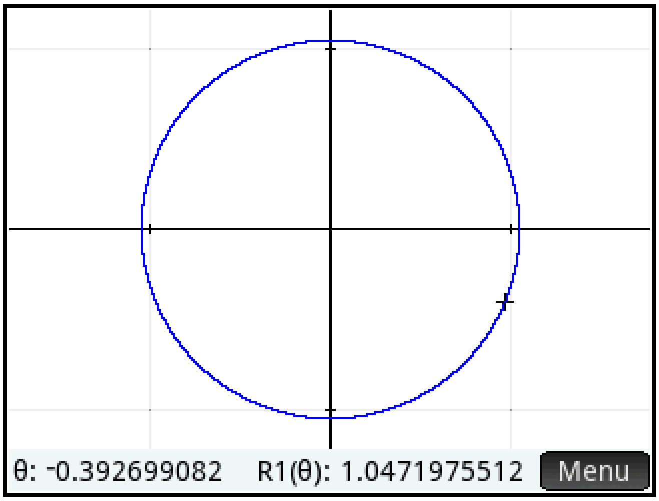
Zoom **Trig**



Trace

This section applies to the Advanced Graphing, Function, Parametric, Polar, Sequence, Solve, Statistics 1 Var, and Statistics 2Var apps.

The tracing function enables you to move a cursor (the trace cursor) along the current graph. You move the trace cursor by pressing ◀ or ▶. You can also move the trace cursor by tapping on or near the current plot. The trace cursor jumps to the point on the graph that is closest to where you tapped.



θ: -0.392699082 R1(θ): 1.0471975512

The current coordinates of the cursor are shown at the bottom of the screen. (If menu buttons are hiding the coordinates, tap Menu to hide the buttons.)

Trace mode and coordinate display are automatically turned on when a plot is drawn.

Selecting a plot

Except in the Advanced Graphing app, if there is more than one plot displayed, press ▲ or ▼ until the trace cursor is on the plot you are interested in.

In the Advanced Graphing app, tap-and-hold on the plot you are interested in. Either the plot is selected, or a menu of plots appears for you select one.

Evaluating a function

One of the primary uses of the trace functionality is to evaluate a plotted definition. Suppose in Symbolic view you have defined F1(X) as (X – 1)^2 – 3. Suppose further that you want to know what the value of that function is when X is 25.

1. Open Plot view (Plot ↳Setup).

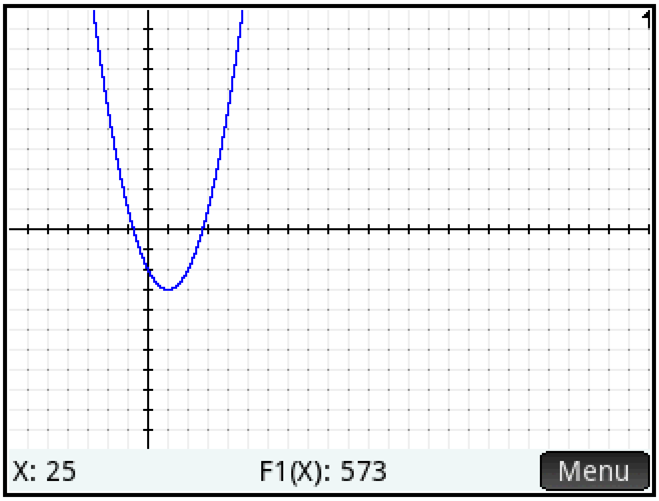
2. If the menu at the bottom of the screen is not open, tap Menu .

3. Tap Go To .

4. Enter 25 and tap OK .

5. Tap Menu .

   The value of F1(X) when X is 25 as shown at the bottom of the screen.



X: 25 F1(X): 573 Menu

This is one of many ways the HP Prime calculator provides for you to evaluate a function for a specific independent variable. You can also evaluate a function in Numeric view (see Common operations in Numeric view on page 97). Moreover, any expression you define in Symbolic view can be evaluated in Home view. For example, suppose F1(X) is defined as (X – 1)^2 – 3. If you enter F1(4) in Home view and press Enter ≈ you get 6, since (4 – 1)^2 – 3 = 6.

Turning tracing on or off

- To turn on tracing, tap Trace• .
- To turn off tracing, tap Trace .

If these options are not displayed, tap Menu .

When tracing is off, pressing the cursor keys no longer constrains the cursor to a plot.

Plot view: Summary of menu buttons

Button	Purpose
Zoom	Displays a menu of zoom options. See Zoom options on page 79.
Trace• / Trace	A toggle button for turning off and turning on trace functionality. See Trace on page 89.
Go To	Displays an input form for you to specify a value you want the cursor to jump to. The value you enter is the value of the independent variable.
Fcn [Function and Statistics 2Var only]	Displays a menu of options for analyzing a plot.
Defn	Displays the symbolic definition of the current function. In the Function and Statistics 2Var apps, this entry is found under the Fcn menu.
Menu	A toggle button that shows and hides the other buttons across the bottom of the screen.

Copy-and-paste operations in Plot view

In many apps, pressing Shift View/Copy in Plot view displays a list of options for copying. You can copy the current display to any graphic variable (G1–G9) or copy the selected x-value or y-value to the clipboard.

Common operations in Plot Setup view

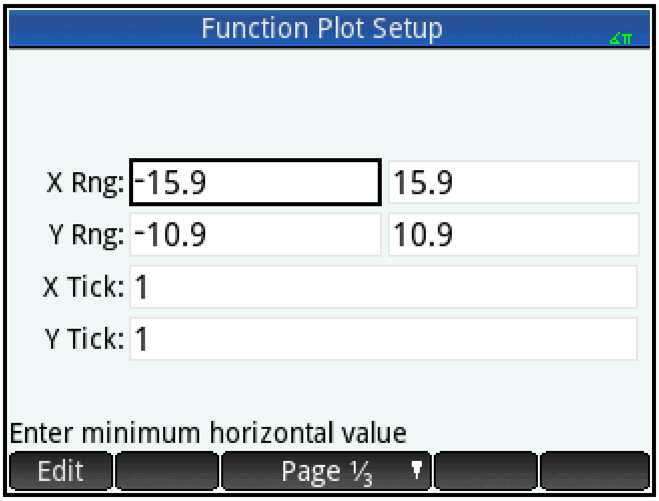
This section covers only operations common to the apps mentioned. See the chapter dedicated to an app for the app-specific operations done in Plot Setup view.

Press Shift Plot/Setup to open Plot Setup view.

Configuring Plot view

This section applies to the Advanced Graphing, Function, Parametric, Polar, Sequence, Statistics 1 Var, and Statistics 2Var apps.

The Plot Setup view is used to configure the appearance of Plot view and to set the method by which graphs are plotted. The configuration options are spread across three pages. Swipe up or down to move between pages, or use the menu keys.



Function Plot Setup

X Rng: -15.9 15.9

Y Rng: -10.9 10.9

X Tick: 1

Y Tick: 1

Enter minimum horizontal value

TIP: When you go to Plot view to see the graph of a definition selected in Symbolic view, there may be no graph shown. The likely cause of this is that the spread of plotted values is outside the range settings in Plot Setup view. A quick way to bring the graph into view is to press View Copy and select **Autoscale**. This also changes the range settings in Plot Setup view.

Page 1

Setup field	Purpose
T RNG [Parametric only]	Sets the range of T-values to be plotted. Note that here are two fields: one for the minimum and one for the maximum value.
T STEP [Parametric only]	Sets the increment between consecutive T-values.
θ RNG [Polar only]	Sets the range of angle values to be plotted. Note that here are two fields: one for the minimum and one for the maximum value.
θ Step [Polar only]	Sets the increment between consecutive angle values.
SEQ PLOT [Sequence only]	Sets the type of plot: Stairstep or Cobweb.
N RNG [Sequence only]	Sets the range of N-values to be plotted. Note that here are two fields: one for the minimum and one for the maximum value.
H WIDTH [Stats 1 Var only]	Sets the width of the bars in a histogram.
H RNG [Stats 1 Var only]	Sets the range of values to be included in a histogram. Note that here are two fields: one for the minimum and one for the maximum value.
X RNG	Sets the initial range of the x-axis. Note that here are two fields: one for the minimum and one for the maximum value. In Plot view the range can be changed by panning and zooming.
Y RNG	Sets the initial range of the y-axis. Note that there are two fields: one for the minimum and one for the maximum value. In Plot view the range can be changed by panning and zooming.

Setup field	Purpose
X TICK	Sets the increment between tick marks on the x-axis.
Y TICK	Sets the increment between tick marks on the y-axis.

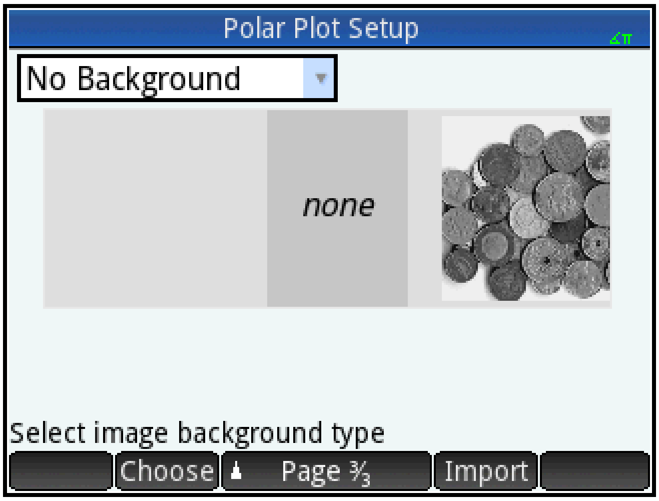
Page 2

Setup field	Purpose
AXES	Shows or hides the axes.
LABELS	Places values at the ends of each axis to show the current range of values.
GRID DOTS	Places a dot at the intersection of each horizontal and vertical grid line.
GRID LINES	Draws a horizontal and vertical grid line at each integer x-value and y-value.
CURSOR	Sets the appearance of the trace cursor: standard, inverting, or blinking.
CONNECT [Stats 2 Var only]	Connects the data points with straight segments.
METHOD [Not in either statistics app]	Sets the graphing method to adaptive, fixed-step segments, or fixed-step dots. Explained below.

Page 3

Some HP Prime apps support the use of a background image in Plot view. Page 3 of the Plot Setup menu can be used to select the image and configure its appearance in Plot view for those apps.

▲ To open the background image menu, press Shift Plot Setup, and then tap Page 1/3 ▼ twice.



No Background

none

Select image background type

To configure a background image:

1. Select the size and position of the background. The options are as follows:

    - **No Background**—By default, no background image is used.
    - **Centered**—The selected image is centered, both vertically and horizontally, in Plot view.
    - **Stretched**—The selected image is stretched, both vertically and horizontally, to fit the entire display in Plot view.
    - **Best fit**—The select image is stretched either horizontally or vertically to fit either the x- or y-dimension in Plot view.
    - **XY Range**—You must enter an x- and y-range to position the image in plot view.

2. Enter an integer between 0 and 100 in the **Opacity** box. 0 is transparent; 100 is completely opaque.
3. Select the background image. All images associated with the app are displayed, followed by all images built into the calculator memory. Swipe left or right to view the available images, and then tap an image.

The background image is now visible in Plot view.

If you drag an axis or perform a 2-finger pinch zoom gesture, you can scroll to a particular feature or zoom in or out on the image if you selected the XY Range option. Otherwise, the image does not change if the Plot view dimensions change.

Page 3 of Plot Setup also allows you to import an image from another HP Prime app.

To import an image from another HP Prime app:

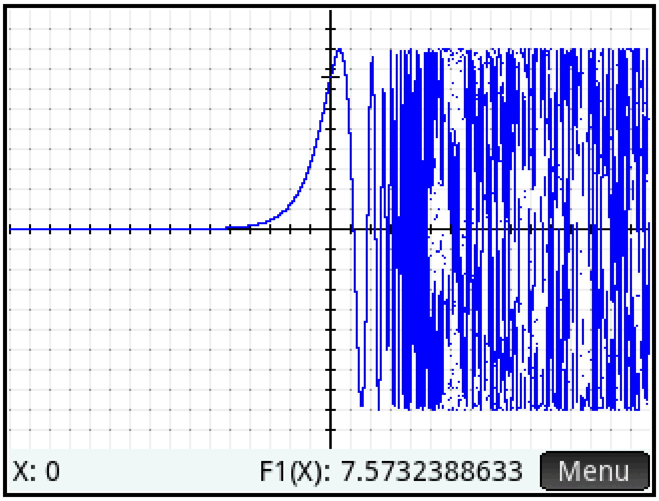
1. Tap Import.
2. Select an HP Prime app.
3. Swipe left or right to view all images associated with the app.
4. Tap an image, and then tap OK to import the image into the current app.

For more information on how to associate an image with an HP Prime app, see the *HP Connectivity Kit User Guide*.

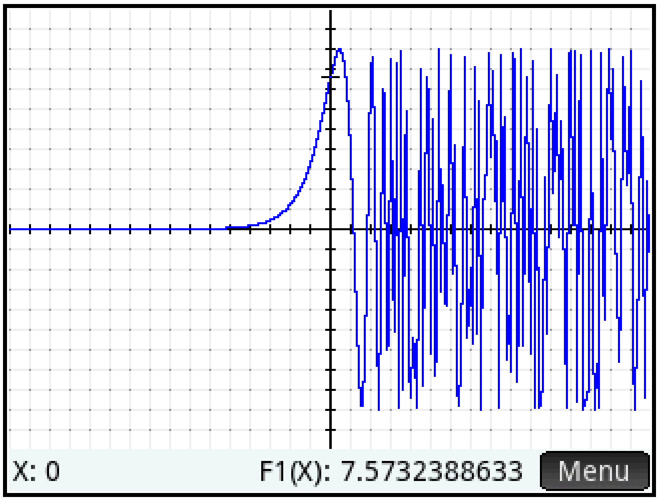
Graphing methods

The HP Prime calculator gives you the option of choosing one of three graphing methods. The methods are described following, with each applied to the function f(x) = 9*sin(e^x).

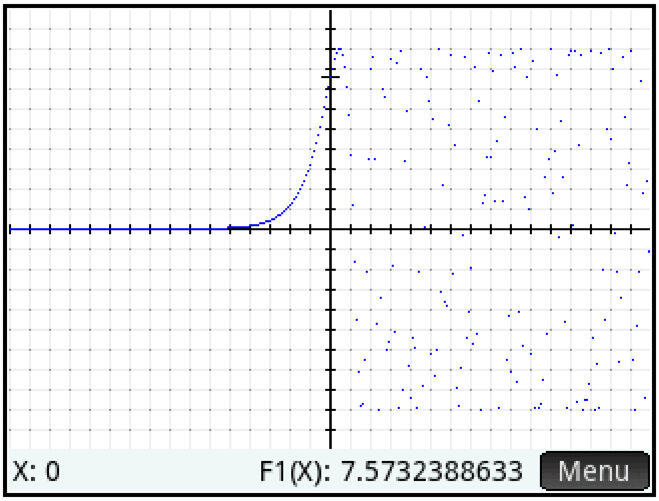
- **Adaptive**—This gives very accurate results and is used by default. With this method active, some complex functions may take a while to plot. In these cases, Sto ▸ appears on the menu bar, enabling you to stop the plotting process if you wish.



- **Fixed-step segments**—This method samples x-values, computes their corresponding y-values, and then plots and connects the points.



- **Fixed-step dots**—This works like fixed-step segments method but does not connect the points.



Restoring default settings

This section applies to the Advanced Graphing, Function, Parametric, Polar, Sequence, Solve, Statistics 1 Var, Statistics 2Var, and Geometry apps.

To restore one field to its default setting:

1. Select the field.
2. Press Del.

To restore all default settings, press Shift Esc Clear.

Common operations in Numeric view

This section applies to the Advanced Graphing, Function, Parametric, and Polar apps.

Numeric view functions that are common to many apps are described in detail in this section. Functions that are available only in a particular app are described in the chapter dedicated to that app.

Numeric view provides a table of evaluations. Each definition in Symbolic view is evaluated for a range of values for the independent variable. You can set the range and fineness of the independent variable, or leave it to the default settings.

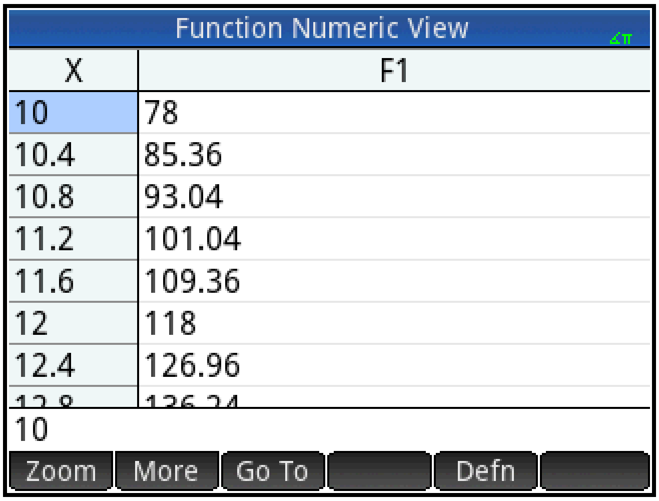
Press Num Setup to open Numeric view.

Zoom

Unlike in Plot view, zooming in Numeric view does not affect the size of what is displayed. Instead, it changes the increment between consecutive values of the independent variable (that is, the **numstep** setting in the Numeric Setup view: see Common operations in Numeric Setup view on page 103). Zooming in decreases the increment; zooming out increases the increment. The row that was highlighted before the zoom remains unchanged.

For the ordinary zoom in and zoom out options, the degree of zooming is determined by the zoom factor. In Numeric view this is the **numzoom** field in the Numeric Setup view. The default value is 4. Thus if the current increment (that is, the **numstep** value) is 0.4, zooming in will further divide that interval by four smaller intervals. So instead of x-values of 10, 10.4, 10.8, 11.2, and so on, the x-values will be 10, 10.1, 10.2, 10.3, 10.4, and so on. (Zooming out does the opposite: 10, 10.4, 10.8, 11.2, and so on becomes 10, 11.6, 13.2, 14.8, 16.4, and so on).

Figure 6-1 Before zooming

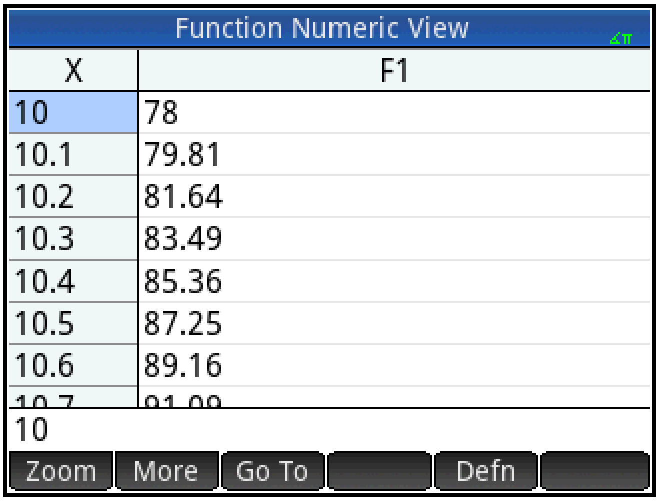


Function Numeric View

X	F1
10	78
10.4	85.36
10.8	93.04
11.2	101.04
11.6	109.36
12	118
12.4	126.96
12.8	136.24

10

Figure 6-2 After zooming



X	F1
10	78
10.1	79.81
10.2	81.64
10.3	83.49
10.4	85.36
10.5	87.25
10.6	89.16
10.7	91.09

10

Zoom options

In Numeric view, several zoom methods can be used.

- 2-finger pinch zoom performed vertically
- Keyboard
- Zoom menu in Numeric view

NOTE: Any zooming you do in Numeric view does not affect Plot view, and vice versa. However, if you choose a zoom option from the **Views** menu (View Copy) while you are in Numeric view, Plot view is displayed with the plots zoomed accordingly. In other words, the zoom options on the **Views** menu apply only to Plot view.

Zooming in Numeric view automatically changes the **numstep** value in the Numeric Setup view.

Zoom gestures

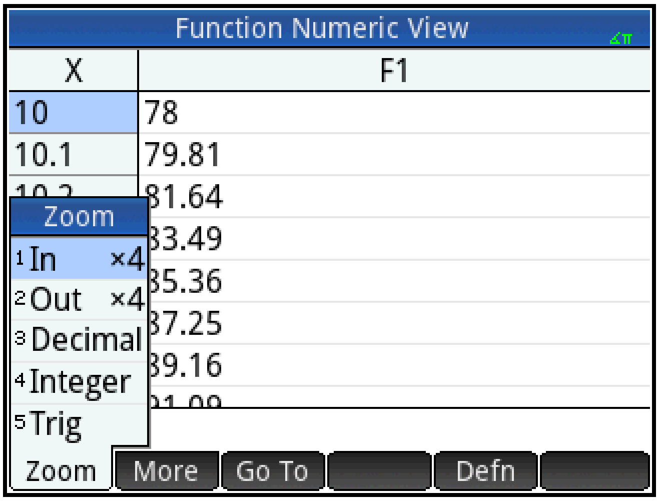
In Numeric view, a 2-finger pinch zoom performed vertically zooms the selected row. A zoom in decreases the common difference in the x-values, and a zoom out increases the common difference in the x-values.

Zoom keys

There are two zoom keys: pressing + zooms in and pressing − zooms out. The extent of the scaling is determined by the numzoom setting (explained previously).

Zoom menu

In Numeric view, tap Zoom, and then tap an option.



X	F1
10	78
10.1	79.81
10.2	81.64
	83.49
	85.36
	87.25
	89.16
	91.09

The zoom options are explained in the following table.

Option	Result
In	The increment between consecutive values of the independent variable becomes the current value divided by the **numzoom** setting. (Shortcut: press + Ans ;.)
Out	The increment between consecutive values of the independent variable becomes the current value multiplied by the **numzoom** setting. (Shortcut: press − Base :.)
Decimal	Restores the default **numstart** and **numstep** values: 0 and 0.1 respectively.
Integer	The increment between consecutive values of the independent variable is set to 1.
Trig	● If the angle measure setting is radians, sets the increment between consecutive values of the independent variable to π/24 (approximately 0.1309). ● If the angle measure setting is degrees, sets the increment between consecutive values of the independent variable to 7.5.
Undo Zoom	Returns the display to the previous settings (**numstart** and **numstep** values). **NOTE:** This option is only available after a zoom operation has been performed.

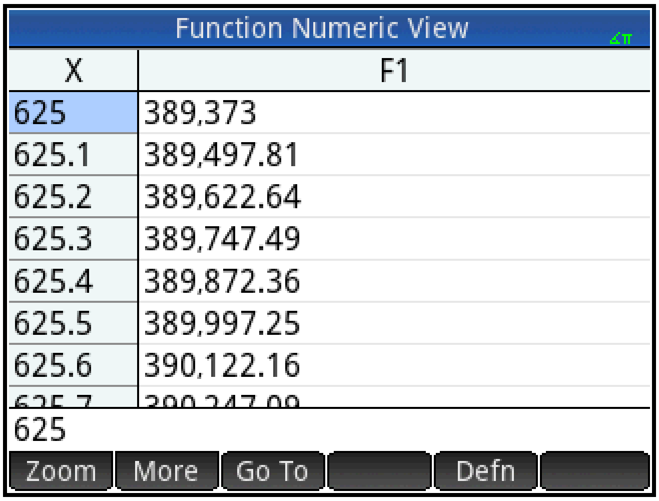
Evaluating

You can step through the table of evaluations in Numeric view by pressing ▲ or ▼. You can also quickly jump to an evaluation by entering the independent variable of interest in the independent variable column and tapping OK.

For example, suppose in the Symbolic view of the Function app, you have defined F1(X) as $(X – 1)^2 – 3$. Suppose further that you want to know what the value of that function is when X is 625.

1. Open Numeric view (Num Setup).
2. Anywhere in the independent column—the left-most column—enter 625.
3. Tap OK.

Numeric view is refreshed, with the value you entered in the first row and the result of the evaluation in a cell to the right. In this example, the result is 389373.



X	F1
625	389,373
625.1	389,497.81
625.2	389,622.64
625.3	389,747.49
625.4	389,872.36
625.5	389,997.25
625.6	390,122.16
625.7	390,247.09

625

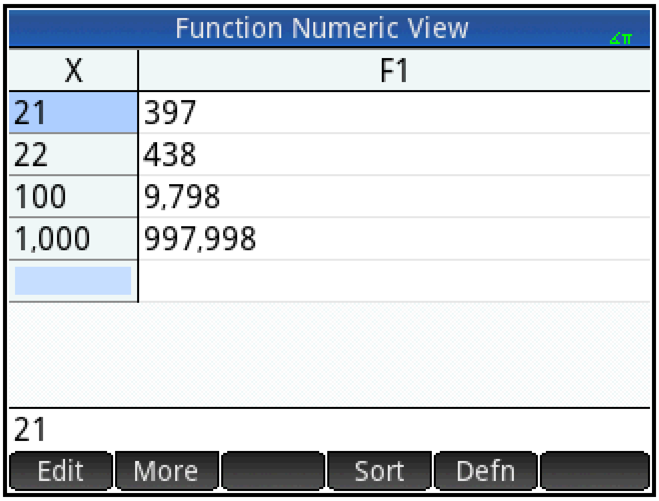
You can also tap Go To and enter a value for the independent variable. Then tap OK to reconfigure the table using the new value.

Custom tables

If you choose **Automatic** for the **numtype** setting, the table of evaluations in Numeric view will follow the settings in the Numeric Setup view. That is, the independent variable will start with the **numstart** setting and increment by the **numstep** setting. (These settings are explained in Common operations in Numeric Setup view on page 103.) However, you can choose to build your own table where just the values you enter appear as independent variables.

1. Open Numeric Setup view (Shift Num Setup).

2. Choose **BuildYourOwn** from the **numtype** menu.



X	F1
21	397
22	438
100	9,798
1,000	997,998

21

3. Open Numeric view (Num Setup).

   Numeric view is empty.

4. In the independent column—the left-most column—enter a value of interest.

5. Tap OK.

6. If you still have other values to evaluate, repeat from step 4.

Deleting data

To delete one row of data in your custom table, place the cursor in that row and press Del.

To delete all the data in your custom table:

1. Press Shift Esc Clear.

2. Tap OK or press Enter ≈ to confirm your intention.

Copy and paste in Numeric view

Copying and pasting a cell

In Numeric view, you can copy and paste the value of any cell.

1. To copy a cell, tap the cell and then press Shift View Copy.

2. To paste the cell to a box or other location, move your cursor to the location and then press Shift Menu Paste.

Copying and pasting a row

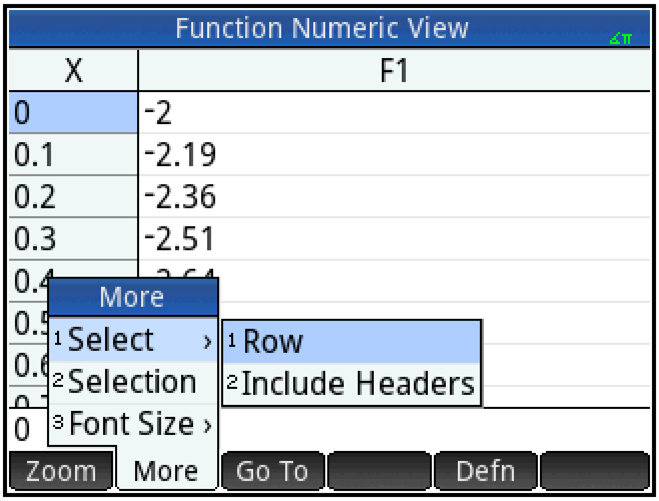
You can copy and paste an entire row, either with or without column headers, using the More menu.

The following example uses the automatic table based on $F1(X)=(X-1)^2-3$.

To copy the second row in the table with headers:

1. Tap the second row.

2. Tap More, tap **Select**, and then tap **Include Headers**.



Function Numeric View

X	F1
0	-2
0.1	-2.19
0.2	-2.36
0.3	-2.51

The second row with headers is now copied to the clipboard.

To paste the row with headers into the Spreadsheet app:

1. Open the Spreadsheet app.
2. Tap the cell where you want the pasted row to start.
3. To open the clipboard, press Shift Menu Paste.
4. Tap the row (in this example, it is the first entry), and then select **Grid data**.

The row with headers is now pasted into the spreadsheet, starting at the selected cell.

Copying and pasting an array of cells

You can copy and paste a rectangular array of cells.

1. Tap and hold a corner cell, and then drag your finger to select more cells.
2. After you have selected all the cells, press Shift View Copy .
3. Navigate to the paste location.
4. Press Shift Menu Paste.
5. Tap the rectangular array (in this example, it is the first entry), and then select **2-dimensional**.

The rectangular array is now pasted, starting at the selected location. You can also use the More menu to change the selection mode, such as only needing a drag gesture to select.

Numeric view: Summary of menu buttons

Button	Purpose
Zoom	Modifies the increment between consecutive values of the independent variable in the table of evaluations. See Zoom on page 97.

Button	Purpose
Edit (BuildYourOwn only)	Copies the highlighted item into the entry line to enable editing.
More	Displays an editing options menu. See More menu on page 103.
Go To	Moves the cursor to the specified item in a list.
Sort (BuildYourOwn only)	Sorts the data in ascending or descending order.
Defn	Displays the definition of the selected column.

More menu

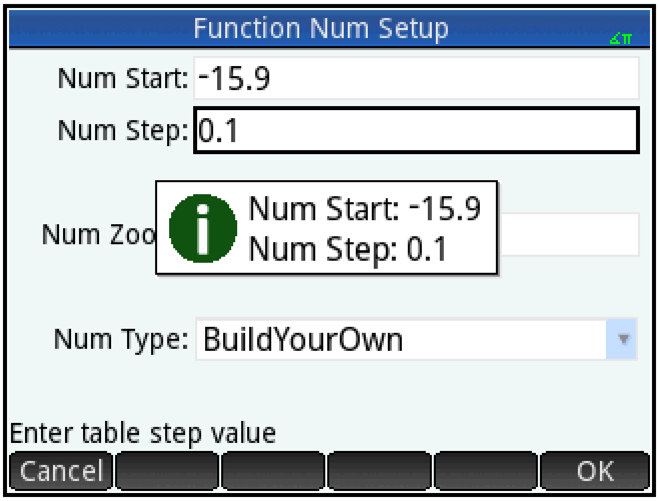
The More menu contains options for editing lists of data. The options are described in the following table.

Option	Sub-option	Purpose
Insert (BuildYourOwn only)	Row	Inserts a new row in the selected list. The new row contains 0 as its element.
Delete (BuildYourOwn only)	Column	Deletes the contents of the selected list. To delete a single item, select it and then press Del .
Select	Row	Selects the row that contains the currently selected cell; the entire row can then be copied.
	Swap Ends	After a multi-cell selection is made, this option appears. It transposes the values of the first and last cells of the current selection.
	Include Headers	Selects the row and row headers that contain the currently selected cell; the entire selection can then be copied.
Selection		Turns selection mode on and off. If selection mode is off, you can tap and hold a cell and then drag your finger to select a rectangular array.
Font Size	Small	Enables the small font.
	Medium	Enables the medium font.
	Large	Enables the large font.

Common operations in Numeric Setup view

Select the field you want to change and either specify a new value, or if you are choosing a type of table for Numeric view—automatic or build-your-own—choose the appropriate option from the **Num Type** menu.

To help you set a starting number and increment that matches the current Plot view, tap Plot→.



Function Num Setup

Num Start: -15.9

Num Step: 0.1

Num Zoo

Num Start: -15.9
Num Step: 0.1

Num Type: BuildYourOwn

Enter table step value

Restoring default settings

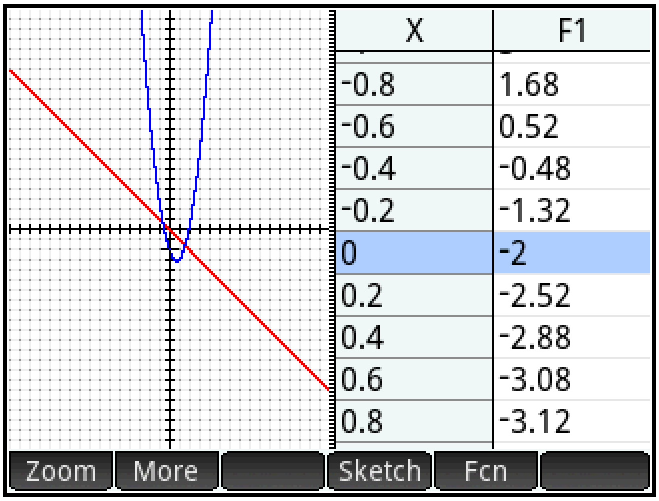
To restore one field to its default setting:

1. Select the field.
2. Press Del.

To restore all default settings, press Shift Esc Clear.

Combining Plot and Numeric Views

You can display Plot view and Numeric view side-by-side. Moving the tracing cursor causes the table of values in Numeric view to scroll. You can also enter a value in the X column. The table scrolls to that value, and the tracing cursor jumps to the corresponding point on the selected plot.



X	F1
-0.8	1.68
-0.6	0.52
-0.4	-0.48
-0.2	-1.32
0	-2
0.2	-2.52
0.4	-2.88
0.6	-3.08
0.8	-3.12

▲ To combine Plot and Numeric view in a split screen, press View Copy and select **Split Screen: Plot Table**.

▲ To return to Plot view, press Num Setup. To return to Numeric view by pressing Num Setup.

Adding a note to an app

You can add a note to an app. Unlike general notes (created via the Note Catalog) an app note is not listed in the Note Catalog. It can only be accessed when the app is open.

An app note remains with the app if the app is sent to another calculator.

To add a note to an app:

1. Open the app.
2. Press Shift Apps Info.

   If a note has already been created for this app, its contents are displayed.
3. Tap Edit and start writing (or editing) your note.

   The format and bullet options available are the same as those in the Note Editor.
4. To exit the note screen, press any key. Your note is automatically saved.

Creating an app

The apps that come with the HP Prime calculator are built in and cannot be deleted. They are always available (simply by pressing Apps Info). However, you can create any number of customized instances of most apps. You can also create an instance of an app that is based on a previously customized app. Customized apps are opened from the application library in the same way that you open a built-in app.

The advantage of creating a customized instance of an app is that you can continue to use the built-in app for some other problem and return to the customized app at any time with all its data still in place. For example, you could create a customized version of the Sequence app that enables you to generate and explore the Fibonacci series. You could continue to use the built-in Sequence app to build and explore other sequences and return, as needed, to your special version of the Sequence app when you next want to explore the Fibonacci series. Or you could create a customized version of the Solve app—named, for example, Triangles—in which you set up, just once, the equations for solving common problems involving right-angled triangles (such as H = O/SIN(θ), A = H*COS(θ), O = A*TAN(θ), and so on). You could continue to use the Solve app to solve other types of problems but use your Triangle app to solve problems involving right-angled triangles. Just open Triangles, select which equation to use—you won’t need to re-enter them—enter the variables you know, and then solve for the unknown variable.

Like built-in apps, customized apps can be sent to another HP Prime calculator. Customized apps can also be reset, deleted, and sorted just as built-in apps can (as explained earlier in this chapter).

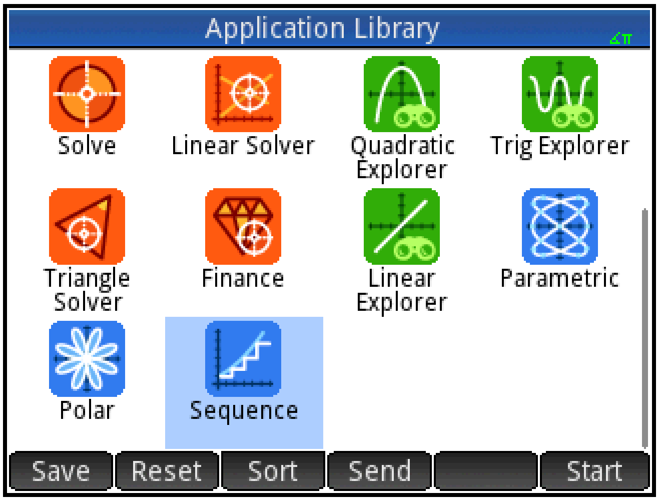
Note that the only apps that cannot be customized are the following:

- Linear Explorer
- Quadratic Explorer
- Trig Explorer

Example

Suppose you want to create a customized app that is based on the built-in Sequence app. The app will enable you to generate and explore the Fibonacci series.

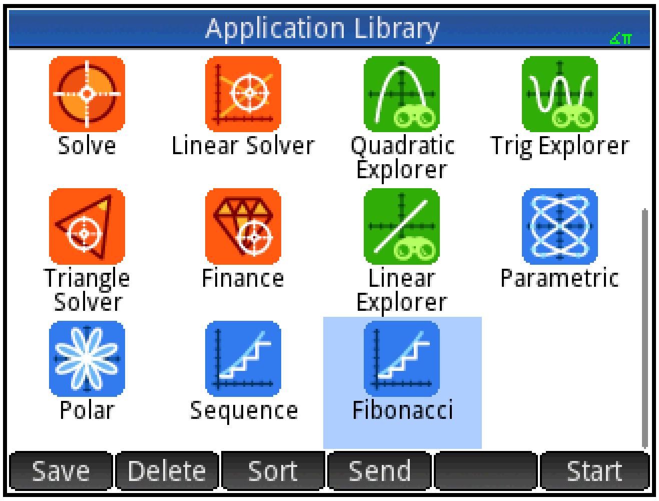
1. Press Apps Info and use the cursor keys to highlight the Sequence app. Don't open the app.



2. Tap Save. This enables you to create a copy of the built-in app and save it under a new name. Any data already in the built-in app is retained, and you can return to it later by opening the Sequence app.

3. In the Name field, enter a name for your new app—say, `Fibonacci`—and press Enter ≈ twice.

   Your new app is added to the Application Library. Note that it has the same icon as the parent app—Sequence—but with the name you gave it: **Fibonacci** in this example.



4. You are now ready to use this app just as you would the built-in Sequence app. Tap on the icon of your new app to open it. You will see in it all the same views and options as in the parent app.

In this example we have used the Fibonacci series as a potential topic for a customized app. The Fibonacci series can be created inside the Sequence app—or an app based on the Sequence app.

As well as cloning a built-in app—as described above—you can modify the internal workings of a customized app using the HP Prime programming language.

App functions and variables

Functions

App functions are used in HP apps to perform common calculations. For example, in the Function app, the Plot view **Fcn** menu has a function called **SLOPE** that calculates the slope of a given function at a given point. The **SLOPE** function can also be used from the Home view or a program.

For example, suppose you want to find the derivative of x^2 – 5 at x = 2. One way, using an app function, is as follows:

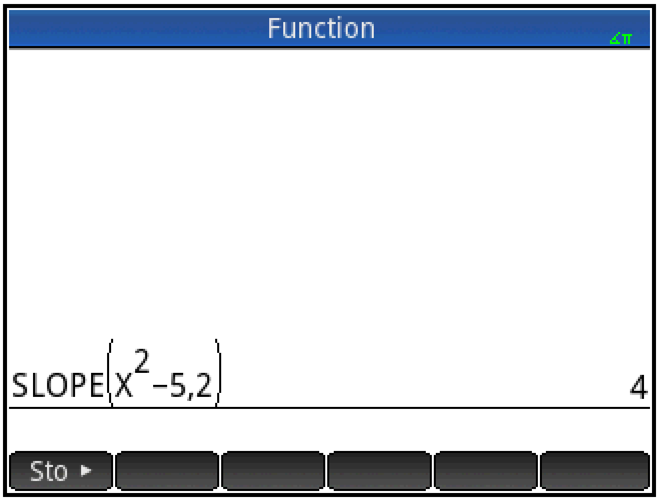
1. Press Mem B.
2. Tap App and select **Function** > **SLOPE**.

   SLOPE() appears on the entry line, ready for you to specify the function and the x-value.
3. Enter the function:

   ALPHA alpha × ∡ X x^2 √ L − Base ⁝ 5
4. Enter the parameter separator:

   , ⅹ Eval O
5. Enter the x-value and press Enter ≈.

   The slope (that is the derivative) at x = 2 is calculated: 4.



```
SLOPE(X^2−5,2)                                    4
```

Variables

All apps have variables, that is, placeholders for various values that are unique to a particular app. These include symbolic expressions and equations, settings for the Plot and Numeric views, and the results of some calculations such as roots and intersections.

Suppose you are in Home view and want to retrieve the mean of a data set recently calculated in the Statistics 1Var app.

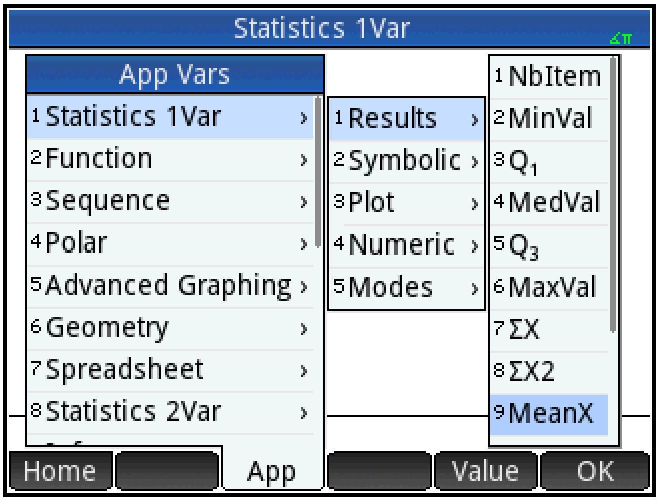
1. Press Vars.

   This opens the Variables menu. From here you can access Home variables, user-defined variables, and app variables.

2. Tap App.

   This opens a menu of app variables.

3. Select **Statistics 1Var > results > MeanX**.



The current value of the variable you chose now appears on the entry line. You can press Enter to see its value. Or you can include the variable in an expression that you are building. For example, if you wanted to calculate the square root of the mean computed in the Statistics 1Var app, you would first press Shift x^2, follow steps 1 to 3 above, and then press Enter.

Qualifying variables

You can qualify the name of any app variable so that it can be accessed from anywhere on the HP Prime calculator. For example, both the Function app and the Parametric app have an variable named **Xmin**. If the app you last had open was the Parametric app and enter **Xmin** in Home view, you will get the value of **Xmin** from the Parametric app. To get the value of **Xmin** in the Function app instead, you could open the Function app and then return to Home view. Alternatively, you could qualify the name of the variable by preceding it with the app name and a period, as in **Function.Xmin**.

7 Function app

The Function app enables you to explore up to 10 real-valued, rectangular functions of y in terms of x; for example, $y = 1 – x$ and $y = (x – 1)^2 – 3$.

Once you have defined a function you can do the following:

- Create graphs to find roots, intercepts, slope, signed area, and extrema
- Create tables that show how functions are evaluated at particular values

This chapter demonstrates the basic functions of the Function app by stepping you through an example. The HP Prime calculator can perform more complex functions.

Getting started with the Function app

The Function app uses the customary app views: Symbolic, Plot, and Numeric.

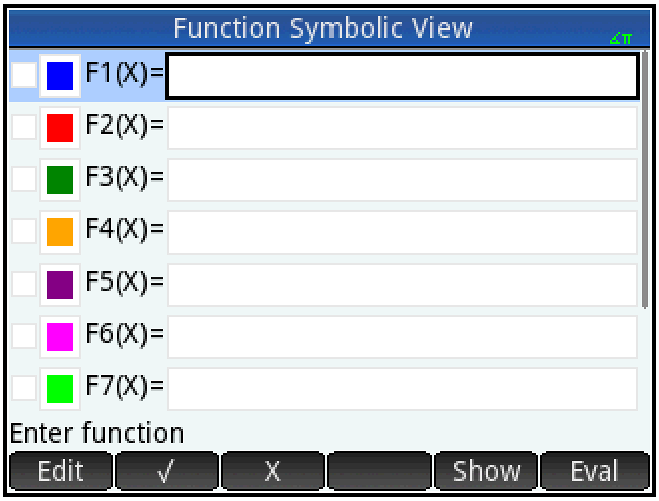
The standard Symbolic, Plot, and Numeric view menu buttons are available.

Throughout this chapter, we will explore the linear function $y = 1 – x$ and the quadratic function $y = (x – 1)^2 – 3$.

Opening the Function app

▲ Press Apps Info , and then select Function to open the **Function** app.

Remember that you can open an app by tapping its icon. You can also open it by using the cursor keys to highlight it and then pressing Enter ≈ .



Function Symbolic View

□ ■ F1(X)=

□ ■ F2(X)=

□ ■ F3(X)=

□ ■ F4(X)=

□ ■ F5(X)=

□ ■ F6(X)=

□ ■ F7(X)=

Enter function

The Function app starts in Symbolic view. This is the “defining view.” It is where you symbolically define (that is, specify) the functions you want to explore.

The graphical and numerical data you see in Plot view and Numeric view are derived from the symbolic expressions defined here.

Defining the expressions

There are 10 fields for defining functions. These are labeled F1(X) through F9(X) and F0(X).

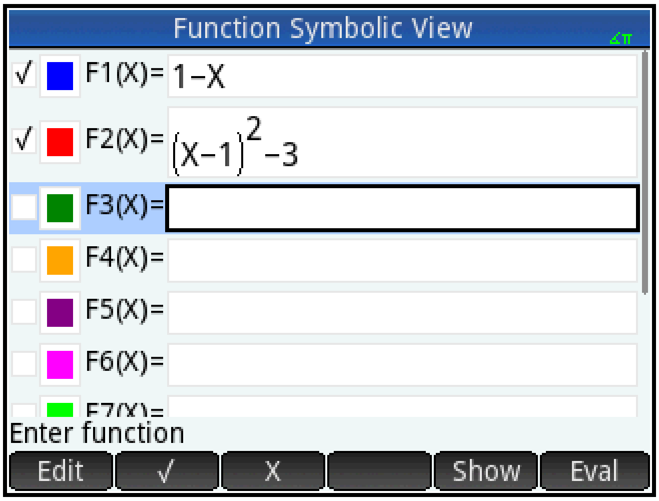
1. Highlight the field you want to use, either by tapping on it or scrolling to it. If you are entering a new expression, just start typing. If you are editing an existing expression, tap Edit and make your changes. When you have finished defining or changing the expression, press Enter.

2. Enter the linear function in F1(X).

   1 − xtθn Enter

3. Enter the quadratic function in F2(X).

   ( ) xtθn − 1 ▶ x^2 − 3 Enter



Function Symbolic View

✓ 🟦 F1(X)= 1−X

✓ 🟥 F2(X)= $(X-1)^2-3$

□ 🟩 F3(X)=

□ 🟧 F4(X)=

□ 🟪 F5(X)=

□ 🟪 F6(X)=

□ 🟩 F7(X)=

Enter function

NOTE: You can tap the X button to assist in the entry of equations. In the Function app, it has the same effect as pressing xtθn. (In other apps, xtθn enters a different character.)

4. Do one of the following:

   - Give one or more functions a custom color when it is plotted.
   - Evaluate a dependent function.
   - Deselect a definition you do not want to explore.
   - Incorporate variables, math commands, and CAS commands in a definition.

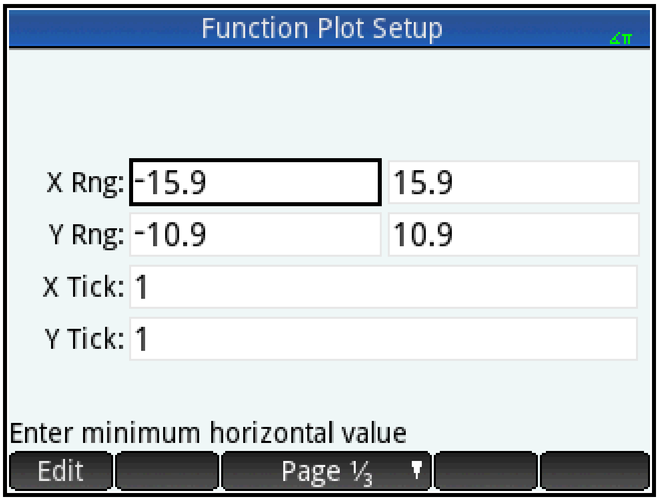
   For the sake of simplicity we can ignore these operations in this example. However, they can be useful and are common Symbolic view operations.

Setting up a plot

You can change the range of the x- and y-axes and the spacing of the tick marks along the axes.

▲ Display Plot Setup view.

Shift Plot ↳Setup



Function Plot Setup

X Rng: -15.9 15.9

Y Rng: -10.9 10.9

X Tick: 1

Y Tick: 1

Enter minimum horizontal value

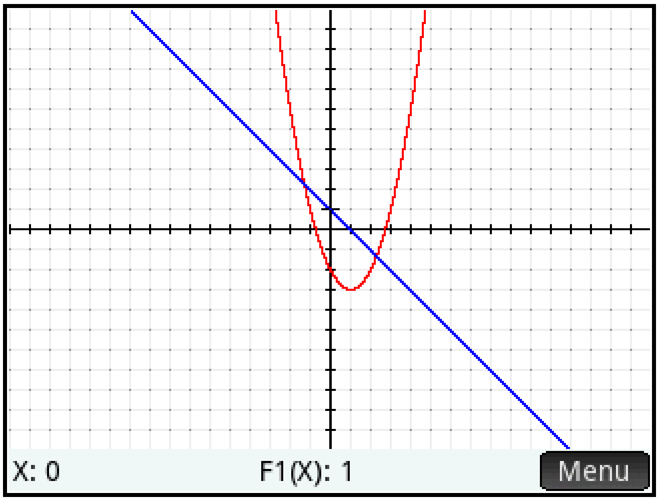
For this example, you can leave the plot settings at their default values. If your settings do not match those in the illustration above, press Shift Esc Clear to restore the default values.

You can use the common Plot view operations to change the appearance of plots.

Plotting a function

▲ Plot the function.

Plot ↳Setup



X: 0 F1(X): 1

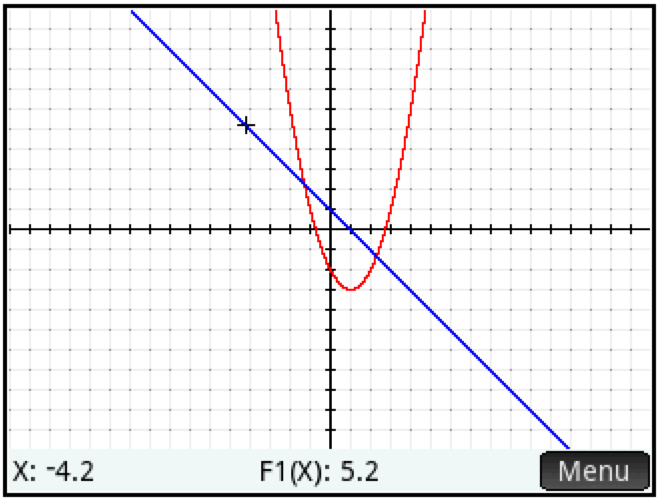
Tracing a graph

By default, the trace function is active. This enables you to move a cursor along a graph. If more than two graphs are shown, the graph that is the highest in the list of functions in Symbolic view is the graph that will be traced by default. Since the linear equation is higher than the quadratic function in Symbolic view, it is the graph on which the tracing cursor appears by default.

1. Trace the linear function.

   ▶ or ◀

   Note how a cursor moves along the plot as you press the buttons. Note too that the coordinates of the cursor appear at the bottom of the screen and change as you move the cursor.



X: -4.2 F1(X): 5.2

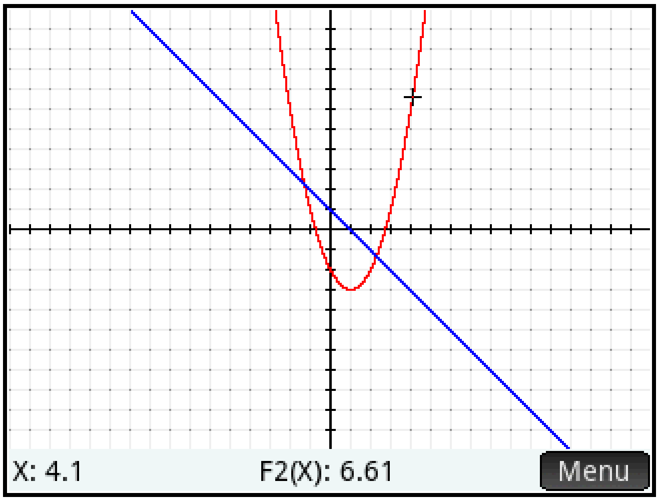
2. Move the tracing cursor from the linear function to the quadratic function.

   ▲ or ▼

3. Trace the quadratic function.

▶ or ◀

Again notice how the coordinates of the cursor appear at the bottom of the screen and change as you move the cursor.



Changing the scale

You can change the scale to see more or less of your graph. This can be done in a number of ways:

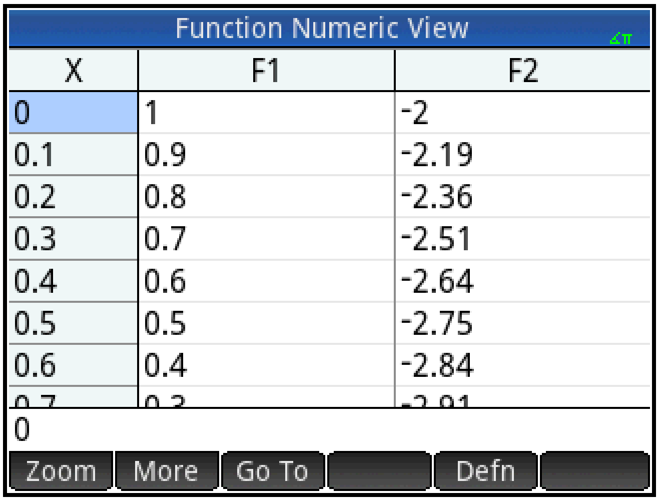
- Use a 2-finger pinch zoom performed diagonally to zoom on the x- and y-axis simultaneously.
- Use a 2-finger pinch zoom performed horizontally to zoom on the x-axis.
- Use a 2-finger pinch zoom performed vertically to zoom on the y-axis.
- Press + to zoom in or − to zoom out on the current cursor position. This method uses the zoom factors set in the **Zoom** menu. The default for both x and y is 2.
- Use the Plot Setup view to specify the exact x-range (**X RNG**) and y-range (**Y RNG**) you want.
- Use options on the **Zoom** menu to zoom in or out, horizontally or vertically, or both..
- Use options on the **View** menu (View) to select a predefined view. Note that the **Autoscale** option attempts to provide a best fit, showing as many of the critical features of each plot as possible.

NOTE: By dragging a finger horizontally or vertically across the screen, you can quickly see parts of the plot that are initially outside the set x and y ranges. This is easier than resetting the range of an axis.

Displaying Numeric view

▲ Display the Numeric view.

Num



Function Numeric View

X	F1	F2
0	1	-2
0.1	0.9	-2.19
0.2	0.8	-2.36
0.3	0.7	-2.51
0.4	0.6	-2.64
0.5	0.5	-2.75
0.6	0.4	-2.84
0.7	0.3	-2.91

0

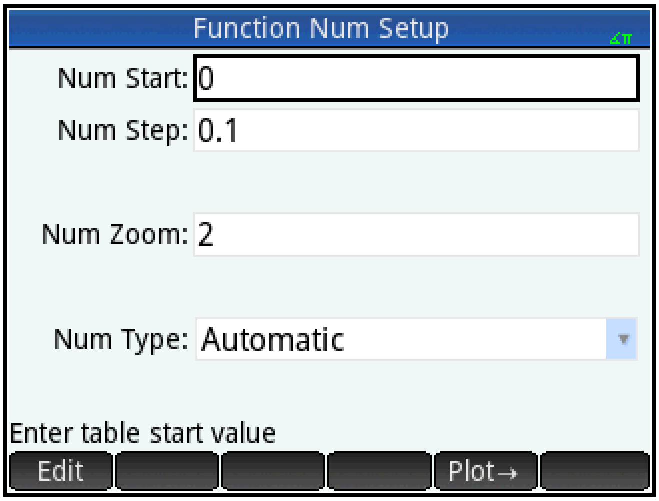
The Numeric view displays data generated by the expressions you defined in Symbolic view. For each expression selected in Symbolic view, Numeric view displays the value that results when the expression is evaluated for various x-values.

For more information about the available buttons, see *Numeric view: Summary of menu buttons* in the *Introduction to HP apps* chapter.

Setting up Numeric view

1. Display the Numeric Setup view.

   Shift Num Setup



Function Num Setup

Num Start: 0

Num Step: 0.1

Num Zoom: 2

Num Type: Automatic

Enter table start value

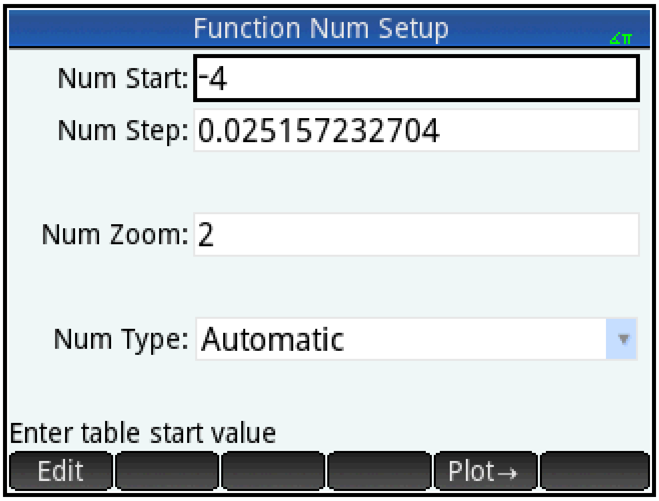
   You can set the starting value and step value (that is, the increment) for the x-column, as well as the zoom factor for zooming in or out on a row of the table. Note that in Numeric view, zooming does not affect the size of what is displayed. Instead, it changes the **Num Step** setting (that is, the increment between consecutive x-values). Zooming in decreases the increment; zooming out increases the increment.

   You can also choose whether the table of data in Numeric view is automatically populated or whether it is populated by you typing in the particular x-values you are interested in. These options—**Automatic** or **BuildYourOwn**—are available from the **Num Type** list. These are custom table options.

2. Press Shift Esc Clear to reset all the settings to their defaults.

3. Make the Numeric view X-column settings (**Num Start** and **Num Step**) match the tracer x-values (Xmin and pixel width) in Plot view.

   Tap Plot→ OK .



Function Num Setup

Num Start: -4

Num Step: 0.025157232704

Num Zoom: 2

Num Type: Automatic

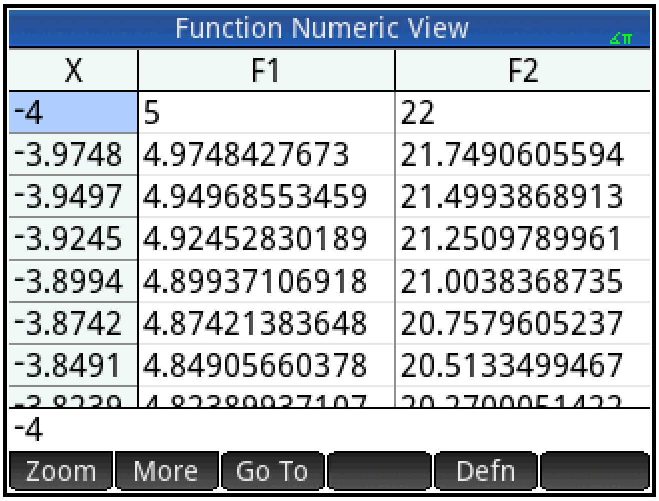
Enter table start value

For example, if you have zoomed in on the plot in Plot view so that the visible x-range is now –4 to 4, this option will set **Num Start** to –4 and **Num Step** to 0.025…

Exploring Numeric view

▲ Display Numeric view.

Num ↪Setup



Function Numeric View

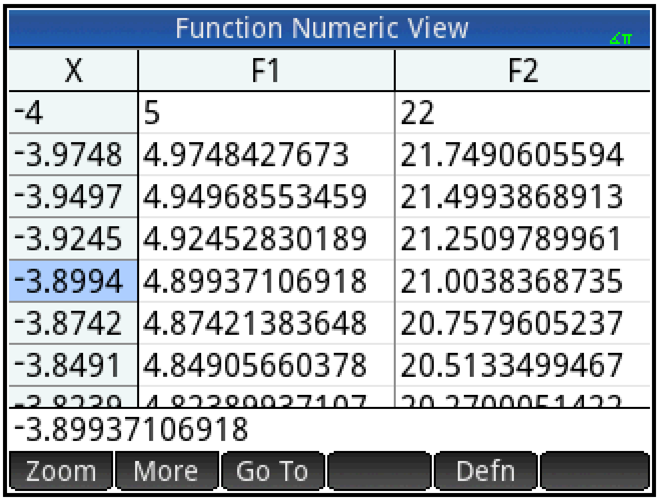
X	F1	F2
-4	5	22
-3.9748	4.9748427673	21.7490605594
-3.9497	4.94968553459	21.4993868913
-3.9245	4.92452830189	21.2509789961
-3.8994	4.89937106918	21.0038368735
-3.8742	4.87421383648	20.7579605237
-3.8491	4.84905660378	20.5133499467
-3.8239	4.82389937107	20.2700051422

-4

Navigating a table

▲ Using the cursor keys, scroll through the values in the independent column (column X). Note that the values in the F1 and F2 columns match what you would get if you substituted the values in the X column for x in the expressions selected in Symbolic view: 1 – x and $(x – 1)^2 – 3$. You can also scroll through the columns of the dependent variables (labeled F1 and F2 in the following figure).

You can also scroll the table vertically or horizontally using tap and drag gestures.



Function Numeric View

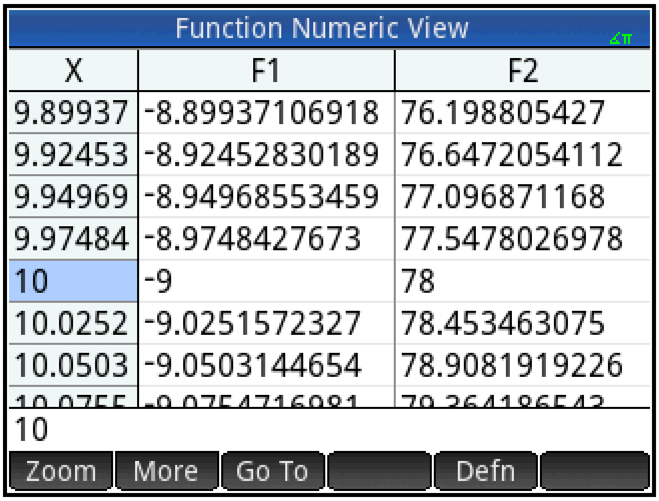
X	F1	F2
-4	5	22
-3.9748	4.9748427673	21.7490605594
-3.9497	4.94968553459	21.4993868913
-3.9245	4.92452830189	21.2509789961
-3.8994	4.89937106918	21.0038368735
-3.8742	4.87421383648	20.7579605237
-3.8491	4.84905660378	20.5133499467
-3.8239	4.82389937107	20.2700051422

-3.89937106918

Going directly to a value

▲ Place the cursor in the X column and type the desired value. For example, to jump straight to the row where x = 10:

10 OK



Function Numeric View

X	F1	F2
9.89937	-8.89937106918	76.198805427
9.92453	-8.92452830189	76.6472054112
9.94969	-8.94968553459	77.096871168
9.97484	-8.9748427673	77.5478026978
10	-9	78
10.0252	-9.0251572327	78.453463075
10.0503	-9.0503144654	78.9081919226
10.0755	-9.0754716981	79.364186543

10

Accessing the zoom options

You can zoom in or out on a selected row in a table using a 2–finger pinch zoom gesture. Zooming in decreases the increment; zooming out increases the increment. The values in the row you zoom in or out on remain the same.

For more precise control over the zoom factor, press + (or −). This zooms in (or out) by the **Num Zoom** value set in the Numeric Setup view. The default value is 4. Thus if the current increment (that is, the **Num Step** value) is 0.4, zooming in on the row whose x-value is 10 will further divide that interval into four smaller intervals. So instead of x-values of 10, 10.4, 10.8, 11.2, and so on, the x-values will be 10, 10.1, 10.2, 10.3, 10.4, and so on. (Zooming out does the opposite: 10, 10.4, 10.8, 11.2, and so on become 10, 11.6, 13.2, 14.8, 16.4, and so on.)

In addition, more zoom options are available by tapping Zoom.

Other options

Numeric view menu options include the following:

- Change the size of the font: small, medium, or large
- Display the definition responsible for generating a column of values

You can also combine the Plot and Numeric views.

Analyzing functions

The Function menu (Fcn) in Plot view enables you to find roots, intersections, slopes, signed areas, and extrema for any function defined in the Function app. You can add a tangent line to a function graph. You can also sketch a function with your finger and then transform the sketch into a function graph with its expression saved in Symbolic view. Then, you can translate and dilate the function, or edit its expression in Plot view.

Displaying the Plot view menu

The **Function** menu is a sub-menu of the Plot view menu. First, display the Plot view menu:

Plot ↳Setup Menu

The menu buttons are as follows.

Button	Purpose
Zoom	Opens the Zoom menu, which contains options for zooming in and out.
Trace	Enables and disables the tracing cursor. If disabled, the cursor can move freely.
Go To	Displays an input form for you to specify the x-value you want to jump to.
Sketch	Starts Sketch Mode, which enables you to sketch a function with your finger.
Fcn	Opens the Function menu. See Summary of FCN operations on page 129.
Menu	Opens or closes the Plot view menu.

Sketching functions

You can sketch a function with your finger and transform the sketch into the graph of a function.

To enter Sketch Mode and save a sketch:

1. In the Plot view menu, tap Sketch.

2. After the menu bar displays **Sketch a function**, use your finger to sketch any of the following function types:

TIP: You can press Esc Clear at any time to cancel the current sketch and exit Sketch Mode.

- **Linear**—m*x + b
- **Quadratic**—a*x^2 + b*x + c
- **Cubic**—a*x^3 + b*x^2 + c*x + d
- **Exponential**—a*e^(b*x + c)
- **Logarithmic**—a*LN(x) + b
- **Sinusoidal**—a*SIN(b*x + c) + d

3. After you lift your finger from the calculator display, the sketch is transformed into a function from one of the listed types. The graph is displayed in a thick line style with the expression displayed at the bottom-left corner of the display. To save this graph and expression in the first available definition (F0–F9) in Symbolic view, tap OK. If you do not want to save this graph and expression, draw a new sketch. It overwrites the existing sketch.

4. After you tap OK, you can continue to sketch more functions.

5. After you are finished sketching, tap OK to exit sketch mode and return to Plot view.

In Plot view, you can tap **Definition** to edit the definition of the sketched function or tap **Transform** to translate and dilate the function.

Modifying function graphs

In the Function menu, the Definition and Transform options allow you to dynamically transform and edit function definitions.

To edit a selected function in Plot view:

1. In the Function menu, tap **Definition** to open the editor.

2. Select one of the following options:

   - Edit—Moves the cursor to the end of the selected definition to allow you to edit it. You can also tap anywhere in the expression to move the cursor to edit the expression. Make your edits, and then tap OK to see the new graph.

   - F1(X)—Opens a list of currently defined functions in Symbolic view. You can then select a function to edit it.

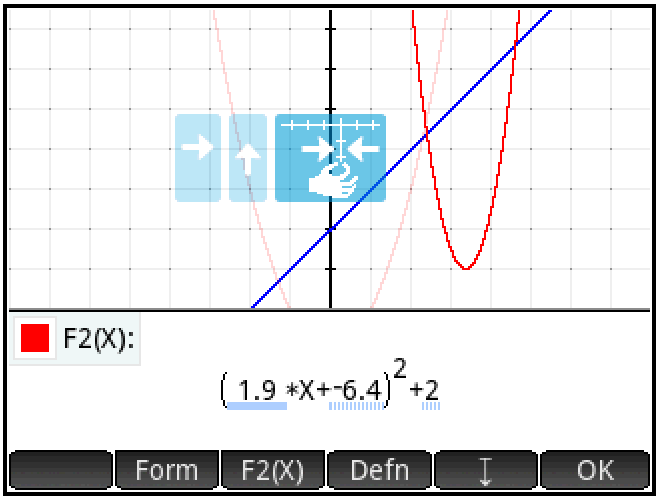
   TIP: The numerical value shown on this button corresponds to the number of currently defined functions in Symbolic view (1–9 and 0).

   - Transf.—Starts Transformation Mode, which enables you to directly translate and dilate the function graph and observe the changes in the parameters of the function definition. You can also select **Transform** in the Function menu.

   - ↓—Closes the editor.

   - Menu—Closes the editor and opens the Plot view menu.

3. If you selected Transf., a white hand on a blue rectangle is displayed.



🟥 F2(X):

$(1.9*X+-6.4)^2+2$

You can drag the graph vertically or horizontally, but not diagonally. The affected parameter in the function definition changes in real time to reflect the translation.

You can also perform a 2-finger pinch zoom horizontally to dilate the graph.

Several indicators help you record the transformations made to the graph:

- Light blue rectangles record the last few transformations, and a dark blue triangle indicates the current transformation.
- All affected parameters are underlined in blue. A dotted blue underline indicates previous transformations, and a dark blue underline indicates the current transformation.
- A transparent version of the original graph is displayed in the background.

After you perform a transformation, the Form button is displayed. Tap the button to select an alternate form for the function definition. The forms available depend on the selected definition.

If you tap Form and change the form of the definition, the Simplify button is displayed. Tap this button to simplify the selected definition. It also rounds the parameter values to one or two decimal places.

4. Tap OK to save your changes.
5. If you need to edit the expression further, tap the expression or tap Defn. Enter the exact expression.
6. Tap OK to save your changes.
7. Tap ↓ or Menu to close the editor.

Finding a root of a quadratic equation

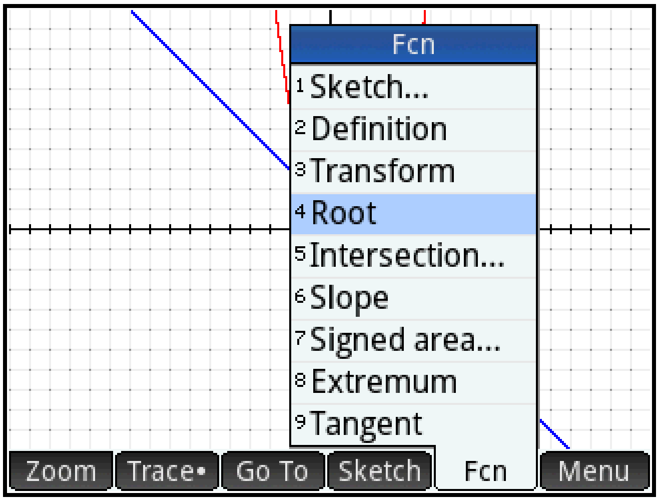
Suppose you want to find the root of the quadratic equation defined earlier. Since a quadratic equation can have more than one root, you will need to move the cursor closer to the root you are interested in than to any other root. In this example, you will find the root of the quadratic close to where x = 3.

1. If it is not already selected, select the quadratic equation:

   ▲ or ▼

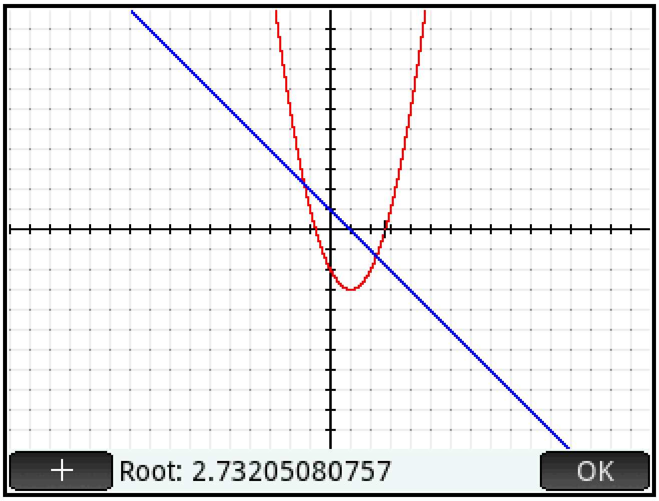
2. Press ▶ or ◀ to move the cursor near to where x = 3.

3. Tap Fcn and select **Root**.

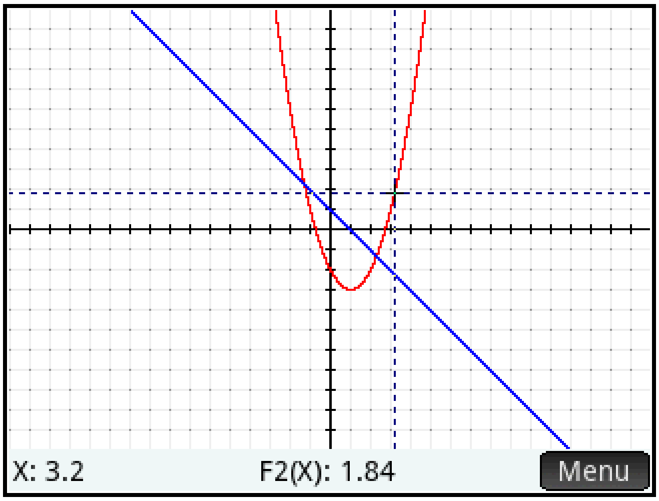


The root is displayed at the bottom of the screen.

If you now move the trace cursor close to x = –1 (the other place where the quadratic crosses the x-axis) and select **Root** again, the other root is displayed.


Root: 2.73205080757

Note the + button. If you tap this button, vertical and horizontal dotted lines are drawn through the current position of the tracer to highlight its position. Use this feature to draw attention to the cursor location. You can also choose a blinking cursor in Plot Setup. Note that the functions in the **Fcn** menu all use the current function being traced as the function of interest and the current tracer x-coordinate as an initial value. Finally, note that you can tap anywhere in Plot view and the tracer will move to the point on the current function that has the same x-value as the location you tapped. This is a faster way of choosing a point of interest than using the trace cursor. (You can move this tracing cursor using the cursor keys if you need finer precision.)



Finding an intersection of two functions

Just as there are two roots of the quadratic equation, there are two points at which both functions intersect. As with roots, you need to position your cursor closer to the point you are interested in. In this example, the intersection close to x = –1 will be determined.

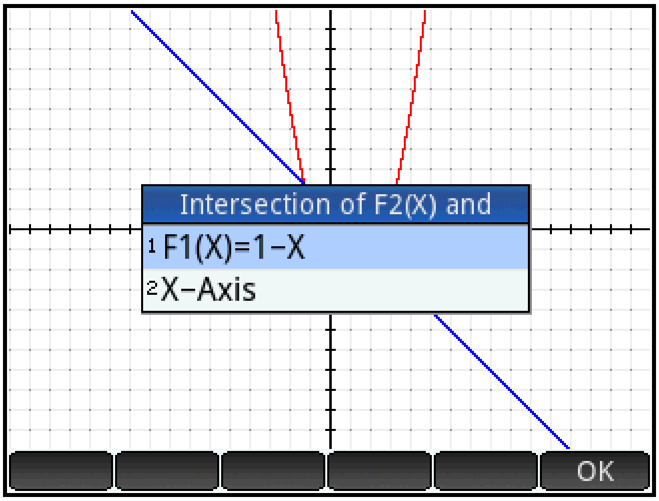
The **Go To** command is another way of moving the trace cursor to a particular point.

1. Tap OK to re-display the menu, tap Go To, enter +/– 1, and tap OK.

   The tracing cursor will now be on one of the functions at x = 1.

2. Tap Fcn and select **Intersection**.

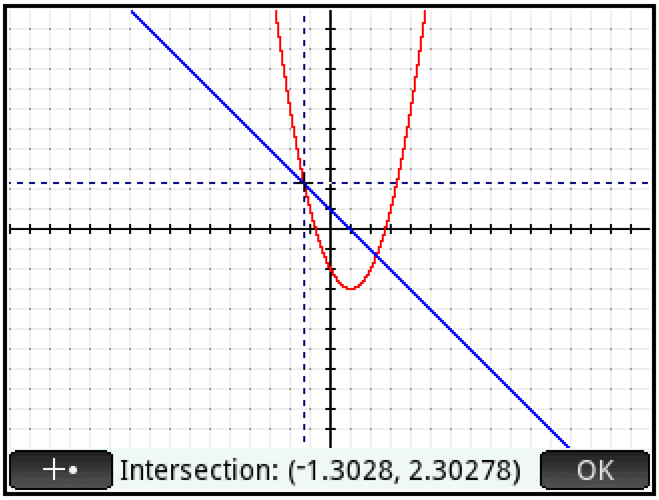
   A list appears giving you a choice of functions and axes.



3. Select the function that has the point of intersection with the currently selected function you wish to find.

   The coordinates of the intersection are displayed at the bottom of the screen.

   Tap + on the screen near the intersection, and repeat from step 2. The coordinates of the intersection nearest to where you tapped are displayed at the bottom of the screen.



Intersection: (-1.3028, 2.30278)

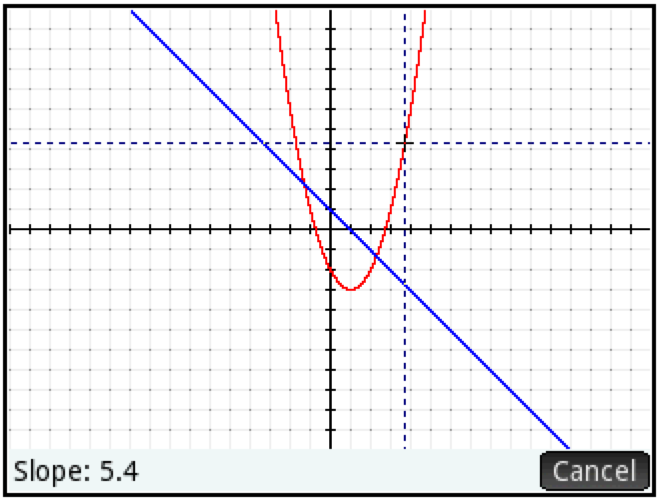
Finding the slope of a quadratic equation

To find the slope of the quadratic function at the intersection point:

1. Tap OK to display the menu, tap Fcn , and then select **Slope**.

   The slope (that is, the gradient) of the function at the intersection point is displayed at the bottom of the screen.

   You can press ◀ or ▶ to trace along the curve and see the slope at other points. You can also press ▼ or ▲ to jump to another function and see the slope at points on it.



Slope: 5.4

2. Press Cancel to display the Plot menu.

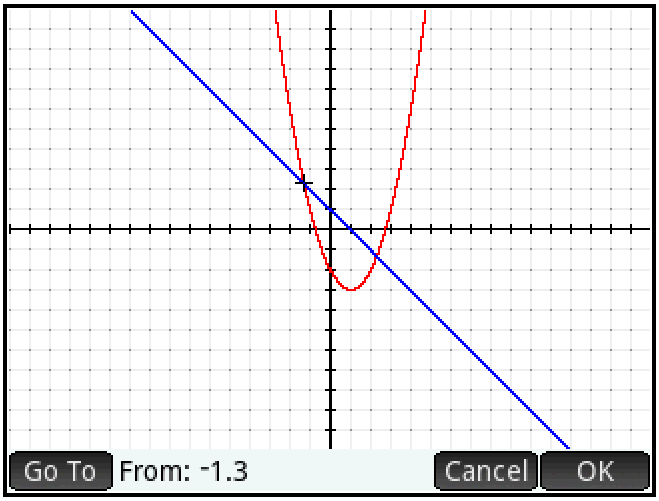
Finding the signed area between two functions

To find the area between the two functions in the range –1.3 ≤ x ≤ 2.3:

1. Tap Fcn and select **Signed area**.

2. Specify the start value for x.

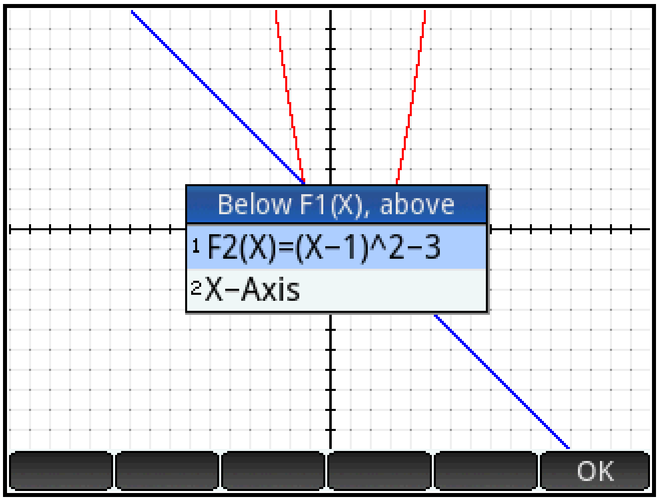
   Tap Go To and press +/– 1 . 3 Enter .



Go To From: -1.3 Cancel OK

3. Tap OK .

4. Select the other function as the boundary for the integral. (If F1(X) is the currently selected function, you would choose F2(X) here, and vice versa.)



Below F1(X), above

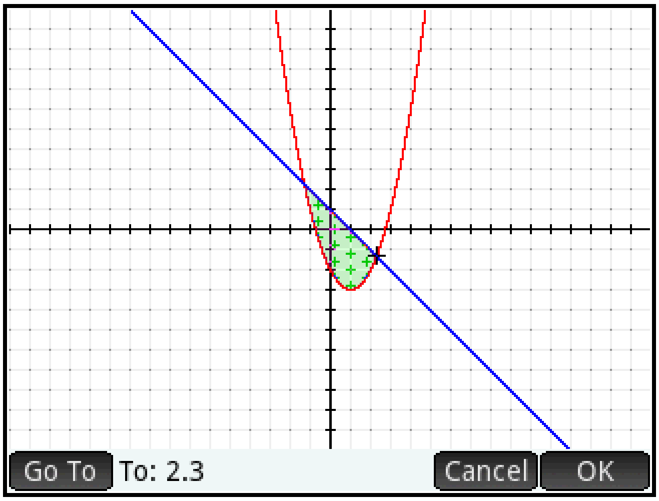
1 F2(X)=(X–1)^2–3

2 X–Axis

5. Specify the end value for x.

   Tap Go To and press 2 . 3 Enter .

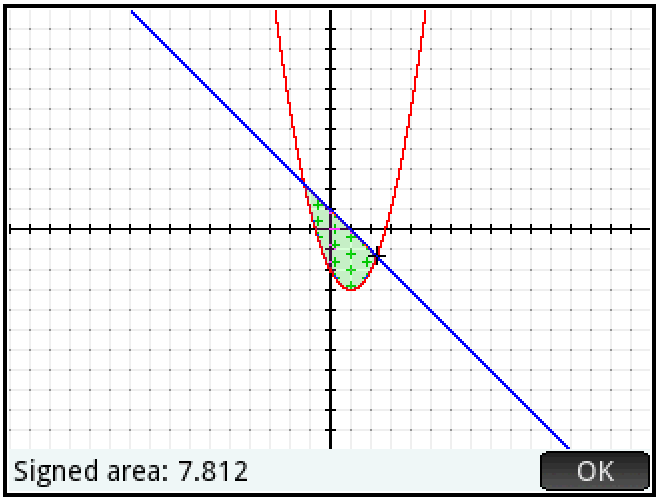
   The cursor jumps to x = 2.3 and the area between the two functions is shaded.



Go To To: 2.3 Cancel OK

6. To display the numerical value of the integral, tap OK .

7. Tap OK to return to the Plot menu. Note that the sign of the area calculated depends both on which function you are tracing and whether you enter the endpoints from left to right or right to left.

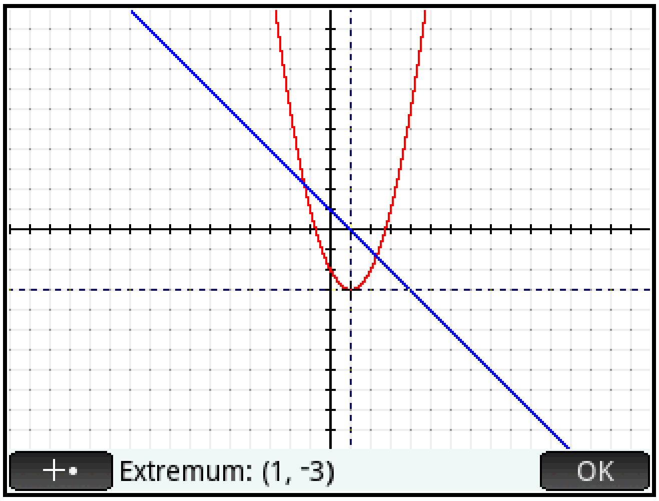


Signed area: 7.812 OK

TIP: When the **Goto** option is available, you can display the **Go To** screen simply by typing a number. The number you type appears on the entry line. Just tap OK to accept it.

Finding the extremum of the quadratic equation

▲ To calculate the coordinates of the extremum of the quadratic equation, move the tracing cursor near the extremum of interest (if necessary), tap Fcn and select **Extremum**.



Extremum: (1, ⁻3)

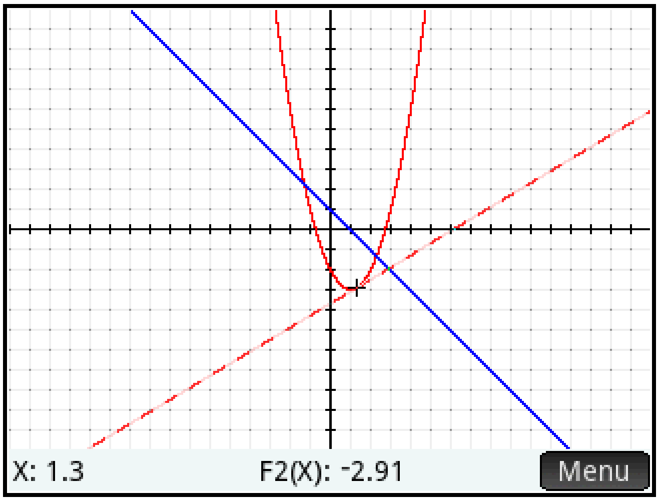
The coordinates of the extremum are displayed at the bottom of the screen.

NOTE: The **ROOT**, **INTERSECTION**, and **EXTREMUM** operations return only one value even if the function in question has more than one root, intersection, or extremum. The app will only return values that are closest to the cursor. You will need to move the cursor closer to other roots, intersections, or extrema if you want the app to calculate values for those.

Adding a tangent to a function

To add a tangent to a function through the trace point:

1. Use ▲ or ▼ to move the tracer to the function.
2. Tap Fcn, and then select **Tangent**. The tangent is drawn as you move the tracer. This option is a toggle; select it again to remove the tangent.



X: 1.3 F2(X): ⁻2.91

Function variables

The result of each numerical analysis in the Function app is assigned to a variable. These variables are named as follows:

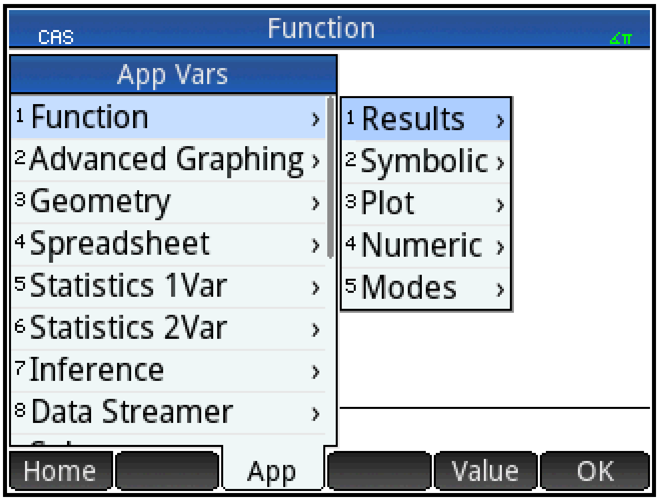
- Root
- Isect (for Intersection)
- Slope
- SignedArea
- Extremum

The result of each new analysis overwrites the previous result. For example, if you find the second root of a quadratic equation after finding the first, the value of Root changes from the first to the second root.

Accessing function variables

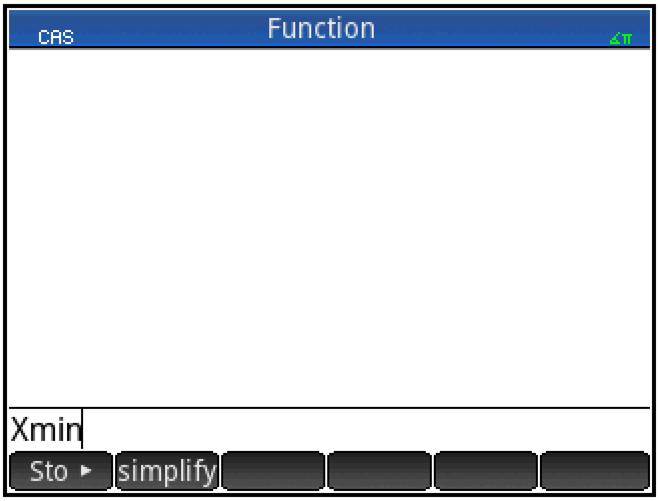
The Function variables are available in Home view and in the CAS, where they can be included as arguments in calculations. They are also available in Symbolic view.

1. To access the variables, press Vars Chars A , tap App , and then select **Function**.
2. Select **Results** and then the variable of interest.



The variable's name is copied to the insertion point and its value is used in evaluating the expression that contains it. You can also enter the value of the variable instead of its name by tapping Value .

For example, in Home view or the CAS you could select **SignedArea** from the **Vars** menus, press × 3 Enter ≈ and get the current value of **SignedArea** multiplied by three.



Xmin

Function variables can also be made part of a function's definition in Symbolic view. For example, you could define a function as $x^2 - x - \text{Root}$.

Summary of FCN operations

Operation	Description
Sketch	Starts Sketch Mode, which enables you to sketch a function with your finger. **NOTE:** This is the same function as Sketch.
Definition	Opens the editor for the selected function definition in Plot view, which enables you to directly edit the function definition or transform the graph.
Transform	Starts Transformation Mode. In Transformation Mode, you can translate the selected function vertically or horizontally, dilate the selected function horizontally, or directly edit the function definition.
Root	Select **Root** to find the root of the current function nearest to the tracing cursor. The cursor is moved to the root value on the x-axis and the resulting x-value is saved in a variable named **Root**. If no root is found, but only an extremum, then the result is labeled **Extremum** instead of **Root**.
Intersection	Select **Intersection** to find the intersection of the graph you are currently tracing and another graph. You need to have at least two selected expressions in Symbolic view. Finds the intersection closest to the tracing cursor. Displays the coordinate values and moves the cursor to the intersection. The resulting x-value is saved in a variable named **Isect**.
Slope	Select **Slope** to enable or disable the ability to display the numeric derivative of the current function at the current position of the tracing cursor. The result is saved in a variable named **Slope**.
Signed area	Select **Signed area** to find the numeric integral. (If there are two or more expressions selected, then you will be asked to choose the second expression from a list that includes the x-axis.) Select a starting point and an ending point. The result is saved in a variable named **SignedArea**.
Extremum	Select **Extremum** to find the maximum or minimum of the current function nearest to the tracing cursor. The cursor moves to the extremum and the coordinate values are displayed. The resulting x-value is saved in a variable named **Extremum**.
Tangent	Select **Tangent** to enable or disable the ability to draw a line tangent to the current function graph through the current position of the tracing cursor.

Defining functions in terms of derivatives or integrals

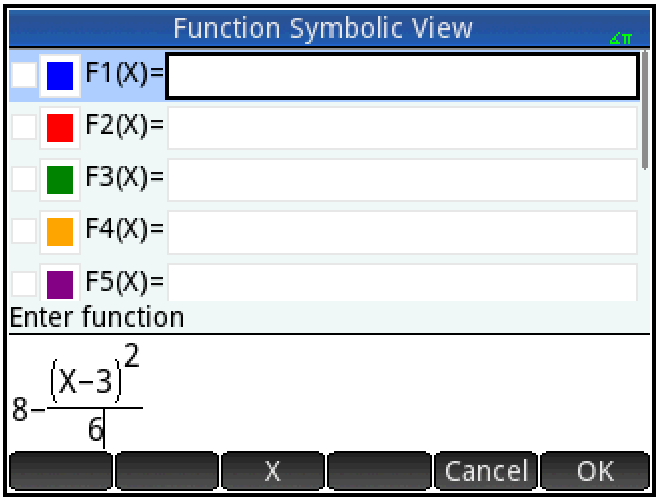
The Function app accepts functions defined in terms of derivatives or integrals. This section describes the methods for each of these cases, with examples.

Functions defined by derivatives

Suppose we wish to plot the graph of the function f(x), defined by $f(x) = \frac{\delta\left(8-(x-3)^2/6\right)}{\delta x}$. We could enter this function directly, but here we define the function $8 - \frac{(x-3)^2}{6}$ as F1(X) and its derivative in F2(X).

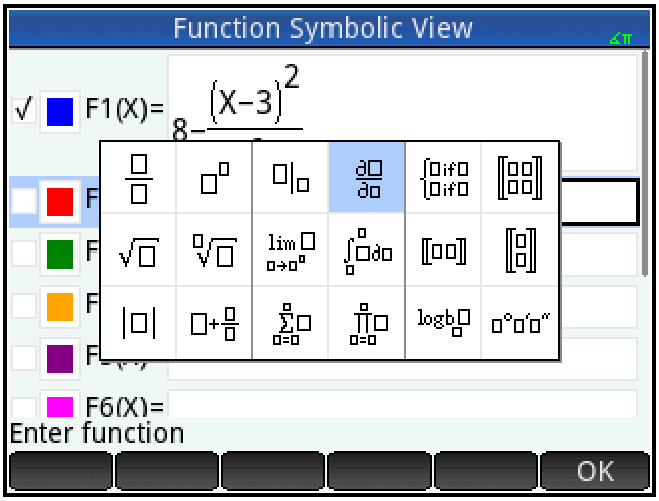
1. Press Symb Setup to go to the Symbolic view.

2. Select the F1(X) field and enter the function as shown in the following figure.



$8-\frac{(X-3)^2}{6}$

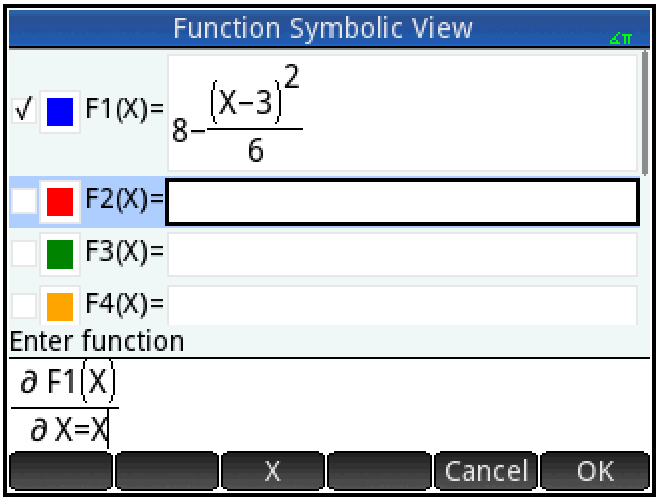
3. Select the F2(X) field, press Units to open the template menu, and then select the derivative template.



F1(X)= $8-\frac{(X-3)^2}{6}$

4. Enter the numerator as F1(X).

5. Outside of the CAS, this template is used to find the derivative of a function at a point. In this case, the denominator is of the form X = a, where a is a real number. In order to indicate our more formal preference here, we enter the denominator as X = X, as shown in the following figure.



Function Symbolic View

√ 🟦 F1(X)= $8-\frac{(X-3)^2}{6}$

□ 🟥 F2(X)=

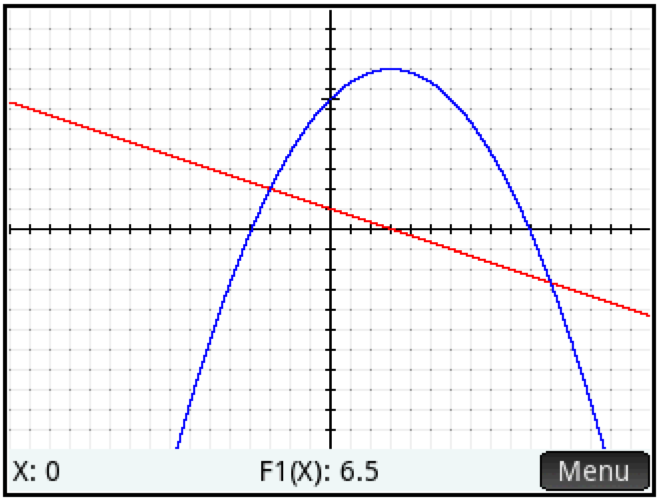
□ 🟩 F3(X)=

□ 🟧 F4(X)=

Enter function

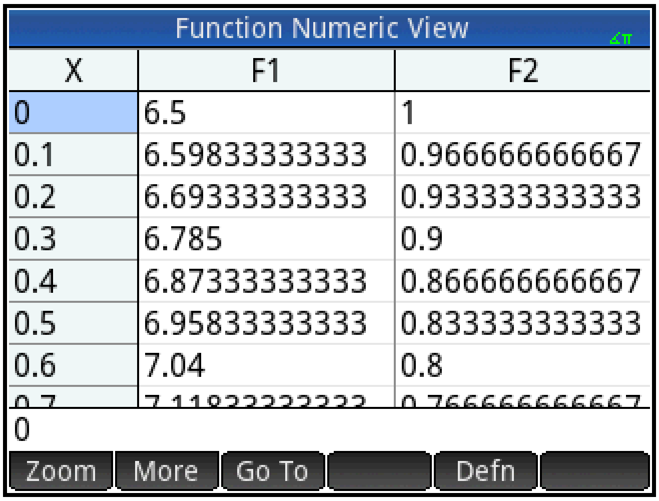
$\frac{\partial F1(X)}{\partial X=X}$

6. Press Plot Setup to see the graphs of both the function (in blue) and its derivative (in red) in the default window.



X: 0 F1(X): 6.5

7. Press Num Setup to see a table of values for both the function and its derivative.



Function Numeric View

X	F1	F2
0	6.5	1
0.1	6.59833333333	0.966666666667
0.2	6.69333333333	0.933333333333
0.3	6.785	0.9
0.4	6.87333333333	0.866666666667
0.5	6.95833333333	0.833333333333
0.6	7.04	0.8
0.7	7.11833333333	0.766666666667

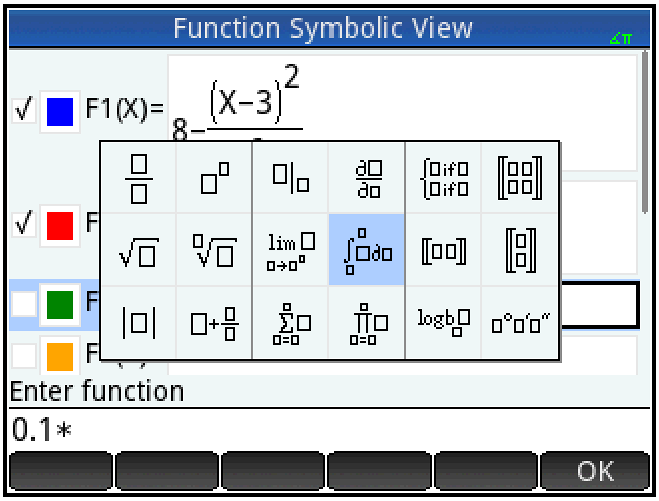
0

Functions defined by integrals

Now, define F3(X) as $0.1\cdot\int_0^X F1(T)\delta T$.

1. Return to the Symbolic view, select F3(X), and enter `0.1` ×.

2. Press Units to open the template menu and select the integral template.



Function Symbolic View

√ ■ F1(X)= $8-\frac{(X-3)^2}{}$

√ ■ F

■ F

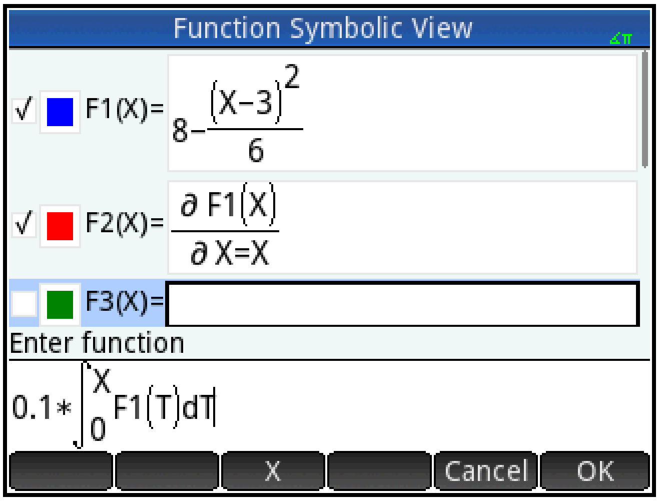
■ F

Enter function

0.1*

3. Enter 0 for the lower limit and X for the upper limit.

4. Enter the rest of your information in the template, as shown in the following figure.



Function Symbolic View

√ 🟦 F1(X)= $8-\frac{(X-3)^2}{6}$

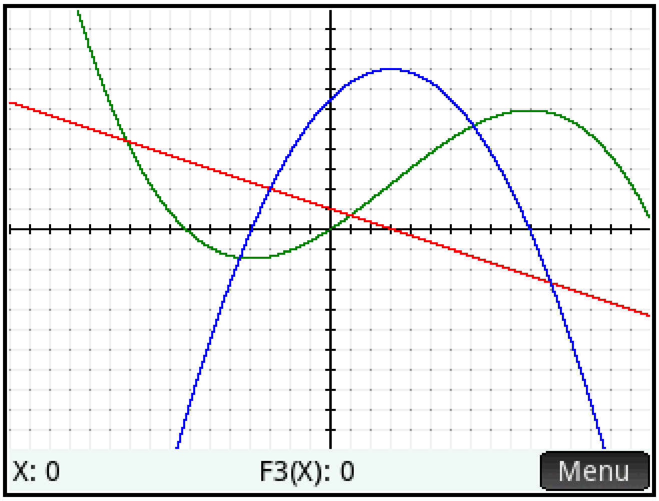
√ 🟥 F2(X)= $\frac{\partial F1(X)}{\partial X=X}$

🟩 F3(X)=

Enter function

$0.1*\int_{0}^{X} F1(T)dT$

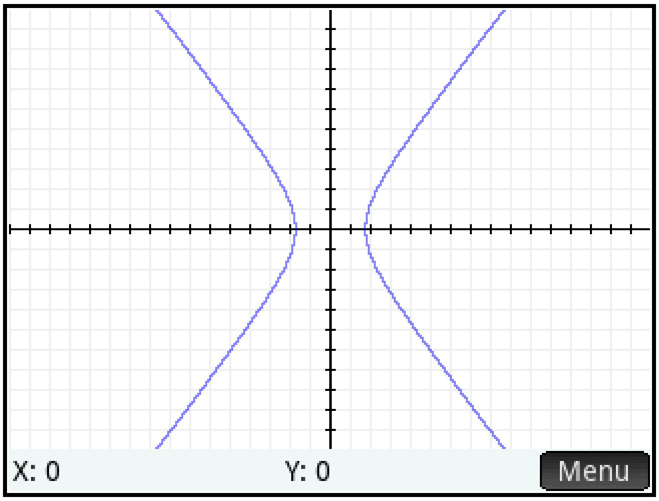
5. Press Plot ↳Setup to see the integral function plotted in green.



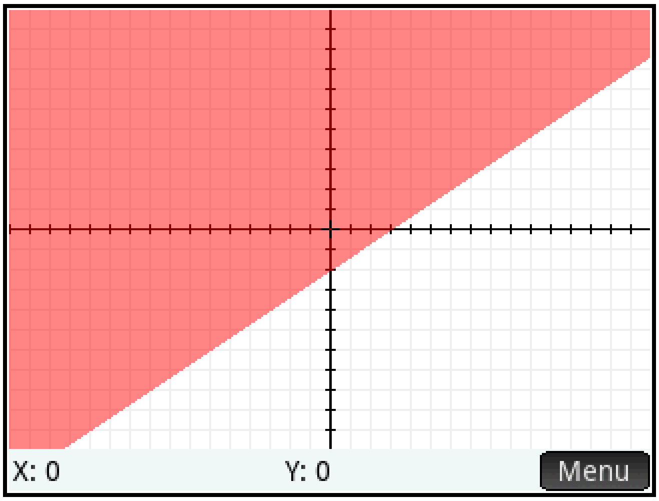
8 Advanced Graphing app

The Advanced Graphing app enables you to define and explore the graphs of symbolic open sentences in x, y, both, or neither. You can plot conic sections, polynomials in standard or general form, inequalities, and functions. The following are examples of the sorts of open sentences you can plot:

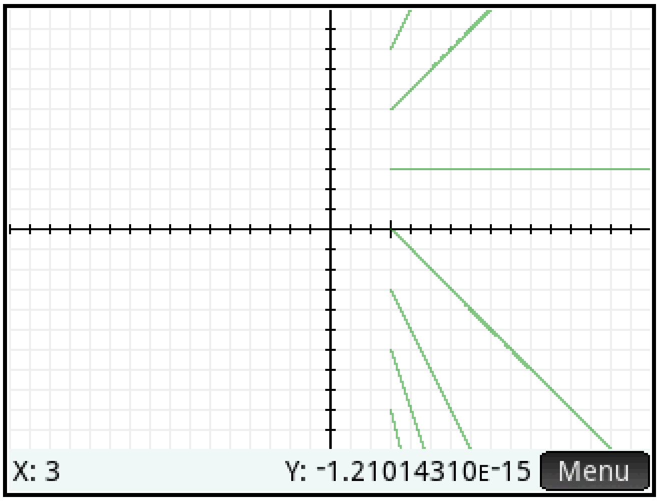
- $x^2/3 - y^2/5 = 1$



- $2x - 3y \leq 6$

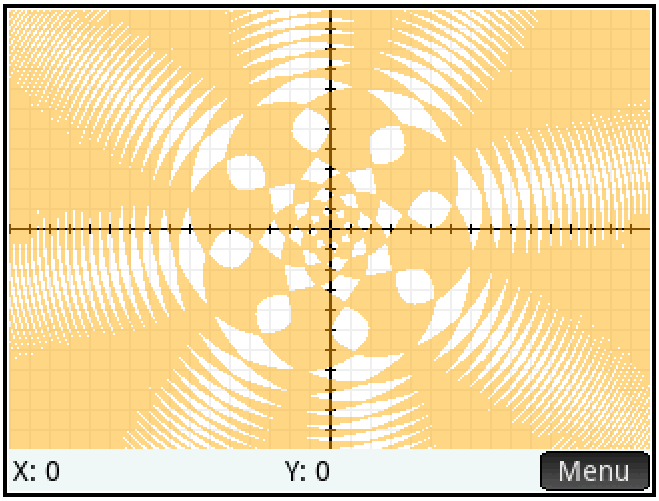


- y mod x = 3



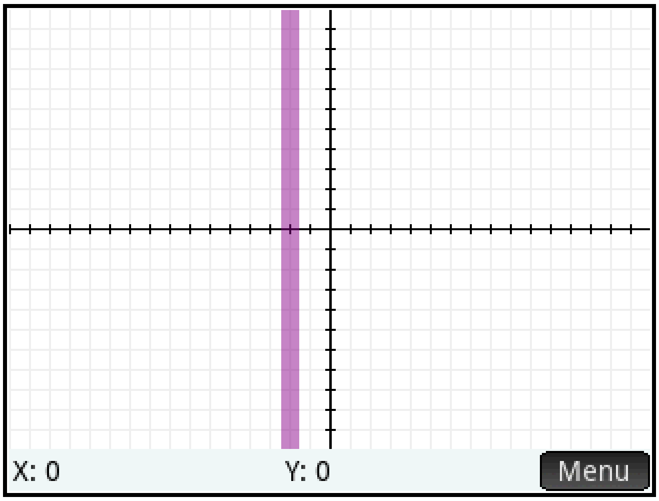
X: 3 Y: -1.21014310E-15

- $\sin((\sqrt{x^2+y^2}-5)^2) > \sin\left(8 \cdot \text{atan}\left(\frac{y}{x}\right)\right)$



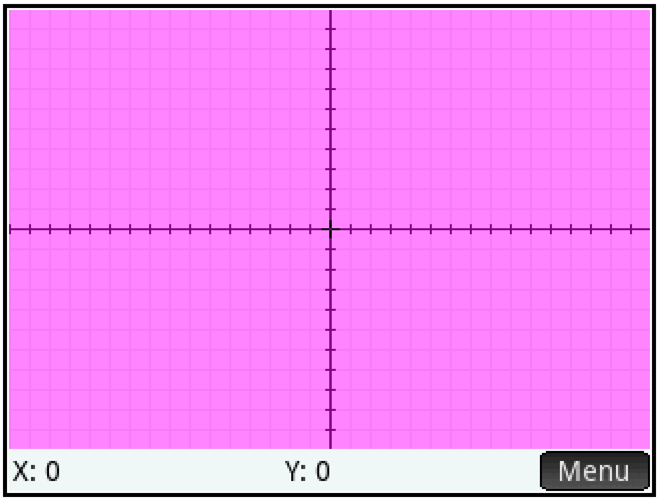
X: 0 Y: 0

- $x^2 + 4x = -4$



X: 0 Y: 0

- 1 > 0



Getting started with the Advanced Graphing app

The Advanced Graphing app uses the customary app views: Symbolic, Plot, and Numeric.

The Symbolic, Plot, and Numeric view buttons are available.

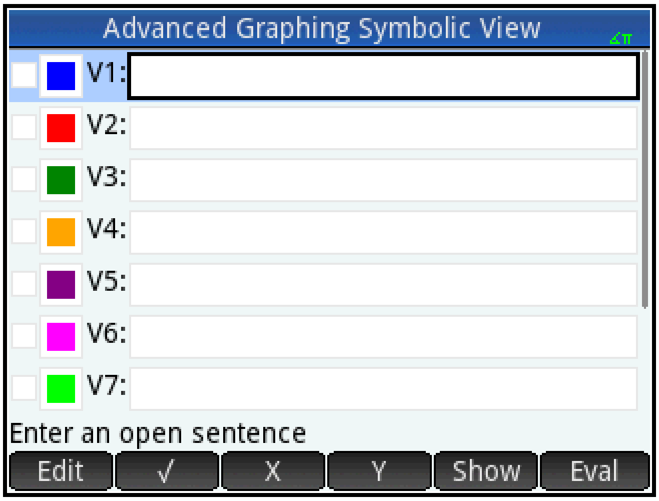
The Trace option in the Advanced Graphing app works differently than in other apps and is described in detail in this chapter.

In this chapter, we will explore the rotated conic defined by the following equation:

$$\frac{x^2}{2}-\frac{7xy}{10}+\frac{3y^2}{4}-\frac{x}{10}+\frac{y}{5}-10<0$$

Opening the Advanced Graphing app

▲ Select Apps Info , and then select **Advanced Graphing**.



Enter an open sentence

The app opens in the Symbolic view.

Defining an open sentence

1. Define the open sentence.

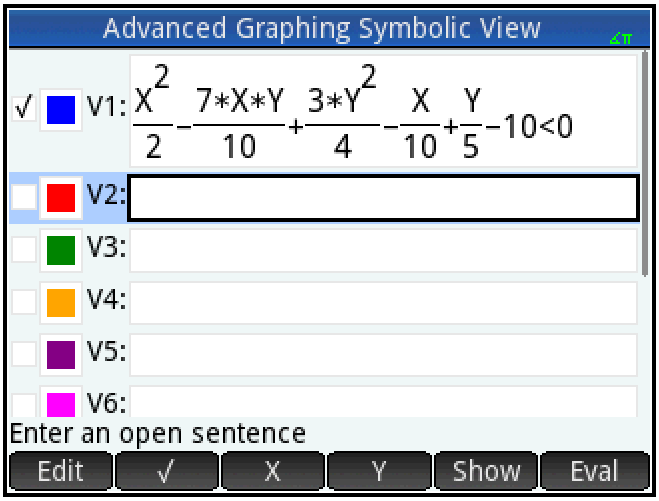
   X x^2 ÷ 2 ▶ − 7 X Y ÷ 10 ▶ + 3 Y

   x^2 ÷ 4 ▶ − X ÷ 10 ▶ + Y ÷ 5 ▶ −

   10 ≤,≥,≠ <0 Enter ≈

NOTE: ≤,≥,≠ displays the relations palette from which relational operators can be easily selected.

This is the same palette that appears if you press Shift 6 ≤,≥,≠ W .



Advanced Graphing Symbolic View

☑ 🟦 V1: $\frac{X^2}{2}-\frac{7*X*Y}{10}+\frac{3*Y^2}{4}-\frac{X}{10}+\frac{Y}{5}-10<0$

□ 🟥 V2:

□ 🟩 V3:

□ 🟧 V4:

□ 🟪 V5:

□ 🟪 V6:

Enter an open sentence

2. Decide if you want to do any of the following:

   - Give an open sentence a custom color when it is plotted
   - Evaluate a dependent function
   - Deselect a definition that you do not want to explore
   - Incorporate variables, math commands and CAS commands in a definition

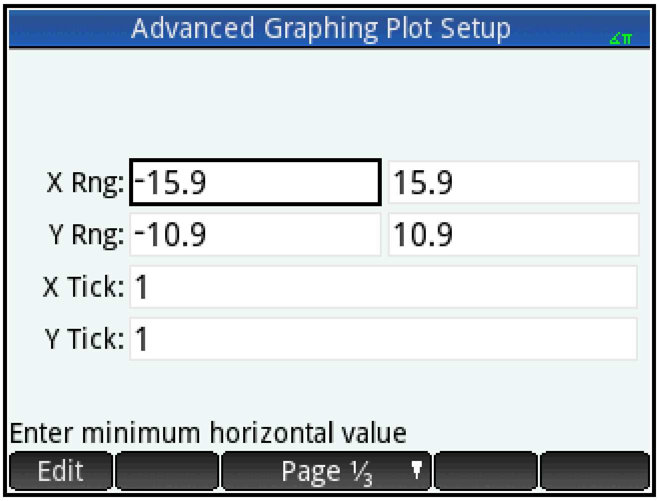
   For the sake of simplicity we can ignore these operations in this example. However, they can be useful and are common Symbolic view operations.

Setting up the plot

You can change the range of the x- and y-axes and the spacing of the interval marks along the axes.

▲ Display Plot Setup view.

Shift Plot ↳Setup



Advanced Graphing Plot Setup

X Rng: -15.9 15.9

Y Rng: -10.9 10.9

X Tick: 1

Y Tick: 1

Enter minimum horizontal value

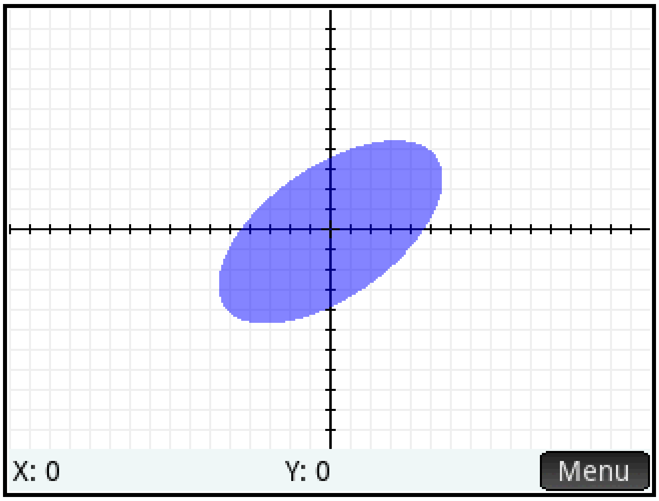
For this example, you can leave the plot settings at their default values. If your settings do not match those in the previous figure, press Shift Esc Clear to restore the default values.

Common Plot view operations can be used to change the appearance of plots.

Plotting the selected definitions

▲ Plot the selected definitions.

Plot ↳Setup



Exploring the graph

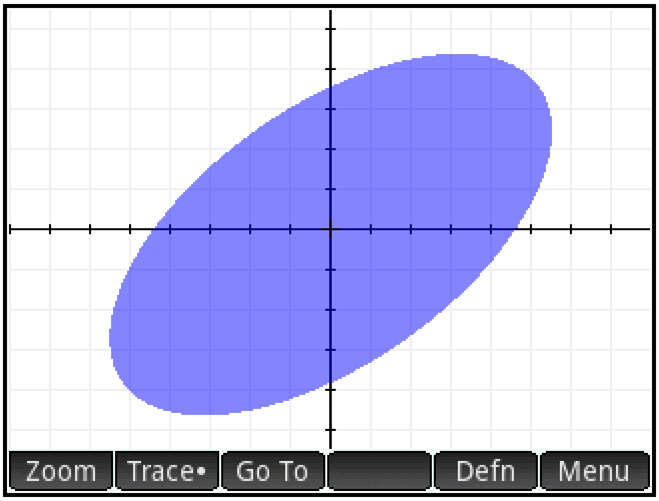
1. Tap Menu to display the Plot view menu items.

   Note that you have options to zoom, trace, go to a specified point, and display the definition of the selected graph.

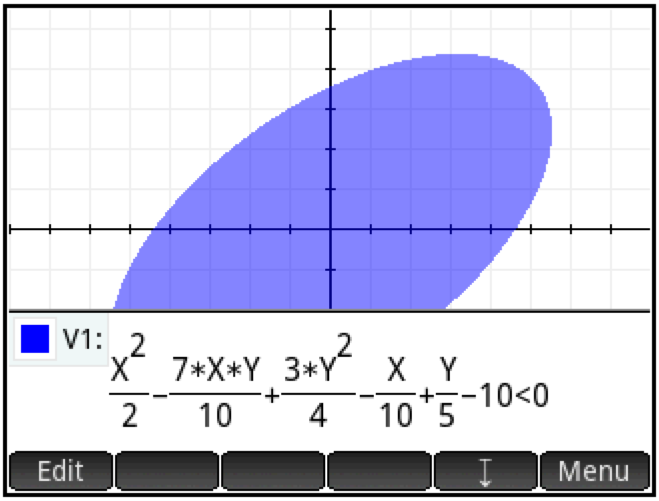
   You can use the zoom and split-screen functions. You can scroll the Plot view, or use a 2-finger pinch zoom to zoom both in and out. A horizontal pinch zooms the x-axis only; a vertical pinch zooms the y-axis only; a diagonal pinch zooms both simultaneously. You can also zoom in or out on the cursor position by pressing + Ans or − Base, respectively.

2. Tap Zoom and select **In**.

   A special feature of the Advanced Graphing app enables you to edit the definition of a graph from within the Plot view.



3. Tap Defn. The definition as you entered it in Symbolic view appears at the bottom of the screen.



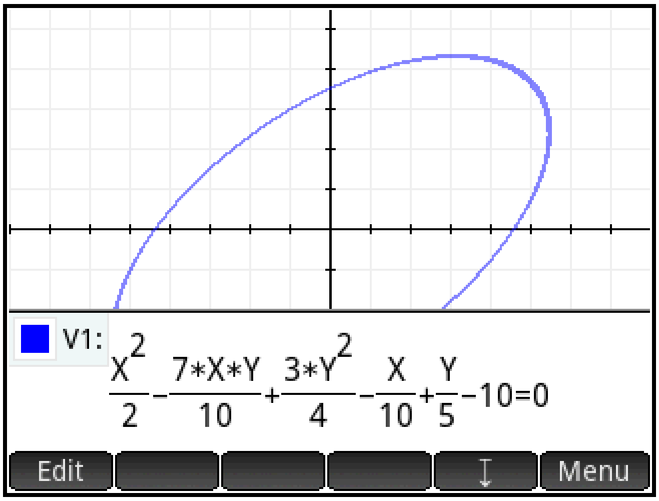
V1: $\frac{X^2}{2}-\frac{7*X*Y}{10}+\frac{3*Y^2}{4}-\frac{X}{10}+\frac{Y}{5}-10<0$

4. Tap Edit.

   The definition can now be edited.

5. Change the < to = and tap OK.

   Notice that the graph changes to match the new definition. The definition in Symbolic view also changes.



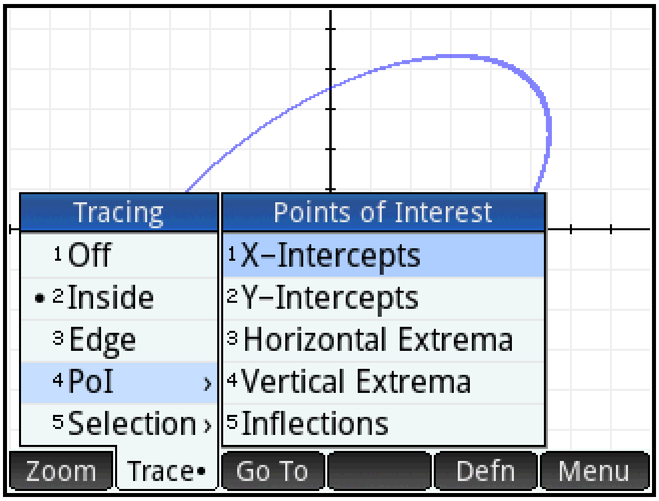
V1: $\frac{X^2}{2}-\frac{7*X*Y}{10}+\frac{3*Y^2}{4}-\frac{X}{10}+\frac{Y}{5}-10=0$

6. Tap ↧ to drop the definition to the bottom of the screen so that you can see the full graph. The definition is converted from textbook mode to algebraic mode to save screen space.

Trace in Plot view

In most HP apps, the Plot view contains Trace•, a toggle to turn tracing a function on and off. In the Advanced Graphing app, the relations plotted in Plot view may or may not be functions. So, instead of a toggle, Trace• becomes a menu for selecting how the tracer will behave. The Trace menu contains the following options:

- Off
- Inside
- PoI (Points of Interest)
  - X-Intercepts
  - Y-Intercepts
  - Horizontal Extrema
  - Vertical Extrema
  - Inflections



- Selection

The tracer does not extend beyond the current Plot view window. The following table contains brief descriptions of each option.

Option	Description
Off	Turns tracing off so that you can move the cursor freely in Plot view.
Inside	Constrains the tracer to move within a region where the current relation is true. You can move in any direction within the region. Use this option for inequalities, for example.
Edge	Constrains the tracer to move along an edge of the current relation, if one can be found. Use this option for functions as well as for inequalities, and so on.
PoI > X-Intercepts	Jumps from one x-intercept to another on the current graph.
PoI > Y-Intercepts	Jumps from one y-intercept to another on the current graph.
PoI > Horizontal Extrema	Jumps between the horizontal extrema on the current graph.
PoI > Vertical Extrema	Jumps between the vertical extrema on the current graph.

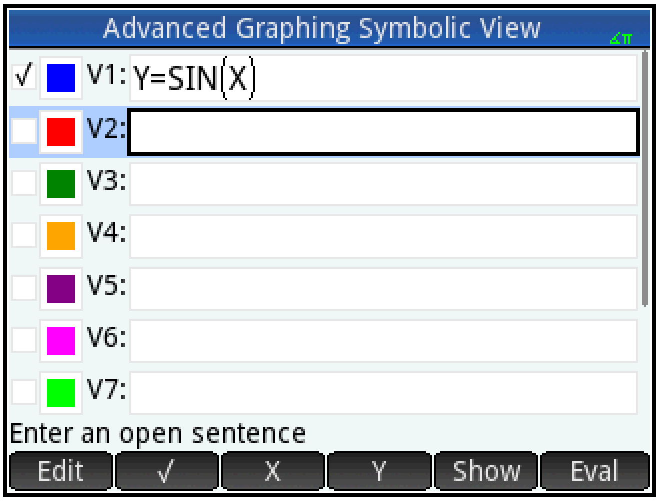
Option	Description
PoI > Inflections	Jumps from one inflection point to another on the current graph.
Selection	Opens a menu so you can select which relation to trace. This option is needed because ▲ and ▼ no longer jump from relation to relation for tracing. All four cursor keys are needed for moving the tracer in the Advanced Graphing app.

Numeric view

The Numeric view of most HP apps is designed to explore 2-variable relations using numerical tables. Because the Advanced Graphing app expands this design to relations that are not necessarily functions, the Numeric view of this app becomes significantly different, though its purpose is still the same. The unique features of the Numeric view are illustrated in the following sections.

▲ Press Symb ↳Setup to return to Symbolic view and define V1 as Y=SIN(X).

NOTE: You do not have to erase the previous definition first. Just enter the new definition and tap OK.



Advanced Graphing Symbolic View

√ 🟦 V1: Y=SIN(X)

□ 🟥 V2:

□ 🟩 V3:

□ 🟧 V4:

□ 🟪 V5:

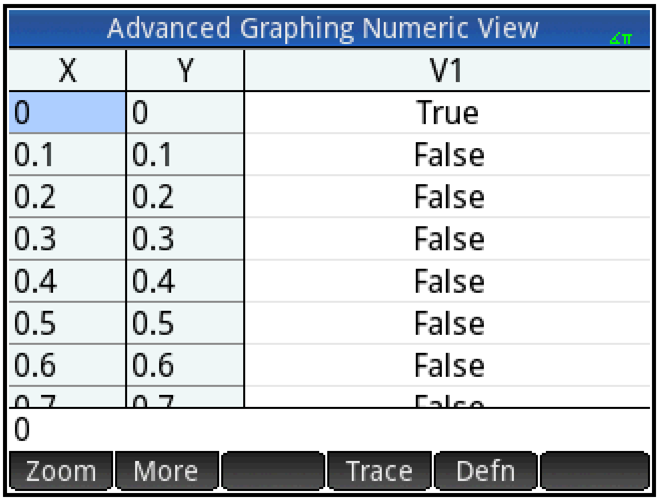
□ 🟪 V6:

□ 🟩 V7:

Enter an open sentence

Displaying Numeric view

▲ Press Num ↳Setup to display Numeric view.


Advanced Graphing Numeric View

X	Y	V1
0	0	True
0.1	0.1	False
0.2	0.2	False
0.3	0.3	False
0.4	0.4	False
0.5	0.5	False
0.6	0.6	False
0.7	0.7	False

0

By default, the Numeric view displays rows of x- and y-values. In each row, the 2 values are followed by a column that tells whether or not the x–y pair satisfies each open sentence (True or False).

Exploring Numeric view

- With the cursor in the X column, type a new value and tap OK .The table scrolls to the value you entered.

You can also enter a value in the Y column and tap OK . Press ▶ and ◀ to move between the columns in Numeric view.

You can customize the values shown in the table, using the same options available for customizing the tracer in Plot view. For example, you can display only the x-intercepts or inflection points. The values displayed correspond to the points of interest visible in Plot view.

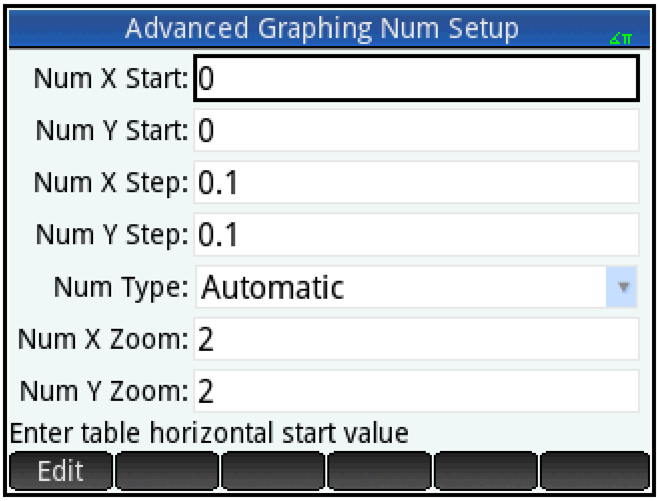
You can also zoom in or out on the X-variable or Y-variable using the options available in the Zoom menu. Note that in Numeric view, zooming decreases or increases the increment between consecutive x- and y-values. Zooming in decreases the increment; zooming out increases the increment. This and other options are common Numeric view operations.

Numeric Setup view

Although you can configure the X- and Y-values shown in Numeric view by entering values and zooming in or out, you can also directly set the values shown using Numeric setup.

- Display the Numeric Setup view.

Shift Num Setup



Advanced Graphing Num Setup

Num X Start: 0

Num Y Start: 0

Num X Step: 0.1

Num Y Step: 0.1

Num Type: Automatic

Num X Zoom: 2

Num Y Zoom: 2

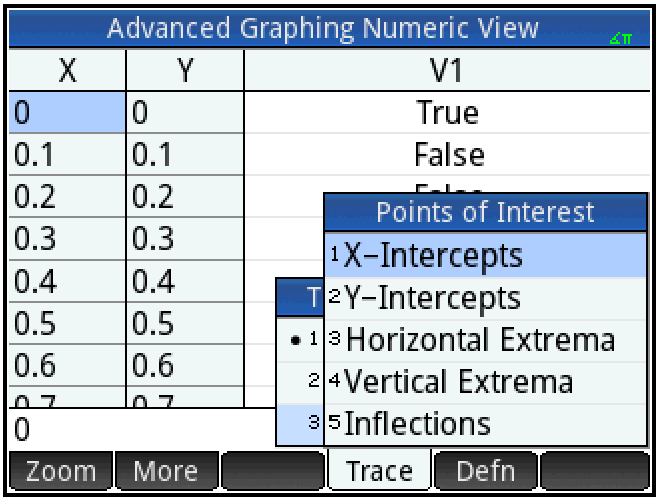
Enter table horizontal start value

You can set the starting value and step value (that is, the increment) for both the X-column and the Y-column, as well as the zoom factor for zooming in or out on a row of the table. You can also choose whether the table of data in Numeric view is automatically populated or whether it is populated by you typing in the particular x-values and y-values you are interested in. These options—**Automatic** or **BuildYourOwn**—are available from the **Num Type** list. These are custom table options.

Trace in Numeric view

Besides the default configuration of the table in Numeric view, there are other options available in the Trace menu. The trace options in Numeric view mirror the trace options in Plot view. Both are designed to help you investigate the properties of relations numerically using a tabular format. Specifically, the table can be configured to show any of the following:

- Edge values (controlled by X or Y)
- PoI (Points of Interest)
  - X-Intercepts
  - Y-Intercepts
  - Horizontal Extrema
  - Vertical Extrema
  - Inflections


Advanced Graphing Numeric View

X	Y	V1
0	0	True
0.1	0.1	False
0.2	0.2	False
0.3	0.3	
0.4	0.4	
0.5	0.5	
0.6	0.6	
0.7	0.7	

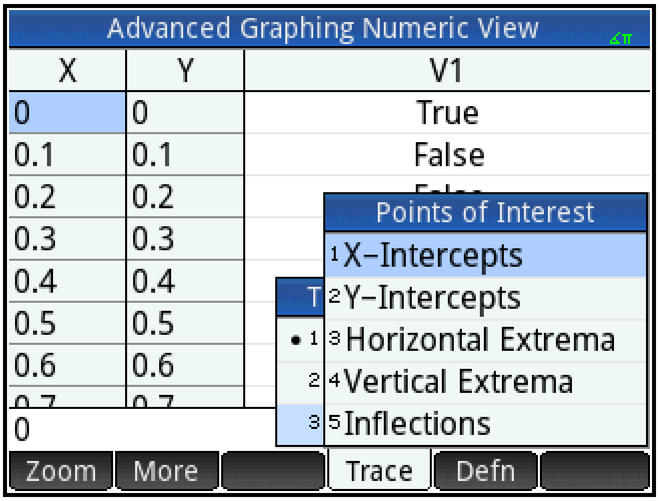
0

The values shown using the Trace options depend on the Plot view window; that is, the values shown in the table are restricted to points visible in Plot view. Zoom in or out in Plot view to get the values you want to see in the table in Numeric view.

Edge

1. Tap Trace• and select **Edge**.

   Now the table shows (if possible) pairs of values that make the relation true. By default, the first column is the Y-column and there are multiple X-columns in case more than one X-value can be paired with the Y-value to make the relation true. Tap X to make the first column an X-column followed by a set of Y-columns. In the following figure, for Y = 0, there are 10 values of X in the default Plot view that make the relation Y = SIN(X) true. These are shown in the first row of the table. It can be clearly seen that the sequence of X-values have a common difference of π.


Advanced Graphing Numeric View

X	Y	V1
0	0	True
0.1	0.1	False
0.2	0.2	False
0.3	0.3	
0.4	0.4	
0.5	0.5	
0.6	0.6	
0.7	0.7	

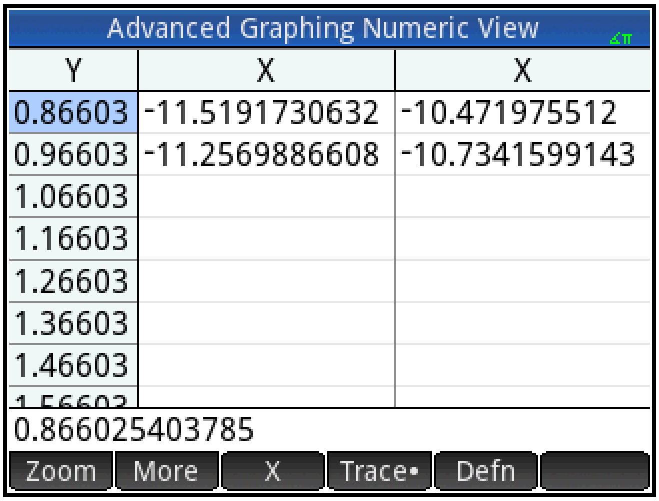
0

   Again, you can enter a value for Y that is of interest.

2. With **0** highlighted in the Y-column, enter $\frac{\sqrt{3}}{2}$.

   Shift x^2 3 ÷ 2 Enter

3. Tap Column and select 4.



Y	X	X
0.86603	-11.5191730632	-10.471975512
0.96603	-11.2569886608	-10.7341599143
1.06603		
1.16603		
1.26603		
1.36603		
1.46603		
1.56603		

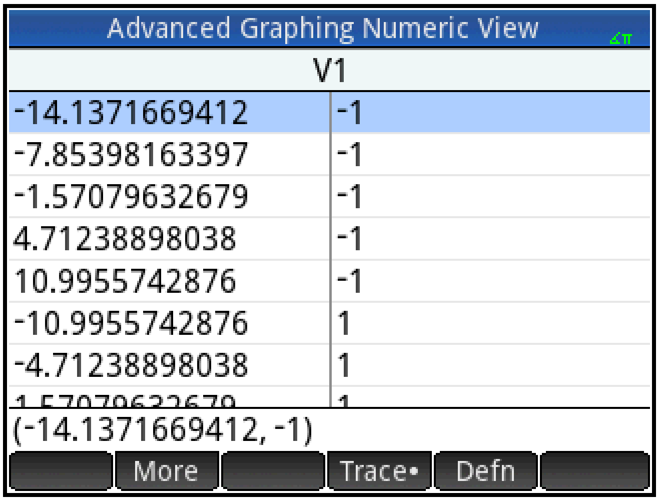
0.866025403785

The first row of the table now illustrates that there are two branches of solutions. In each branch, the consecutive solution values are 2π apart.

PoI

1. Tap Trace•, select PoI and select Vertical Extrema to see the extrema listed in the table.

2. Tap Column and select 2 to see just two columns.



V1	
-14.1371669412	-1
-7.85398163397	-1
-1.57079632679	-1
4.71238898038	-1
10.9955742876	-1
-10.9955742876	1
-4.71238898038	1
1.57079632679	1

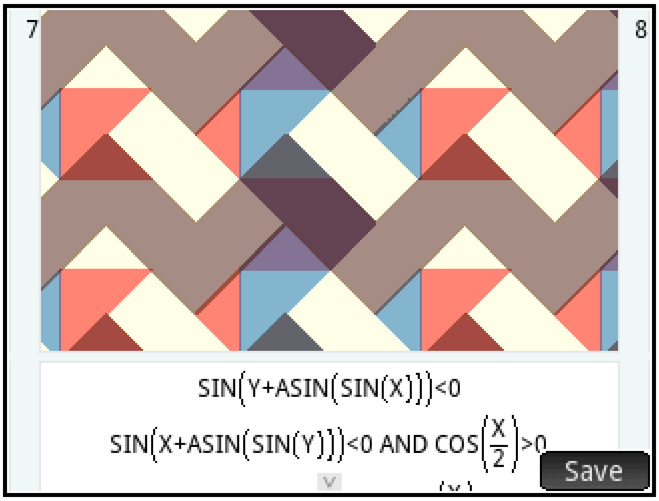
(-14.1371669412, -1)

The table lists the 5 minima visible in Plot view, followed by the 5 maxima.

Plot Gallery

A gallery of interesting graphs—and the equations that generated them—is provided with the calculator. You open the gallery from Plot view:

1. With Plot view open, press the **Menu** key. Note that you press the Menu key here, not the Menu touch button on the screen.
2. From the menu, select **Visit Plot Gallery**. The first graph in the Gallery appears, along with its equation.



SIN(Y+ASIN(SIN(X)))<0

SIN(X+ASIN(SIN(Y)))<0 AND COS$\left(\frac{X}{2}\right)$>0

3. Press ▶ to display the next graph in the Gallery, and continue likewise until you want to close the Gallery.
4. To close the Gallery and return to Plot view, press Plot ↳Setup .

Exploring a plot from the Plot Gallery

If a particular plot in the Plot Gallery interests you, you can save a copy of it. The copy is saved as a new app—a customized instance of the Advanced Graphing app. You can modify and explore the app as you would with built-in version of the Advanced Graphing app.

To save a plot from the Plot Gallery:

1. With the plot of interest displayed, tap Save .
2. Enter a name for your new app and tap OK .
3. Tap OK again. Your new app opens, with the equations that generated the plot displayed in Symbolic view. The app is also added to the Application Library so that you can return to it later.

9 Geometry

The Geometry app enables you to draw and explore geometric constructions. A geometric construction can be composed of any number of geometric objects, such as points, lines, polygons, curves, tangents, and so on. You can take measurements (such as areas and distances), manipulate objects, and note how measurements change.

There are five app views:

- Plot view: provides drawing tools for you to construct geometric objects
- Symbolic view: provides editable definitions of the objects in Plot view
- Numeric view: for making calculations about the objects in Plot view
- Plot Setup view: for customizing the appearance of Plot view
- Symbolic Setup view: for overriding certain system-wide settings

There is no Numeric Setup view in this app.

To open the Geometry app, press Apps Info and select **Geometry**. The app opens in Plot view.

Getting started with the Geometry app

The following example shows how you can graphically represent the derivative of a curve, and have the value of the derivative automatically update as you move a point of tangency along the curve. The curve to be explored is y = 3sin(x).

Since the accuracy of our calculation in this example is not too important, we will first change the number format to fixed at 3 decimal places. This will also help keep our geometry workspace uncluttered.

Preparation

1. Press Shift CAS Settings .
2. On the first **CAS settings** page, set the number format to **Standard** and the number of decimal places to **4**.

Opening the app and plotting the graph

1. Press Apps Info and select **Geometry**.

   If there are objects showing that you don’t need, press Shift Esc Clear and confirm your intention by tapping OK.

   The app opens in Plot view. This view displays a Cartesian plane with a menu bar at the bottom. Next to the menu bar, this view displays the coordinates of the cursor. After you interact with the app, the bottom of the display displays the currently active tool or command, help for the current tool or command, and a list of all objects recognized as being under the current pointer location.

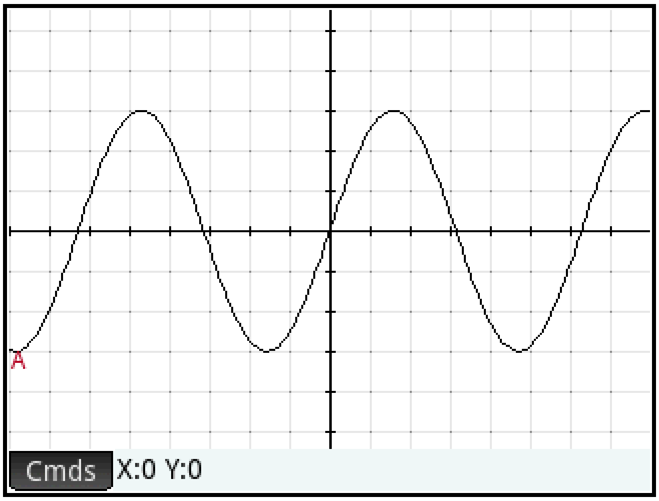
2. Select the type of graph you want to plot. In this example we are plotting a simple sinusoidal function, so choose:

   Cmds > **Plot** > **Function**

3. With `plotfunc(` on the entry line, enter 3*sin(x):

   3 × SIN ALPHA Shift × Enter

   Note that x must be entered in lowercase in the Geometry app.



If your graph doesn't resemble the previous figure, adjust the **X Rng** and **Y Rng** values in Plot Setup view (Shift Plot).

We'll now add a point to the curve, a point that will be constrained always to follow the contour of the curve.

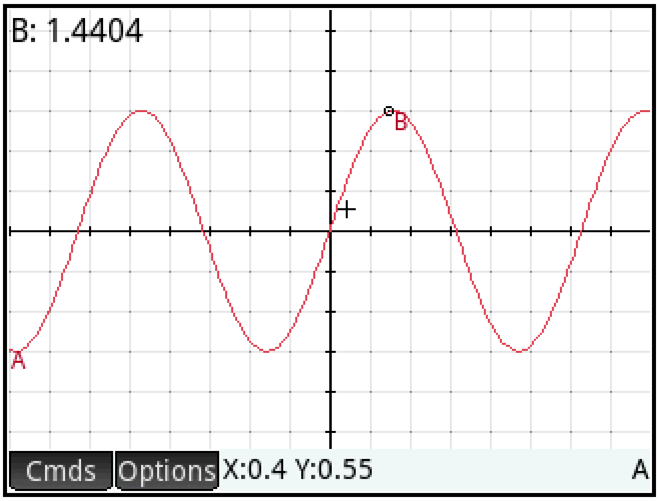
Adding a constrained point

1. Tap Cmds, tap **Point**, and then select **Point On**.

   Choosing **Point On** rather than **Point** means that the point will be constrained to whatever it is placed on.

2. Tap anywhere on the graph, press Enter and then press Esc.

Notice that a point is added to the graph and given a name (**B** in this example). Tap a blank area of the screen to deselect everything. (Objects colored light blue are selected.)


B: 1.4404

X:0.4 Y:0.55

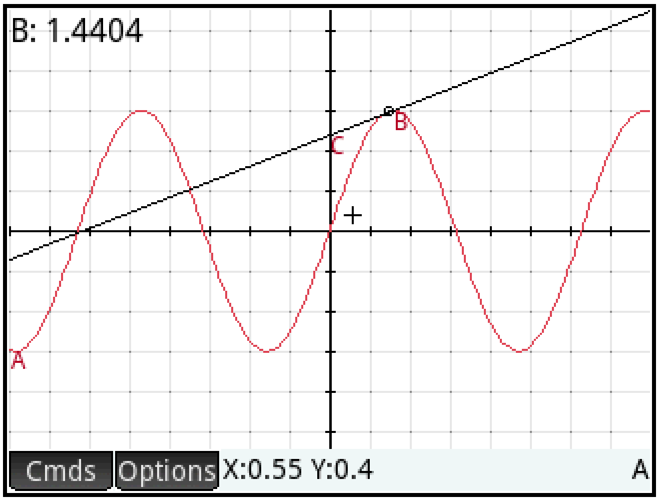
Adding a tangent

1. We will now add a tangent to the curve, making point **B** the point of tangency:

   Cmds **> Line > Tangent**

2. When prompted to select a curve, tap anywhere on the curve and press Enter. When prompted to select a point, tap point **B** and press Enter to see the tangent. Press Esc to close the Tangent tool.

   Depending on where you placed point **B**, your graph might be different from the following figure. Now, make the tangent stand out by giving it a bright color.


B: 1.4404

X:0.55 Y:0.4

3. Tap on the tangent to select it. After the tangent is selected, the new menu key Options appears. Tap Options or press Menu Paste, and then select **Choose color**.

4. Pick a color, and then tap on a blank area of the screen to see the new color of the tangent line.

5. Tap point **B** and drag it along the curve; the tangent moves accordingly. You can also drag the tangent line itself.

6. Tap point **B** and then press Enter ≈ to select the point. The point turns light blue to show that it has been selected. Now, you can either drag the point with your finger or use the cursor keys for finer control of the movement of point **B**. To deselect point **B**, either press Esc Clear or tap point **B** and press Enter ≈.

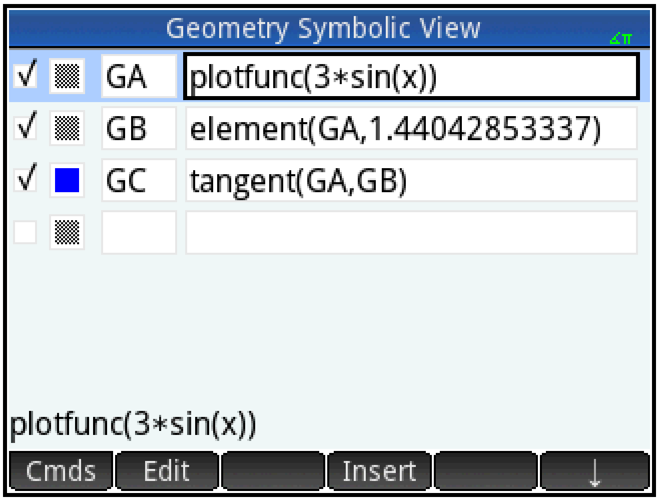
Note that whatever you do, point **B** remains constrained to the curve. Moreover, as you move point **B**, the tangent moves as well. If it moves off the screen, you can bring it back by dragging your finger across the screen in the appropriate direction.

Creating a derivative point

The derivative of a graph at any point is the slope of its tangent at that point. We'll now create a new point that will be constrained to point **B** and whose ordinate value is the derivative of the graph at point **B**. We'll constrain it by forcing its *x* coordinate (that is, its abscissa) to always match that of point **B**, and its *y* coordinate (that is, its ordinate) to always equal the slope of the tangent at that point.

1. To define a point in terms of the attributes of other geometric objects, press Symb Setup to go to Symbolic view.

   Note that each object you have so far created is listed in Symbolic view. Note too that the name for an object in Symbolic view is the name it was given in Plot view but prefixed with a "G". Thus the graph—labeled **A** in Plot view—is labeled **GA** in Symbolic view.



Geometry Symbolic View

✓	▨	GA	plotfunc(3*sin(x))
✓	▨	GB	element(GA,1.44042853337)
✓	■	GC	tangent(GA,GB)
□	▨		

plotfunc(3*sin(x))

2. Highlight the blank definition following **GC** and tap New.

   When creating objects that are dependent on other objects, the order in which they appear in Symbolic view is important. Objects are drawn in Plot view in the order in which they appear in Symbolic view. Since we are about to create a new point that is dependent on the attributes of **GB** and **GC**, it is important that we place its definition after that of both **GB** and **GC**. That is why we made sure we were at the bottom the list of definitions before tapping New. If the new definition appeared higher up in Symbolic view, the point created in the following step would not be active in Plot view.

3. Tap Cmds and choose **Point > point**.

   You now need to specify the *x* and *y* coordinates of the new point. The former is defined as the abscissa of point **B** (referred to as **GB** in Symbolic view) and the latter is defined as the slope of tangent line **C** (referred to as **GC** in Symbolic view).

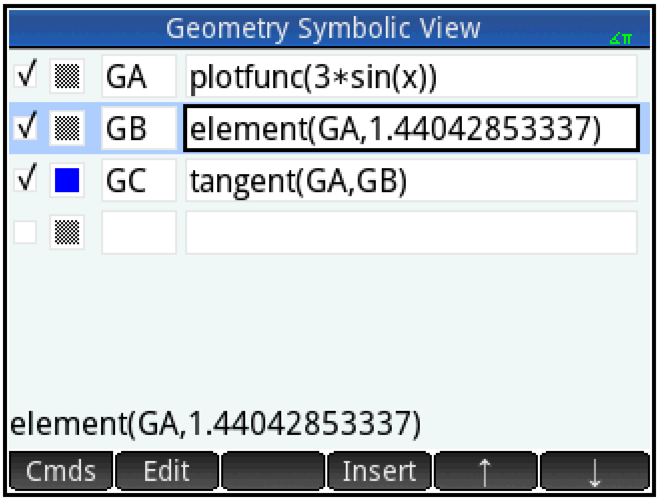
4. You should have `point()` on the entry line. Between the parentheses, add:

   ```
   abscissa(GB),slope(GC)
   ```

   For the abscissa command, press Mem B and tap Catlg. Press Vars Chars A to jump to commands that start with the letter A and then scroll to **abscissa** and tap OK. For the slope command, press Mem B and tap Catlg. Press 9 to jump to commands that start with the letter S and then scroll to **slope** and tap OK. Of course, you can type the commands in letter by letter as well. Press ALPHA alpha Shift ALPHA alpha for the lower-case alpha lock. Press ALPHA alpha again to unlock.

5. Tap OK .

The definition of your new point is added to Symbolic view. When you return to Plot view, you will see a point named **D** and it will have the same *x*-coordinate as point **B**.



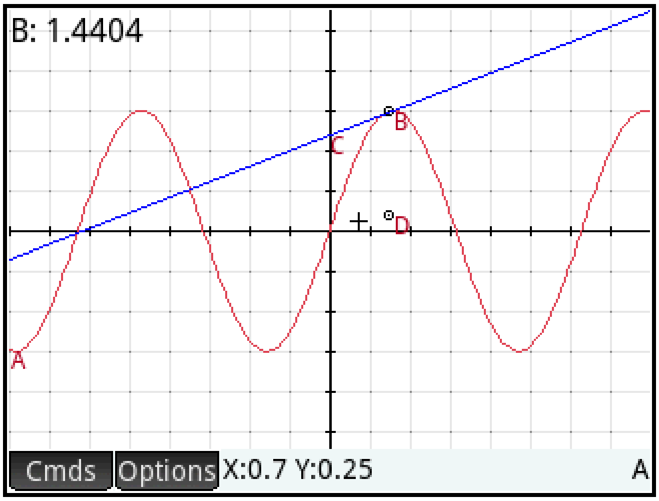
Geometry Symbolic View

√	▩	GA	plotfunc(3∗sin(x))
√	▩	GB	element(GA,1.44042853337)
√	■	GC	tangent(GA,GB)
□	▩		

element(GA,1.44042853337)

6. Press Plot ↳Setup .

If you can’t see point **D**, pan until it comes into view. The *y* coordinate of **D** will be the derivative of the curve at point **B**.



B: 1.4404

X:0.7 Y:0.25

Since it is difficult to read coordinates off the screen, we’ll add a calculation that will give the exact derivative (to three decimal places) and which we can display in Plot view.

Adding some calculations

1. Press Num ↳Setup .

   Numeric view is where you enter calculations.

2. Tap New .

3. Tap Cmds and choose **Measure > slope**.

4. Between parentheses, add the name of the tangent, namely `GC`, and tap OK.

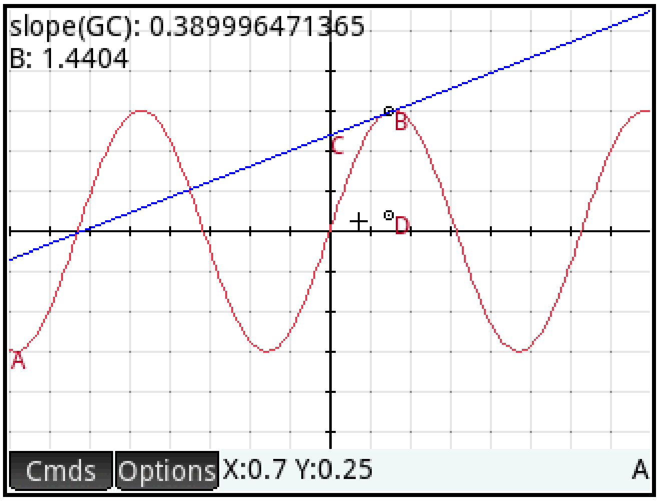
   Notice that the current slope is calculated and displayed. The value here is dynamic, that is, if the slope of the tangent changes in Plot view, the value of the slope is automatically updated in Numeric view.

5. With the new calculation highlighted in Numeric view, tap √.

   Selecting a calculation in Numeric view means that it will also be displayed in Plot view.

6. Press Plot ↳Setup to return to Plot view.

   Notice the calculation that you have just created in Numeric view is displayed at the top left of the screen.



slope(GC): 0.389996471365
B: 1.4404

Cmds Options X:0.7 Y:0.25 A

   Let's now add two more calculations to Numeric view and have them displayed in Plot view.

7. Press Num ↳Setup to return to Numeric view.

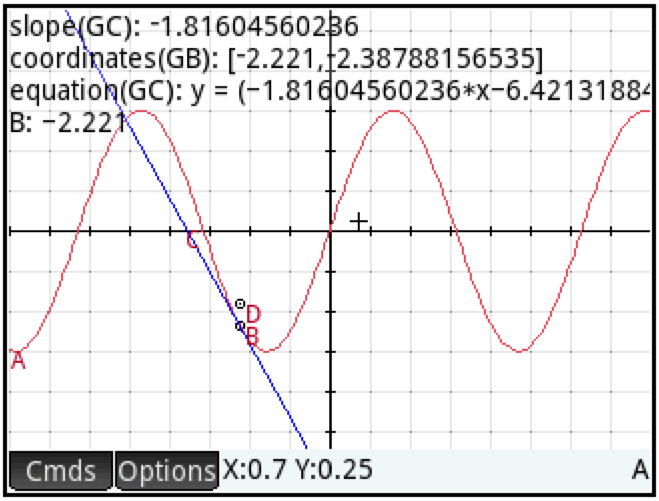
8. Tap the last blank field to select it, and then tap New to start a new calculation. Tap Cmds, select **Cartesian**, and then select **Coordinates**. Between the parentheses, enter `GB` and then tap OK.

9. To start a third calculation, tap Cmds, select **Cartesian**, and then select **Equation of**. Between the parentheses, enter `GC` and then tap OK.

10. Make sure both of these new equations are selected (by choosing each one and pressing √).

11. Press Plot ↳Setup to return to Plot view.

Notice that your new calculations are displayed.



slope(GC): -1.81604560236
coordinates(GB): [-2.221,-2.38788156535]
equation(GC): y = (−1.81604560236*x−6.42131884
B: −2.221

12. Tap point **B** and then press Enter ≈ to select it.

13. Use the cursor keys to move point **B** along the graph. Note that with each move, the results of the calculations shown at the top left of the screen change. To deselect point **B**, tap point **B** and then press Enter ≈.

Calculations in Plot view

By default, calculations in Plot view are docked to the upper left of the screen. You can drag a calculation from its dock and position it anywhere you like; however, after being undocked, the calculation scrolls with the display. Tap and hold a calculation to edit its label. An edit line opens so that you can enter your own label. You can also tap Choose and select a different color for the calculation and its label. Tap OK when you are done.

Trace the derivative

Point **D** is the point whose ordinate value matches the derivative of the curve at point **B**. It is easier to see how the derivative changes by looking at a plot of it rather than comparing subsequent calculations. We can do that by tracing point **D** as it moves in response to movements of point **B**.

First we'll hide the calculations so that we can better see the trace curve.

1. Press Num ↳Setup to return to Numeric view.

2. Select each calculation in turn and tap √. All calculations should now be deselected.

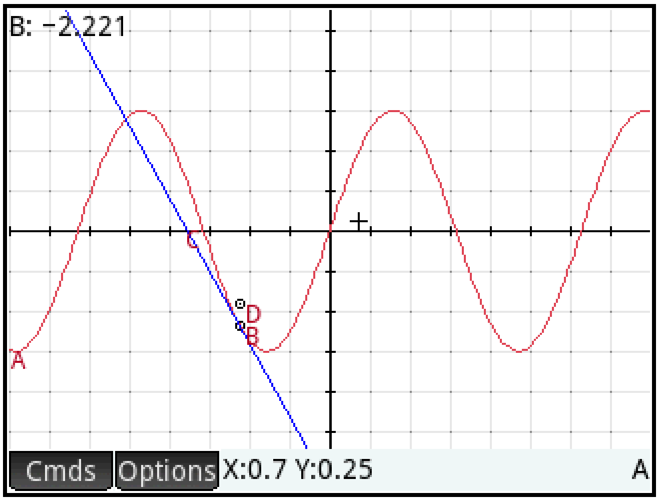
3. Press Plot ↳Setup to return to Plot view.

4. Tap point **D** and then press Enter ≈ to select it.

5. Tap Options (or press Menu) and then select **Trace**. Press Enter to deselect point **D**.

6. Tap point **B** and then press Enter to select it.

7. Using the cursor keys, move point **B** along the curve. Notice that a shadow curve is traced out as you move point **B**. This is the curve of the derivative of 3sin(x). Tap point **B** and then press Enter to deselect it.

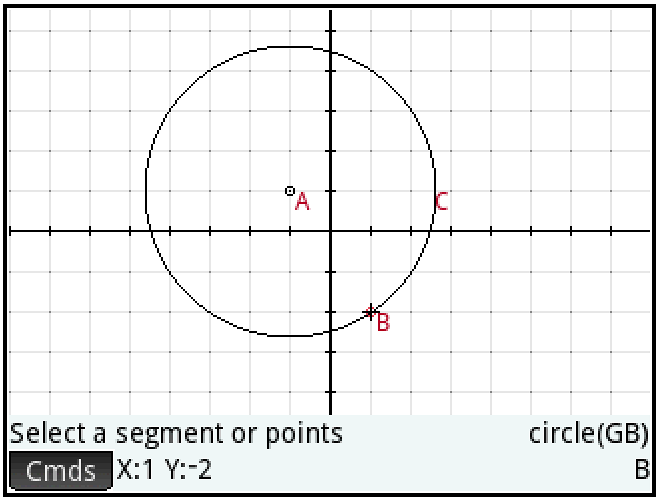


B: −2.221

X:0.7 Y:0.25 A

Plot view in detail

In Plot view you can directly draw objects on the screen using various drawing tools. For example, to draw a circle, tap Cmds, tap **Curve**, and then select **Circle**. Now, tap where you want the center of the circle to be and press Enter. Next, tap a point that is to be on the circumference and press Enter. A circle is drawn with a center at the location of your first tap, and with a radius equal to the distance between your first tap and second tap.



Select a segment or points circle(GB)

X:1 Y:-2 B

Note that there are on-screen instructions to help you. These instructions appear near the bottom of the screen, next to the command listing for the active tool (circle, point, and so on).

You can draw any number of geometric objects in Plot view. See Plot view: Cmds menu on page 170 for a list of the objects you can draw. The drawing tool you choose—line, circle, hexagon, and so on—remains selected until you deselect it. This enables you to quickly draw a number of objects of the same type (such as a number of hexagons). After you have finished drawing objects of a particular type, deselect the drawing tool by pressing Esc Clear . You can tell if a drawing tool is still active by the presence of the on-screen instructions and the command name at the bottom of the screen.

An object in Plot view can be manipulated in numerous ways, and its mathematical properties can be easily determined (see Listing all objects on page 169).

Selecting objects

Selecting an object involves at least two steps: tapping the object and pressing Enter. Pressing Enter is necessary to confirm your intention to select an object.

When you tap a location, objects recognized as being under the pointer are colored light red and added to the list of objects in the bottom right corner of the display. You can select any or all of these objects by pressing Enter. You can tap the screen and then use the cursor keys to accurately position the pointer before pressing Enter.

When more than one object is recognized as being under the pointer, in most cases, preference is given to any point under the pointer when Enter is pressed. In other cases, a pop-up box appears enabling you to select the desired objects.

You can also select multiple objects using a selection box. Tap and hold your finger at the location on the screen that represents one corner of the selection rectangle. Then drag your finger to the opposite corner of the selection rectangle. A light blue selection rectangle is drawn as you drag. Objects that touch this rectangle are selected.

Hiding names

You can choose to hide the name of an object in Plot view:

1. Select the object whose label you want to hide.
2. Tap Options or press Menu Paste.
3. Select **Hide Label**.

Redisplay a hidden name by repeating this procedure and selecting **Show Label**.

Moving objects

There are many ways to move objects. First, to move an object quickly, you can drag the object without selecting it.

Second, you can tap an object and press Enter to select it. Then, you can drag the object to move it quickly or use the cursor keys to move it one pixel at a time. With the second method, you can select multiple objects to move together. When you have finished moving objects, tap a location where there are no objects and press Enter to deselect everything. If you have selected a single object, you can tap the object and press Enter to deselect it.

Third, you can move a point on an object. Each point on an object has a calculation labeled with its name in Plot view. Tap and hold this item to display a slider bar. You can drag the slider or use the cursor keys to move it. Edit appears as a new menu key. Tap this key to display a dialog box where you can specify the start, step, and stop values for the slider. Also, you can create an animation based on this point using the slider. You can set the speed and pause for the animation, as well as its type. To start or stop an animation, select it, tap Options, and then select or clear the **Animate** option.

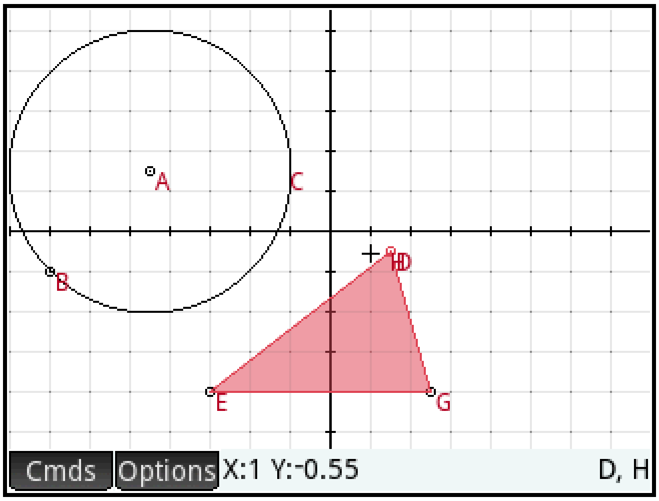
Coloring objects

Objects are colored black by default. The procedure to change the color of an object depends on which view you are in. In both the Symbolic and Numeric view, each item includes a set of color icons. Tap these icons and select a color. In Plot view, select the object, tap Options (or press Menu Paste), tap **Choose Color**, and then select a color.

Filling objects

An object with closed contours (such as a circle or polygon) can be filled with color.

1. Select the object.
2. Tap Options or press Menu Paste.
3. Select **filled**.



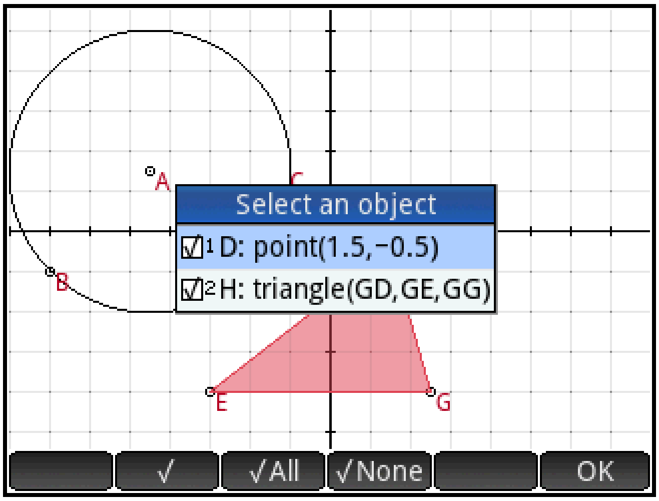
Filled is a toggle. To remove a fill, repeat the above procedure.

Clearing an object

To clear one object, select it and tap Del. Note that an object is distinct from the points you entered to create it. Thus deleting the object does not delete the points that define it. Those points remain in the app.

For example, if you select a circle and press Del, the circle is deleted but the center point and radius point remain.

If other objects are dependent on the one you have selected for deletion, a pop-up displays the selected object and all dependent objects checked for deletion. Confirm your intention by tapping OK.


Select an object

☑1 D: point(1.5,−0.5)

☑2 H: triangle(GD,GE,GG)

You can select multiple items for deletion. Either select them one at a time or use a selection box, and then press Del.

Note that points you add to an object once the object has been defined are cleared when you clear the object. Thus if you place a point (say **D**) on a circle and delete the circle, the circle and **D** are deleted, but the defining points—the center and radius points—remain.

Clearing all objects

To clear the app of all geometric objects, press Shift Esc Clear. You will be asked to confirm your intention to do so. Tap OK to clear all objects defined in Symbolic view or Cancel to keep the app as it is. You can clear all measurements and calculations in Numeric view in the same way.

Gestures in Plot view

You can pan by dragging a finger across the screen: either up, down, left, or right. You can also use the cursor keys to pan once the cursor is at the edge of the screen. You can use a pinch gesture to zoom in or out. Place two fingers on the screen. Move them apart to zoom in or bring them together to zoom out. You can also press + Ans ; to zoom in on the pointer or press − Base ± to zoom out on the pointer.

Zooming

Zooming can be performed in any of the following ways:

- Use a 2-finger pinch zoom.
- Press + or − to zoom in or out, respectively.
- Tap Zoom and choose a zoom option. The zoom options are the same as you find in the Plot view of many apps in the calculator.

Plot view: buttons and keys

Button or key	Purpose
Cmds	Opens the Commands menu. See Plot view: Cmds menu on page 170.
Options	Opens the Options menu for the selected object.
Vars	Hides (or displays) the axes.
Units	Selects the circle drawing tool. Follow the instructions on the screen (or see Circle on page 175).
a b/c	Erases all trace lines.
TAN	Selects the intersection drawing tool. Follow the instructions on the screen (or see Intersection on page 171).
x^2	Selects the line drawing tool. Follow the instructions on the screen (or see Line on page 172).
EEX	Selects the point drawing tool. Follow the instructions on the screen (or see Point on page 170).
9	Selects the segment drawing tool. Follow the instructions on the screen (or see Segment on page 171).
÷	Selects the triangle drawing tool. Follow the instructions on the screen (or see Triangle on page 173).
Del	Deletes a selected object (or the character to the left of the cursor if the entry line is active).
Esc	Deselects the current drawing tool.
Shift Esc	Clears the Plot view of all geometric objects or the Numeric view of all measurements and calculations.

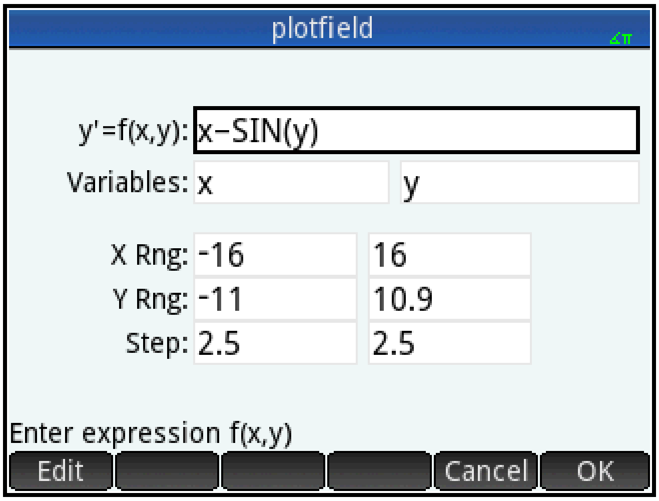
The Options menu

When you select an object, a new menu key appears: Options. Tap this key to view and select options for the selected object, such as color. The Options menu changes depending on the type of object selected. The complete set of Geometry options are listed in the following table and are also displayed when you press Menu Paste.

Option	Purpose
Choose Color	Displays a set of color icons so you can select a color for the selected object.
Hide	Hides the selected object. This is a shortcut for deselecting the object in Symbolic view. To select an object to display after it has been hidden, go to Symbolic or Numeric view.
Hide Label	Hides the label of a selected object. This option changes to **Show Label** if the selected object has a hidden label.
Filled	Fills the selected object with a color. Clear this option to remove the fill.
Trace	Starts tracing for any selected point if selected, then stops tracing for the selected point.
Clear Trace	Erases the current trace of the selected point but does not stop tracing.
Animate	Starts the current animation of a selected point on an object. If the selected point is currently animated, this option stops the animation.

Using the slopefield command

In Symbolic view, selecting the slopefield command enters plotfield() into the command line. To complete the command, enter an expression for y’ and, if necessary, enter values for the other parameters.



plotfield

y'=f(x,y): x–SIN(y)

Variables: x y

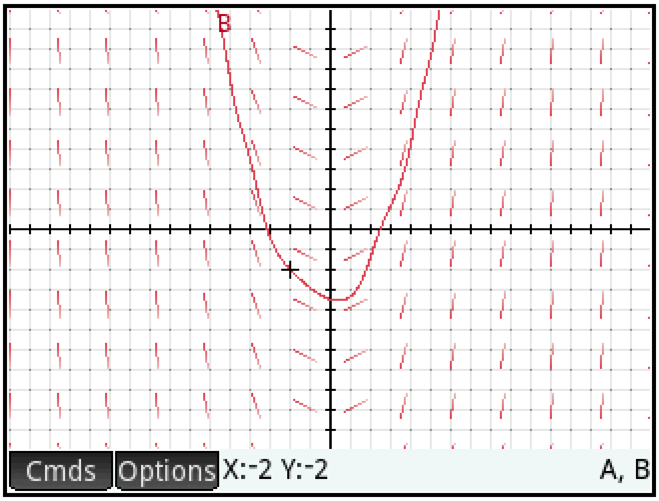
X Rng: -16 16

Y Rng: -11 10.9

Step: 2.5 2.5

Enter expression f(x,y)

In Plot view, selecting the slopefield command opens the Slopefield wizard. In the wizard, you can enter an expression for y’ and, if necessary, enter values for the other parameters.



For example, you can enter the expression `y' = x - sin(y)` and enter `2` for the value of the Step parameters.

NOTE: Enter the variables in lowercase.

Press Plot ↳Setup to display the slopefield in the default Plot view window.

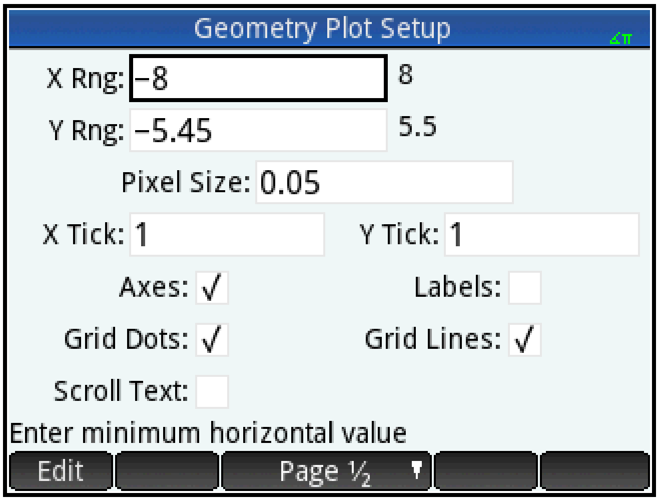
To find the plot of a solution to the expression, move the cursor to a point and press Enter ≈.

For example, move the cursor to the point (-2, -2) and press Enter ≈. The plot of the solution to the equation y' = x – sin(y) with the initial condition x = -2, y = -2 is drawn.

To see the definitions of the slopefield (plotfield) and the solution to the expression (plotode), press Symb ↳Setup.

Plot Setup view

The Plot Setup view enables you to configure the appearance of Plot view.



Geometry Plot Setup

X Rng: –8 8

Y Rng: –5.45 5.5

Pixel Size: 0.05

X Tick: 1 Y Tick: 1

Axes: ☑ Labels: □

Grid Dots: ☑ Grid Lines: ☑

Scroll Text: □

Enter minimum horizontal value

The fields and options are as follows:

- **X Rng**: There are two boxes, but only the minimum x-value is editable. The maximum x-value is calculated automatically, based on the minimum value and the pixel size. You can also change the x range by panning and zooming in Plot view.
- **Y Rng**: There are two boxes, but only the minimum y-value is editable. The maximum y-value is calculated automatically, based on the minimum value and the pixel size. You can also change the y range by panning and zooming in Plot view.
- **Pixel Size**: Each pixel in the Plot view must be square. You can change the size of each pixel. The lower left corner of the Plot view display remains the same, but the upper right-corner coordinates are automatically recalculated.
- **Axes**: A toggle option to hide (or show) the axes in Plot view.

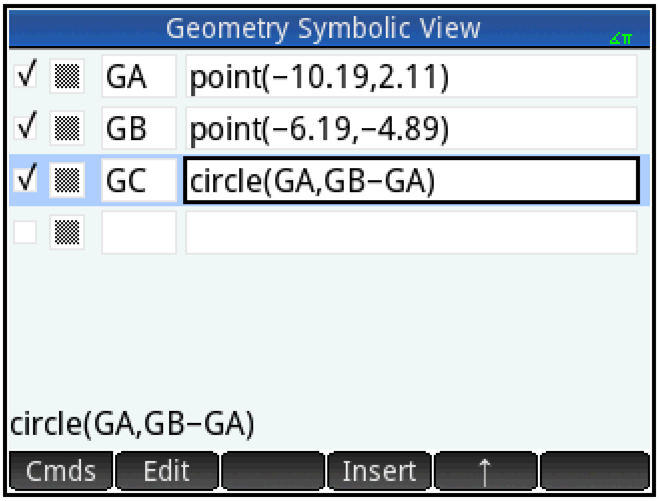
  Keyboard shortcut: Vars Chars A
- **Labels**: A toggle option to hide (or show) the labels for the axes.
- **Grid Dots**: A toggle option to hide (or show) the grid dots.
- **Grid Lines**: A toggle option to hide (or show) the grid lines.

Symbolic view in detail

Every object—whether a point, segment, line, polygon, or curve—is given a name, and its definition is displayed in Symbolic view (Symb Setup). The name is the name for it you see in Plot view, but prefixed by “G”. Thus a point labeled A in Plot view is given the name GA in Symbolic view.

The G-prefixed name is a variable that can be read by the computer algebra system (CAS). Thus in the CAS you can include such variables in calculations. Note in the illustration above that GC is the name of the variable that represents a circle drawn in Plot view. If you are working in the CAS and wanted to know what the area of that circle is, you could enter `area(GC)` and press Enter ≈ .

NOTE: Calculations referencing geometry variables can be made in the CAS or in the Numeric view of the Geometry app (explained below on Numeric view in detail on page 166).



Geometry Symbolic View

√	▩	GA	point(−10.19,2.11)
√	▩	GB	point(−6.19,−4.89)
√	▩	GC	circle(GA,GB−GA)
□	▩		

circle(GA,GB−GA)

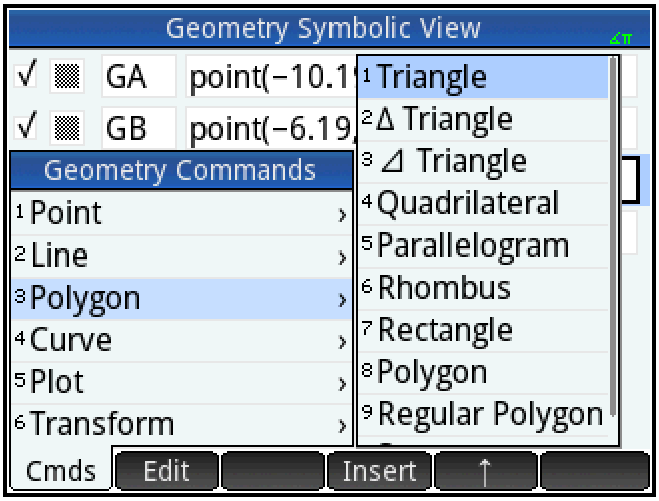
You can change the definition of an object by selecting it, tapping Edit , and altering one or more of its defining parameters. The object is modified accordingly in Plot view. For example, if you selected point **GB** in the illustration above, tapped Edit, changed one or both of the point's coordinates, and tapped OK, you would find, on returning to Plot view, a circle of a different size.

Creating objects

You can also create an object in Symbolic view. Tap New, define the object—for example, `point(4,6)` —and press Enter ≈. The object is created and can be seen in Plot view.

Another example: to draw a line through points P and Q, enter `line(GP,GQ)` in Symbolic view and press Enter ≈. When you return to Plot view, you will see a line passing through points P and Q.

The object-creation commands available in Symbolic view can be seen by tapping Cmds. The syntax for each command is given in Geometry functions and commands on page 189.



Reordering entries

You can reorder the entries in Symbolic view. Objects are drawn in Plot view in the order in which they are defined in Symbolic view. To change the position of an entry, highlight it and tap either ↓ (to move it down the list) or ↑ (to move it up).

Hiding an object

To prevent an object displaying in Plot view, deselect it in Symbolic view:

1. Highlight the item to be hidden.
2. Tap √.

   – or –

   Select the check box for an object and press + Ans to select it and press − Base to clear it.

Repeat the procedure to make the object visible again.

Deleting an object

As well as deleting an object in Plot view (see Clearing an object on page 159) you can delete an object in Symbolic view.

1. Highlight the definition of the object you want to delete.
2. Press Del.

To delete all objects, press Shift Esc Clear. When prompted, tap OK to confirm the deletion.

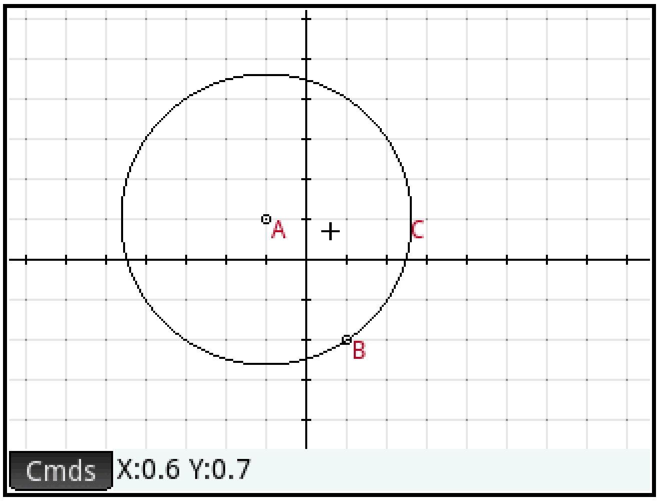
Symbolic Setup view

The Symbolic view of the Geometry app is common with many apps. It is used to override certain system-wide settings.

Numeric view in detail

Numeric view (Num Setup) enables you to do calculations in the Geometry app. The results displayed are dynamic—if you manipulate an object in Plot view or Symbolic view, any calculations in Numeric view that refer to that object are automatically updated to reflect the new properties of that object.

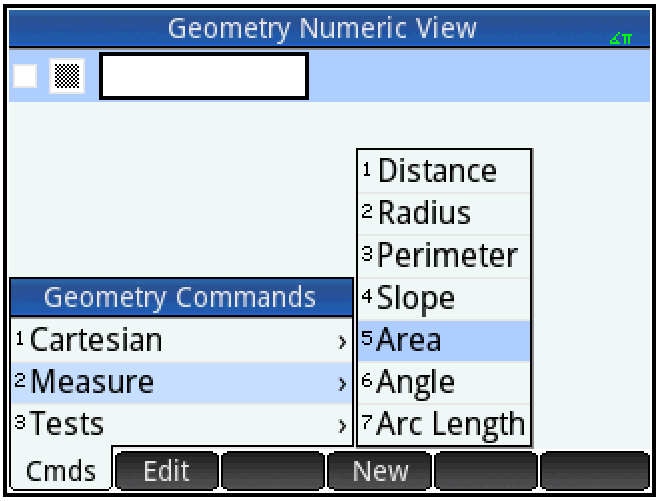
Consider circle **C** in the following figure. To calculate the area and radius of C:



1. Press Num Setup to open Numeric view.
2. Tap New.

3. Tap Cmds and choose **Measure > Area**.

   Note that **area()** appears on the entry line, ready for you to specify the object whose area you are interested in.

4. Tap Vars, choose **Curves** and then the curve whose area you are interested in.

   The name of the object is placed between the parentheses.

   You could have entered the command and object name manually, that is, without choosing them from menus. If you enter object names manually, remember that the name of the object in Plot view must be given a “G” prefix if it is used in any calculation. Thus the circle named **C** in Plot view must be referred to as **GC** in Numeric view and Symbolic view.

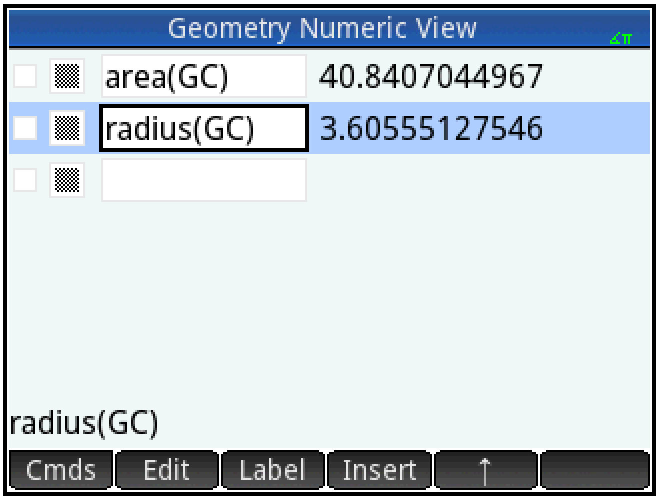
5. Press Enter ≈ or tap OK. The area is displayed.

6. Tap New.

7. Enter `radius(GC)` and tap OK. The radius is displayed. Use √ to verify both of these measurements so that they will be available in Plot view.

Note that the syntax used here is the same as you use in the CAS to calculate the properties of geometric objects.

The Geometry functions and their syntax are described in Geometry functions and commands on page 189.



□	▩	area(GC)	40.8407044967
□	▩	radius(GC)	3.60555127546
□	▩		

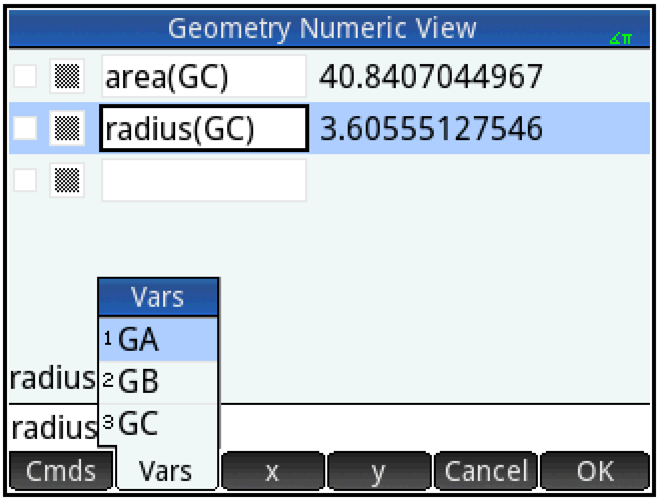
radius(GC)

8. Press Plot ↳Setup to go back to Plot view. Now, manipulate the circle in some way that changes its area and radius. For example, select the center point **(A)** and use the cursor keys to move it to a new location. Notice that the area and radius calculations update automatically as you move the point. Remember to press Esc Clear when you are finished.

NOTE: If an entry in Numeric view is too long for the screen, you can press ▶ to scroll the rest of the entry into view. Press ◀ to scroll back to the original view.

Listing all objects

When you are creating a new calculation in Numeric view, the Vars menu item appears. Tapping Vars gives you a list of all the objects in your Geometry workspace.


Geometry Numeric View

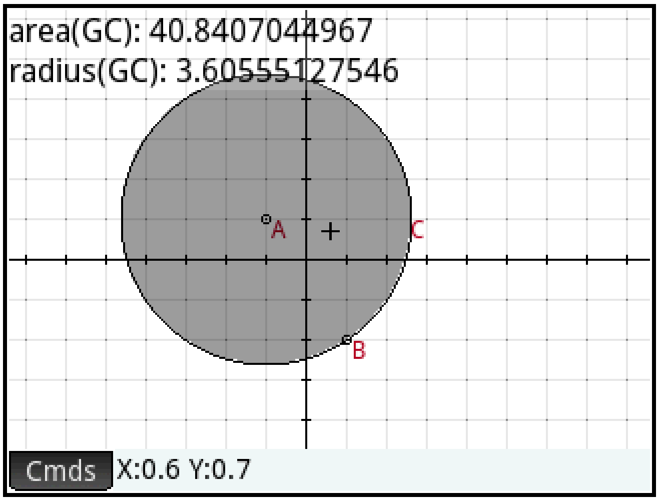
□	▩	area(GC)	40.8407044967
□	▩	radius(GC)	3.60555127546
□	▩		

radius
radius

If you are building a calculation, you can select an object’s variable name from this menu. The name of the selected object is placed at the insertion point on the entry line.

Displaying calculations in Plot view

To have a calculation made in Numeric view appear in Plot view, just highlight it in Numeric view and tap √. A checkmark appears beside the calculation.


area(GC): 40.8407044967
radius(GC): 3.60555127546

X:0.6 Y:0.7

Repeat the procedure to prevent the calculation being displayed in Plot view. The checkmark is cleared.

Editing a calculation

1. Highlight the calculation you want to edit.
2. Tap Edit to change the calculation or tap Label to change the label.
3. Make your changes and tap OK.

Deleting a calculation

1. Highlight the calculation you want to delete.
2. Press Del.

To delete all calculations, press Shift Esc. Note that deleting a calculation does not delete any geometric objects from either the Plot or Symbolic view.

Plot view: Cmds menu

The geometric objects discussed in this section are those that can be created in either Plot view or Symbolic view using the Commands menu (Cmds). This section discusses how to use the commands in Plot view. Objects can also be created in Symbolic view—more, in fact, than in Plot view—but these are discussed in Geometry functions and commands on page 189. Finally, measurements and other calculations can be performed in Plot view as well.

In Plot view, you choose a drawing tool to draw an object. The tools are listed in this section. Note that once you select a drawing tool, it remains selected until you deselect it. This enables you to quickly draw a number of objects of the same type (such as a number of circles). To deselect the current drawing tool, press Esc. You can tell if a drawing tool is still active by the presence of the on-screen help in the bottom left-side of the screen and the current command statement to its right.

The steps provided in this section are based on touch entry. For example, to add a point, the steps will tell you to **tap** the screen where you want the point to be and press Enter. However, you can also use the cursor keys to position the cursor where you want the point to be and then press Enter.

The drawing tools for the geometric objects listed in this section can be selected from the Commands menu at the bottom of the screen (Cmds). Some objects can also be entered using a keyboard shortcut. For example, you can select the triangle drawing tool by pressing ÷. See Plot view: buttons and keys on page 161.

Point

Point

Tap where you want the point to be and press Enter.

Keyboard shortcut: EEX

Point On

Tap the object where you want the new point to be and press Enter. If you select a point that has been placed on an object and then move that point, the point will be constrained to the object on which it was placed. For example, a point placed on a circle will remain on that circle regardless of how you move the point.

Midpoint

Tap where you want one point to be and press Enter. Tap where you want the other point to be and press Enter. A point is automatically created midway between those two points.

If you choose an object first—such as a segment—choosing the Midpoint tool and pressing Enter adds a point midway between the ends of that object. (In the case of a circle, the midpoint is created at the circle's center.)

Center

Tap a circle and press Enter. A point is created at the center of the circle.

Intersection

Tap the desired intersection and press Enter. A point is created at one of the points of intersection.

Keyboard shortcut: TAN

Intersections

Tap one object other than a point and press Enter. Tap another object and press Enter. The point(s) where the two objects intersect are created and named. Note that an intersections object is created in Symbolic view even if the two objects selected do not intersect.

NOTE: This command creates a point. The command uses the location of this point to look for the desired intersection. You can move the point to select a different nearby intersection.

Random Points

Press Enter to randomly create a point in Plot view. Continue pressing Enter to create more random points. Press Esc when you are done.

Line

Segment

Tap where you want one endpoint to be and press Enter. Tap where you want the other endpoint to be and press Enter. A segment is drawn between the two end points.

Keyboard shortcut: 9

Ray

Tap where you want the endpoint to be and press Enter. Tap a point that you want the ray to pass through and press Enter. A ray is drawn from the first point and through the second point.

Line

Tap a point you want the line to pass through and press Enter. Tap another point you want the line to pass through and press Enter. A line is drawn through the two points.

Keyboard shortcut: x^2

Tap a third point **(C)** and press Enter. A line is drawn through **A** bisecting the angle formed by line **AB** and line **AC**.

Parallel

Tap on a point **(P)** and press Enter. Tap on a line **(L)** and press Enter. A new line is draw parallel to **L** and passing through **P**.

Perpendicular

Tap on a point **(P)** and press Enter. Tap on a line **(L)** and press Enter. A new line is draw perpendicular to **L** and passing through **P**.

Tangent

Tap on a curve **(C)** and press Enter. Tap on a point **(P)** and press Enter. If the point (P) is on the curve **(C)**, then a single tangent is drawn. If the point **(P)** is not on the curve **(C)**, then zero or more tangents may be drawn.

Median

Tap on a point **(A)** and press Enter. Tap on a segment and press Enter. A line is drawn through the point **(A)** and the midpoint of the segment.

Altitude

Tap on a point **(A)** and press Enter. Tap on a segment and press Enter. A line is drawn through the point **(A)** perpendicular to the segment (or its extension).

Angle bisector

Tap the point that is the vertex of the angle to be bisected **(A)** and press Enter. Tap another point **(B)** and press Enter.

Polygon

The **Polygon** menu provides tools for drawing various polygons.

Triangle

Tap at each vertex, pressing Enter after each tap.

Keyboard shortcut: ÷

Isosceles Triangle

Draws an isosceles triangle defined by two of its vertices and an angle. The vertices define one of the two sides equal in length and the angle defines the angle between the two sides of equal length. Like `equilateral_triangle`, you have the option of storing the coordinates of the third point into a CAS variable.

```
isosceles_triangle(point1, point2, angle)
```

Example:

`isosceles_triangle(GA, GB, angle(GC, GA, GB)` defines an isosceles triangle such that one of the two sides of equal length is AB, and the angle between the two sides of equal length has a measure equal to that of ∡ACB.

Right Triangle

Draws a right triangle given two points and a scale factor. One leg of the right triangle is defined by the two points, the vertex of the right angle is at the first point, and the scale factor multiplies the length of the first leg to determine the length of the second leg.

`right_triangle(point1, point2, realk)`

Example:

`right_triangle(GA, GB, 1)` draws an isosceles right triangles with its right angle at point A, and with both legs equal in length to segment AB.

Quadrilateral

Tap at each vertex, pressing Enter after each tap.

Parallelogram

Tap one vertex and press Enter. Tap another vertex and press Enter. Tap a third vertex and press Enter. The location of the fourth vertex is automatically calculated and the parallelogram is drawn.

Rhombus

Draws a rhombus, given two points and an angle. As with many of the other polygon commands, you can specify optional CAS variable names for storing the coordinates of the other two vertices as points.

`rhombus(point1, point2, angle)`

Example:

`rhombus(GA, GB, angle(GC, GD, GE))` draws a rhombus on segment AB such that the angle at vertex A has the same measure as ∡DCE.

Rectangle

Draws a rectangle given two consecutive vertices and a point on the side opposite the side defined by the first two vertices or a scale factor for the sides perpendicular to the first side. As with many of the other polygon commands, you can specify optional CAS variable names for storing the coordinates of the other two vertices as points.

`rectangle(point1, point2, point3)` or `rectangle(point1, point2, realk)`

Examples:

`rectangle(GA, GB, GE)` draws a rectangle whose first two vertices are points A and B (one side is segment AB). Point E is on the line that contains the side of the rectangle opposite segment AB.

`rectangle(GA, GB, 3, p, q)` draws a rectangle whose first two vertices are points A and B (one side is segment AB). The sides perpendicular to segment AB have length 3*AB. The third and fourth points are stored into the CAS variables p and q, respectively.

Polygon

Draws a polygon from a set of vertices.

`polygon(point1, point2, …, pointn)`

Example:

`polygon(GA, GB, GD)` draws ΔABD

Regular Polygon

Draws a regular polygon given the first two vertices and the number of sides, where the number of sides is greater than 1. If the number of sides is 2, then the segment is drawn. You can provide CAS variable names for storing the coordinates of the calculated points in the order they were created. The orientation of the polygon is counterclockwise.

`isopolygon(point1, point2, realn)`, where realn is an integer greater than 1.

Example:

`isopolygon(GA, GB, 6)` draws a regular hexagon whose first two vertices are the points A and B.

Square

Tap one vertex and press Enter ≈. Tap another vertex and press Enter ≈. The location of the third and fourth vertices are automatically calculated and the square is drawn.

Curve

Circle

Tap at the center of the circle and press Enter ≈. Tap at a point on the circumference and press Enter ≈. A circle is drawn about the center point with a radius equal to the distance between the two tapped points.

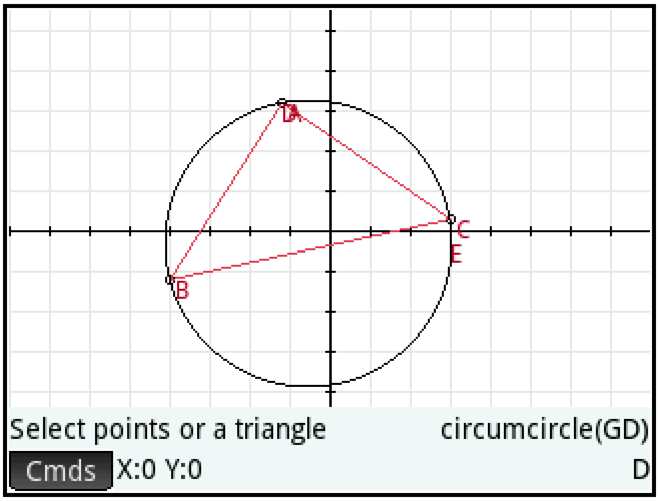
Keyboard shortcut: Units C

You can also create a circle by first defining it in Symbolic view. The syntax is `circle(GA,GB)` where **A** and **B** are two points. A circle is drawn in Plot view such that **A** and **B** define the diameter of the circle.

Circumcircle

A circumcircle is the circle that passes through each of the triangle’s three vertices, thus enclosing the triangle.

Tap at each vertex of the triangle, pressing Enter after each tap.



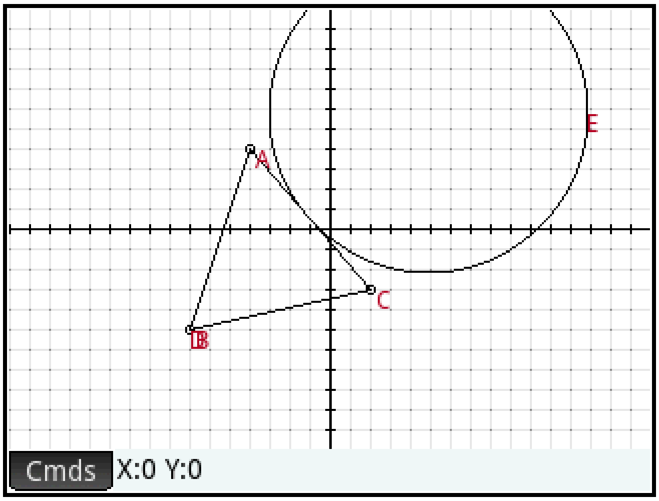
Select points or a triangle circumcircle(GD)

Cmds X:0 Y:0 D

Excircle

An excircle is a circle that is tangent to one segment of a triangle and also tangent to the rays through the segment’s endpoints from the vertex of the triangle opposite the segment. Tap at each vertex of the triangle, pressing Enter after each tap.

The excircle is drawn tangent to the side defined by the last two vertices tapped. In the following figure, the last two vertices tapped were A and C (or C and A). Thus the excircle is drawn tangent to the line segment AC.



Cmds X:0 Y:0

Incircle

An incircle is a circle that is tangent to all three sides of a triangle. Tap each vertex of the triangle, pressing Enter after each tap.

Ellipse

Tap at one focus point and press Enter. Tap at the second focus point and press Enter. Tap at a point on the circumference and press Enter.

Hyperbola

Tap at one focus point and press Enter. Tap at the second focus point and press Enter. Tap at a point on one branch of the hyperbola and press Enter.

Parabola

Tap at the focus point and press Enter. Tap either on a line (the directrix) or a ray or segment and press Enter.

Conic

Plots the graph of a conic section defined by an expression in x and y.

```
conic(expr)
```

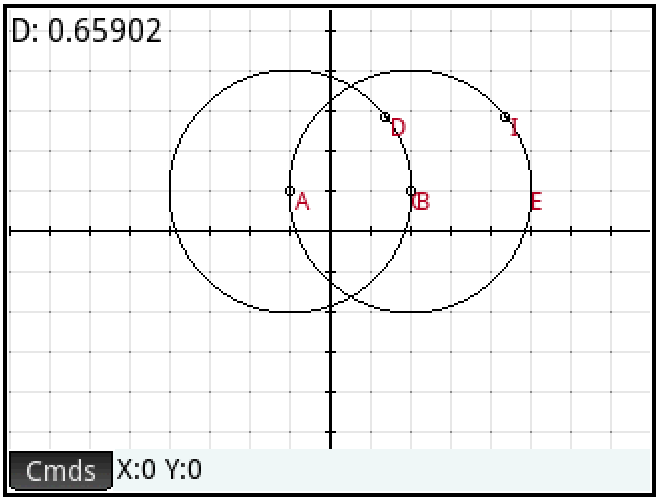
Example:

`conic(x^2+y^2-81)` draws a circle with center at (0,0) and radius of 9

Locus

Takes two points as its arguments: the first is the point whose possible locations form the locus; the second is a point on an object. This second point drives the first through its locus as the second moves on its object.

In the following figure, circle C has been drawn and point D is a point placed on C (using the **Point On** function described above). Point I is a translation of point D. Choosing **Curve > Special > Locus** places **locus(** on the entry line. Complete the command as `locus(GI,GD)` and point I traces a path (its locus) that parallels point D as it moves around the circle to which it is constrained.

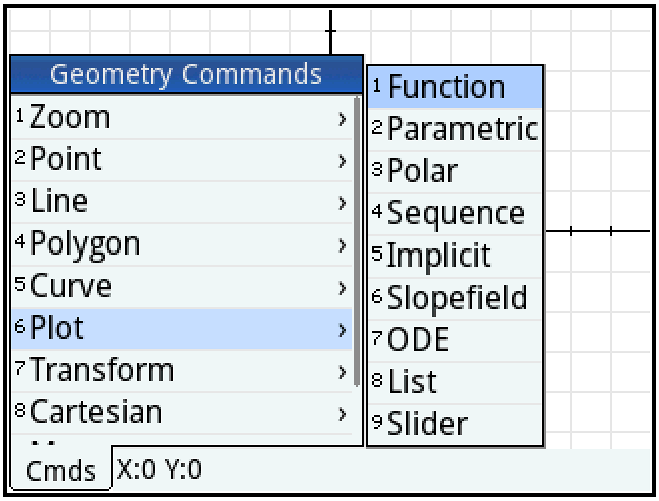


Plot

You can plot expressions of the following types in Plot view:

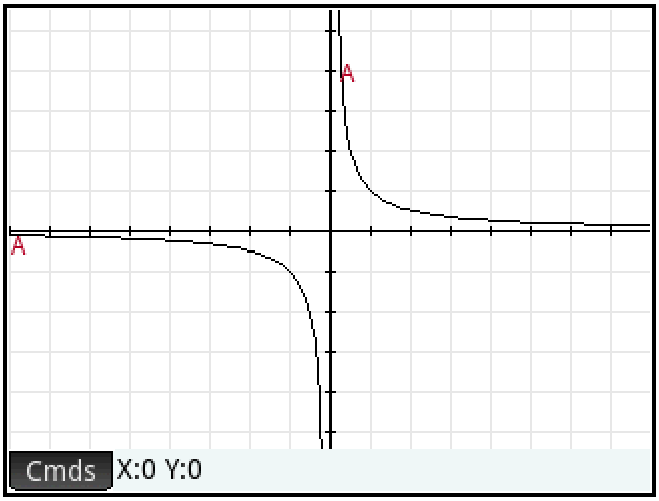
- Function
- Parametric
- Polar
- Sequence

Tap , select **Plot**, and then the type of expression you want to plot. The entry line is enabled for you to define the expression.



Note that the variables you specify for an expression must be in lowercase.

In this example, **Function** has been selected as the plot type and the graph of y = 1/x is plotted.



Function

Syntax: `plotfunc(Expr)`

Draws the plot of a function, given an expression in the independent variable x. An edit line appears. Enter your expression and press Enter. Note the use of lowercase x.

You can also enter an expression in a different variable as long as you declare the variable. To do this, the syntax is `plotfunc(expr(var, var).`

Example:

`plotfunc(3*sin(x))` draws the graph of y=3*sin(x)

`plotfunc(a^2, a) plots the graph of a parabola`

Parametric

Syntax: `plotparam(f(Var)+i*g(Var), Var= Start..Stop, [tstep=Value])`

Takes a complex expression in one variable and an interval for that variable as arguments. Interprets the complex expression `f(t) + i*g(t)` as `x = f(t)` and `y = g(t)` and plots the parametric equation over the interval specified in the second argument. An edit line opens for you to enter the complex expression and the interval.

Examples:

`plotparam(cos(t)+ i*sin(t), t=0..2*π)` plots the unit circle

`plotparam(cos(t)+ i*sin(t), t=0..2*π, tstep=π/3)` plots a regular hexagon inscribed in the unit circle (note the tstep value)

Polar

Syntax: `plotpolar(Expr,Var=Interval, [Step])` or `plotpolar(Expr, Var, Min, Max, [Step])`

Draws a polar graph in Plot view. An edit line opens for you to enter an expression in x as well as an interval (and optional step).

`plotpolar(f(x),x,a,b)` draws the polar curve `r=f(x)` for x in `[a,b]`

Sequence

Syntax: `plotseq(f(Var), Var={Start, Xmin, Xmax}, Integer n)`

Given an expression in x and a list containing three values, draws the line y=x, the plot of the function defined by the expression over the domain defined by the interval between the last two values, and draws the cobweb plot for the first n terms of the sequence defined recursively by the expression (starting at the first value).

Example:

`plotseq(1-x/2, x={3 -1 6}, 5)` plots `y=x` and `y=1-x/2` (from `x=-1` to `x=6`), then draws the first 5 terms of the cobweb plot for `u(n)=1-(u(n-1)/2`, starting at `u(0)=3`

Implicit

Syntax: `plotimplicit(Expr, [XIntrvl, YIntrvl])`

Plots an implicitly defined curve from Expr (in x and y). Specifically, plots Expr=0. Note the use of lowercase x and y. With the optional x-interval and y-interval, this command plots only within those intervals.

Example:

`plotimplicit((x+5)^2+(y+4)^2-1)` plots a circle, centered at the point (-5, -4), with a radius of `1`

Slopefield

Syntax: `plotfield(Expr, [x=X1..X2 y=Y1..Y2], [Xstep, Ystep], [Option])`

Plots the graph of the slopefield for the differential equation y'=f(x,y) over the given x-range and y-range. If Option is `normalize`, the slopefield segments drawn are equal in length.

Example:

`plotfield(x*sin(y), [x=-6..6, y=-6..6],normalize)` draws the slopefield for `y'=x*sin(y)`, from `–6` to `6` in both directions, with segments that are all of the same length

ODE

Syntax: `plotode(Expr, [Var1, Var2, ...], [Val1, Val2. ...])`

Draws the solution of the differential equation y'=f(Var1, Var2, ...) that contains as initial condition for the variables Val1, Val2,... The first argument is the expression f(Var1, Var2,...), the second argument is the vector of variables, and the third argument is the vector of initial conditions.

Example:

`plotode(x*sin(y), [x,y], [-2, 2])` draws the graph of the solution to `y'=x*sin(y)` that passes through the point `(-2, 2)` as its initial condition

List

Syntax: `plotlist(Matrix 2xn)`

Plots a set of n points and connects them with segments. The points are defined by a 2xn matrix, with the abscissas in the first row and the ordinates in the second row.

Example:

`plotlist([[0,3],[2,1],[4,4],[0,3]])` draws a triangle

Slider

Creates a slider bar that can be used to control the value of a parameter. A dialog box displays the slider bar definition and any animation for the slider.

Transform

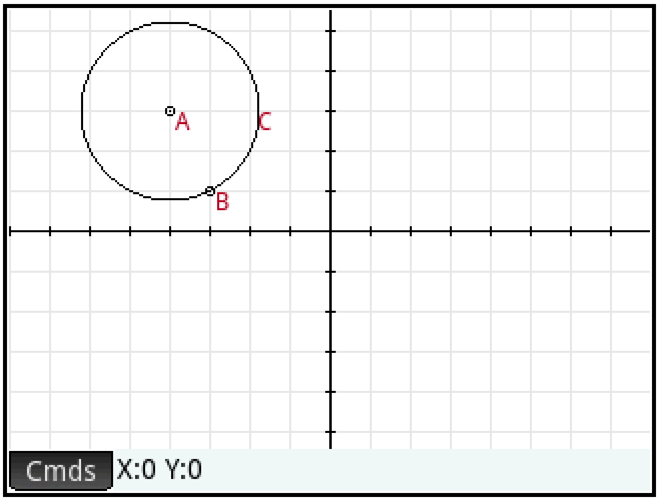
The **Transform** menu provides numerous tools for you to perform transformations on geometric objects in Plot view. You can also define transformations in Symbolic view.

Translation

A translation is a transformation of a set of points that moves each point the same distance in the same direction. T: (x,y)→(x+a, y+b).

Suppose you want to translate circle B in the following figure down a little and to the right:

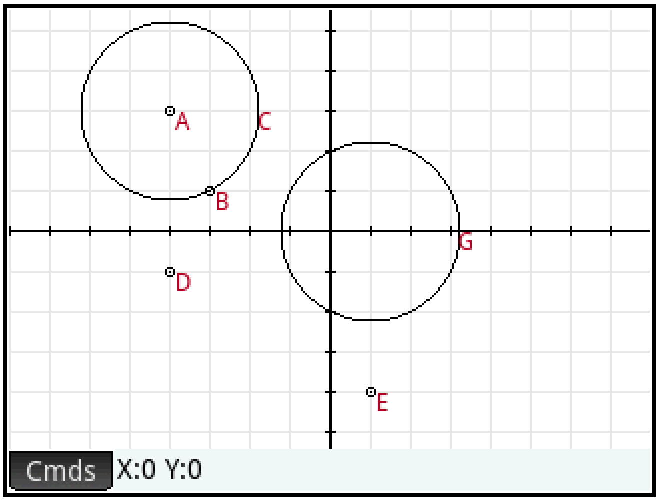
1. Tap Cmds, tap **Transform**, and select **Translation**.
2. Tap the object to be moved and press Enter ≈.



3. Tap an initial location and press Enter ≈.

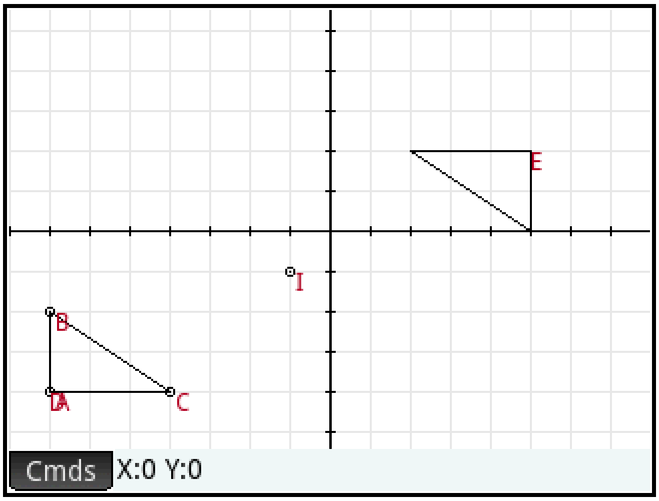
4. Tap a final location and press Enter ≈.

The object is moved the same distance and direction from the initial to the final locations. The original object is left in place.



Reflection

A reflection is a transformation which maps an object or set of points onto its mirror image, where the mirror is either a point or a line. A reflection through a point is sometimes called a half-turn. In either case, each point on the mirror image is the same distance from the mirror as the corresponding point on the original. In the following figure, the original triangle **D** is reflected through point **I**.



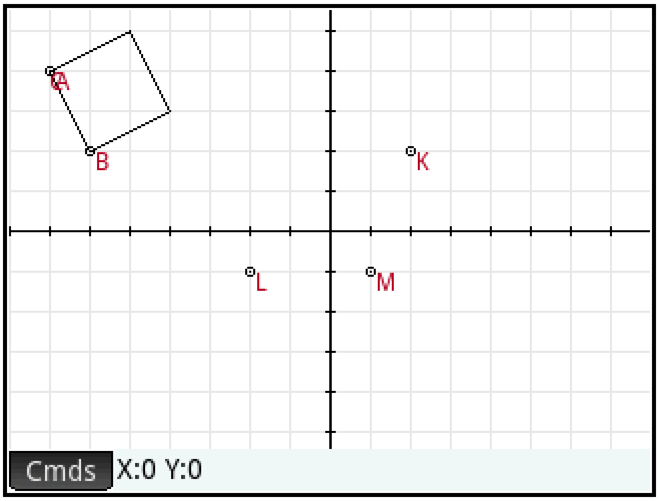
1. Tap Cmds, tap **Transform**, and select **Reflection**.

2. Tap the point or straight object (segment, ray, or line) that will be the symmetry axis (that is, the mirror) and press Enter ≈.

3. Tap the object that is to be reflected across the symmetry axis and press Enter ≈. The object is reflected across the symmetry axis defined in step 2.

Rotation

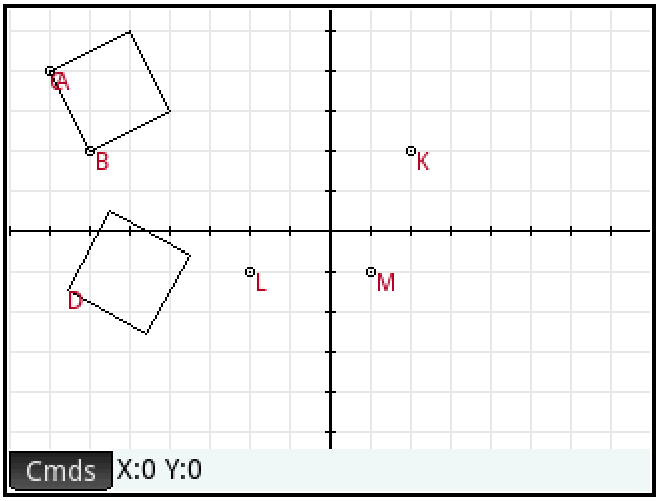
A rotation is a mapping that rotates each point by a fixed angle around a center point. The angle is defined using the `angle()` command, with the vertex of the angle as the first argument. Suppose you wish to rotate the square (**GC**) around point K (**GK**) through ∡**LKM** in the following figure.



1. Tap Cmds, tap **Transform**, and select **Rotation**. `rotation()` appears on the entry line.
2. Between the parentheses, enter:

   `GK,angle(GK,GL,GM),GC`

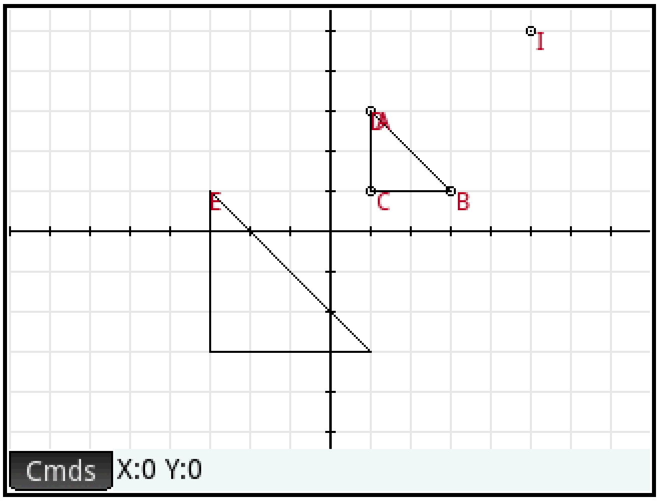
3. Press Enter ≈ or tap OK.
4. Press Plot ↳Setup to return to Plot view to see the rotated square.



Dilation

A dilation (also called a homothety or uniform scaling) is a transformation where an object is enlarged or reduced by a given scale factor around a given point as center.

In the following figure, the scale factor is 2 and the center of dilation is indicated by a point near the top right of the screen (named **I**). Each point on the new triangle is collinear with its corresponding point on the original triangle and point I. Further, the distance from point I to each new point will be twice the distance to the original point (since the scale factor is 2).



1. Tap Cmds, tap **Transform**, and select **Dilation**.
2. Tap the point that is to be the center of dilation and press Enter.
3. Enter the scale factor and press Enter.
4. Tap the object that is to be dilated and press Enter.

Similarity

Dilates and rotates a geometric object about the same center point.

`similarity(point, realk, angle, object)`

Example:

`similarity(0, 3, angle(0,1,i),point(2,0))` dilates the point at (2,0) by a scale factor of 3 (a point at (6,0)), then rotates the result 90° counterclockwise to create a point at (0, 6).

Projection

A projection is a mapping of one or more points onto an object such that the line passing through the point and its image is perpendicular to the object at the image point.

1. Tap Cmds, tap **Transform**, and select **Projection**.
2. Tap the object onto which points are to be projected and press Enter.
3. Tap the point that is to be projected and press Enter.

   Note the new point added to the target object.

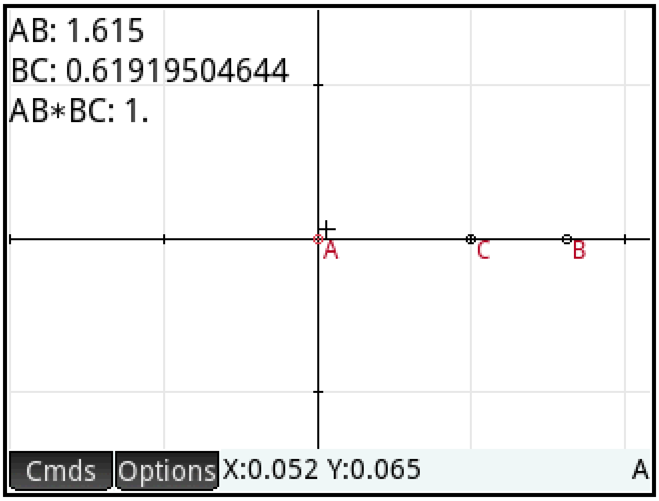
Inversion

An inversion is a mapping involving a center point and a scale factor. Specifically, the inversion of point A through center C, with scale factor k, maps A onto A’, such that A’ is on line CA and CA*CA’=k, where CA and CA’ denote the lengths of the corresponding segments. If k=1, then the lengths CA and CA’ are reciprocals.

Suppose you wish to find the inversion of point B with respect to point A.

1. Tap Cmds, tap **Transform**, and select **Inversion**.
2. Tap point **B** and press Enter.
3. Enter the inversion ratio—use the default value of 1—and press Enter.
4. Tap point **A** and press Enter.

   In the figure, point **C** is the inversion of point **B** in respect to point **A**.



AB: 1.615
BC: 0.61919504644
AB∗BC: 1.

X:0.052 Y:0.065

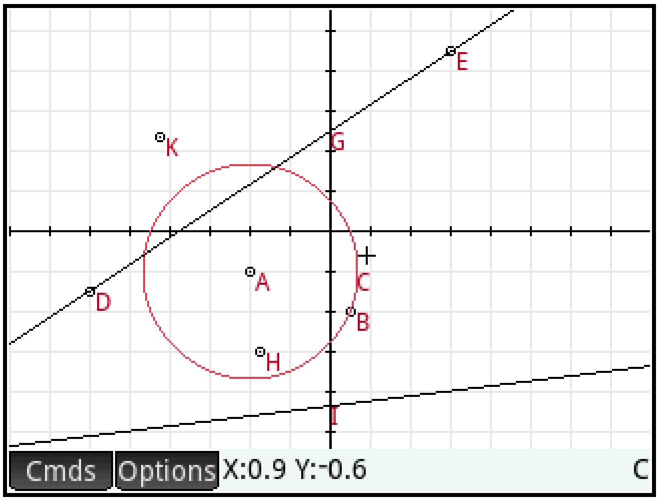
Reciprocation

A reciprocation is a special case of inversion involving circles. A reciprocation with respect to a circle transforms each point in the plane into its polar line. Conversely, the reciprocation with respect to a circle maps each line in the plane into its pole.

1. Tap Cmds, tap **Transform**, and select **Reciprocation**.
2. Tap the circle and press Enter.
3. Tap a point and press Enter to see its polar line.

4. Tap a line and press Enter to see its pole.

   In the following figure, point **K** is the reciprocation of line **DE** (G) and Line **I** (at the bottom of the display) is the reciprocation of point **H**.



Cartesian

Abscissa

Tap a point and press Enter to select it. The abscissa (x-coordinate) of the point will appear at the top left of the screen.

Ordinate

Tap a point and press Enter to select it. The ordinate (y-coordinate) of the point will appear at the top left of the screen.

Point→Complex

Tap a point or a vector and press Enter to select it. The coordinates of the point (or the x- and y-lengths of the vector) will appear as a complex number at the top left of the screen.

Coordinates

Tap a point and press Enter to select it. The coordinates of the point will appear at the top left of the screen.

Equation of

Tap an object other than a point and press Enter to select it. The equation of the object (in x and/or y) is displayed.

Parametric

Tap an object other than a point and press Enter to select it. The parametric equation of the object (x(t)+i*y(t)) is displayed.

Polar coordinates

Tap a point and press Enter to select it. The polar coordinates of the point will appear at the top left of the screen.

Measure

Distance

Tap a point and press Enter to select it. Repeat to select a second point. The distance between the two points is displayed.

Radius

Tap a circle and press Enter to select it. The radius of the circle is displayed.

Perimeter

Tap a circle and press Enter to select it. The perimeter of the circle is displayed.

Slope

Tap a straight object (segment, line, and so on) and press Enter to select it. The slope of the object is displayed.

Area

Tap a circle or polygon and press Enter to select it. The area of the object is displayed.

Angle

Tap a point and press Enter to select it. Repeat to select three points. The measure of the directed angle from the second point through the third point, with the first point as vertex, is displayed.

Arc Length

Tap a curve and press Enter to select it. Then, enter a start value and a stop value. The length of the arc on the curve between the two x-values is displayed.

Tests

Collinear

Tap a point and press Enter to select it. Repeat to select three points. The test appears at the top of the display, along with its result. The test returns 1 if the points are collinear; otherwise, it returns 0.

On circle

Tap a point and press Enter to select it. Repeat to select four points. The test appears at the top of the display, along with its result. The test returns 1 if the points are on the same circle; otherwise, it returns 0.

On object

Tap a point and press Enter to select it. Then tap an object and press Enter. The test appears at the top of the display, along with its result. The test returns a number (1 to n number of sides) representing the segment that contains the point if the point is on the object; otherwise, it returns 0.

Parallel

Tap a straight object (segment, line, and so on) and press Enter to select it. Then tap another straight object and press Enter. The test appears at the top of the display, along with its result. The test returns 1 if the objects are parallel; otherwise, it returns 0.

Perpendicular

Tap a straight object (segment, line, and so on) and press Enter to select it. Then tap another straight object and press Enter. The test appears at the top of the display, along with its result. The test returns 1 if the objects are perpendicular; otherwise, it returns 0.

Isosceles

Tap a triangle and press Enter to select it. Or select three points in order. Returns 0 if the triangle is not isosceles or if the three points do not form an isosceles triangle. If the triangle is isosceles (or the three points form an isosceles triangle), returns the number order of the common point of the two sides of equal length (1, 2, or 3). Returns 4 if the three points form an equilateral triangle or if the selected triangle is equilateral.

Equilateral

Tap a triangle and press Enter to select it. Or select three points in order. Returns 1 if the triangle is equilateral or if the three points form an equilateral triangle; otherwise, it returns 0.

Parallelogram

Tap a point and press Enter to select it. Repeat to select four points. The test appears at the top of the display, along with its result. The test returns 0 if the points do not form a parallelogram. Returns 1 if they form a parallelogram, 2 if they form a rhombus, 3 if they form a rectangle, and 4 if they form a square.

Conjugate

Tap a circle and press Enter to select it. Then, select two points or two lines. The test returns 1 if the two points or lines are conjugates for the circle; otherwise, it returns 0.

Geometry functions and commands

The list of geometry-specific functions and commands in this section covers those that can be found by tapping Cmds in both Symbolic and Numeric view and those that are only available from the Catlg menu.

Calculations referring to geometric objects—in the Numeric view of the Geometry app and in the CAS—must use the G-prefixed name given for it in Symbolic view.

For example, `altitude(GA,GB,GC)` is the form you need to use in calculations.

Further, in many cases the specified parameters in the following syntax can be the name of a point (such as GA) or a complex number representing a point.

Thus `angle(A,B,C)` could be:

- `angle(GP, GR, GB)`
- `angle(3+2i, 1-2i, 5+i)` or
- a combination of named points and points defined by a complex number, as in `angle(GP,1-2*i,i)`.

Symbolic view: Cmds menu

For the most part, the Commands menu in Symbolic View is the same as it is in Plot view. The Zoom category does not appear in Symbolic view, nor do the Cartesian, Measure, and Tests categories, although the latter three appear in Numeric view. In Symbolic view, the commands are entered using their syntax. Highlight a command and press Help to learn its syntax. The advantage of entering or editing a definition in Symbolic view is that you can specify the exact location of points. After the exact locations of points are entered, the properties of any dependent objects (lines, circles, and so on) are reported exactly by the CAS. Use this fact to test conjectures on geometric objects using the Test commands. All these commands can be used in the CAS view, where they return the same objects.

Point

Point

Creates a point, given the coordinates of the point. Each coordinate may be a value or an expression involving variables or measurements on other objects in the geometric construction.

`point(real1, real2)` or `point(expr1, expr2)`

Examples:

`point(3,4)` creates a point whose coordinates are (3,4). This point may be selected and moved later.

`point(abscissa(A), ordinate(B))` creates a point whose x-coordinate is the same as that of a point A and whose y-coordinate is the same as that of a point B. This point will change to reflect the movements of point A or point B.

Point on

Creates a point on a geometric object whose abscissa is a given value or creates a real value on a given interval.

`element(object, real)` or `element(real1..real2)`

Examples:

`element(plotfunc(x^2),-2)` creates a point on the graph of $y = x^2$. Initially, this point will appear at (–2,4). You can move the point, but it will always remain on the graph of its function.

`element(0..5)` creates a slider bar with a value of 2.5 initially. Tap and hold this value to open the slider. Select ▶ or ◀ to increase or decrease the value on the slider bar. Press Esc Clear to close the slider bar. The value that you set can be used as a coefficient in a function that you subsequently plot or in some other object or calculation.

Midpoint

Returns the midpoint of a segment. The argument can be either the name of a segment or two points that define a segment. In the latter case, the segment need not actually be drawn.

`midpoint(segment)` or `midpoint(point1, point2)`

Example:

`midpoint(0,6+6i)` returns `point(3,3)`

Center

Syntax: `center(Circle)`

Plots the center of a circle. The circle can be defined by the circle command or by name (for example, **GC**).

Example:

`center(circle(x^2+y2-x-y))` plots `point(1/2,1/2)`

Intersection

Syntax: `single_inter(Curve1, Curve2, [Point])`

Plots the intersection of Curve1 and Curve2 that is closest to Point.

Example:

`single_inter(line(y=x), circle(x^2+y^2=1), point(1,1))` plots `point((1+i)*√2/2)`

Intersections

Returns the intersection of two curves as a vector.

`inter(Curve1, Curve2)`

Example:

`inter(8-x^2/6, x/2-1)` returns `[[6 2],[-9 -11/2]]`

NOTE: This command creates a point. The command uses the location of this point to look for the desired intersection. You can move the point to select a different nearby intersection.

Line

Segment

Draws a segment defined by its endpoints.

```
segment(point1, point2)
```

Examples:

`segment(1+2i, 4)` draws the segment defined by the points whose coordinates are (1, 2) and (4, 0).

`segment(GA, GB)` draws segment AB.

Ray

Given 2 points, draws a ray from the first point through the second point.

```
half_line((point1, point2)
```

Line

Draws a line. The arguments can be two points, a linear expression of the form a*x+b*y+c, or a point and a slope as shown in the examples.

`line(point1, point2)` or `line(a*x+b*y+c)` or `line(point1, slope=realm)`

Examples:

`line(2+i, 3+2i)` draws the line whose equation is y=x–1; that is, the line through the points (2,1) and (3,2).

`line(2x-3y-8)` draws the line whose equation is 2x–3y=8

`line(3-2i,slope=1/2)` draws the line whose equation is x–2y=7; that is, the line through (3, –2) with slope m=1/2.

Parallel

Draws a line through a given point that is parallel to a given line.

```
parallel(point,line)
```

Examples:

`parallel(A, B)` draws the line through point A that is parallel to line B.

`parallel(3-2i, x+y-5)` draws the line through the point (3, –2) that is parallel to the line whose equation is x+y=5; that is, the line whose equation is y=–x+1.

Perpendicular

Draws a line through a given point that is perpendicular to a given line. The line may be defined by its name, two points, or an expression in x and y.

`perpendicular(point, line)` or `perpendicular(point1, point2, point3)`

Examples:

`perpendicular(GA, GD)` draws a line perpendicular to line D through point A.

`perpendicular(3+2i, GB, GC)` draws a line through the point whose coordinates are (3, 2) that is perpendicular to line BC.

`perpendicular(3+2i,line(x–y=1))` draws a line through the point whose coordinates are (3, 2) that is perpendicular to the line whose equation is x – y = 1; that is, the line whose equation is y = –x + 5.

Tangent

Draws the tangent(s) to a given curve through a given point. The point does not have to be a point on the curve.

`tangent(curve, point)`

Examples:

`tangent(plotfunc(x^2), GA)` draws the tangent to the graph of y=x^2 through point A.

`tangent(circle(GB, GC-GB), GA)` draws one or more tangent lines through point A to the circle whose center is at point B and whose radius is defined by segment BC.

Median

Given three points that define a triangle, creates the median of the triangle that passes through the first point and contains the midpoint of the segment defined by the other two points.

`median_line(point1, point2, point3)`

Example:

`median_line(0, 8i, 4)` draws the line whose equation is y=2x; that is, the line through (0,0) and (2,4), the midpoint of the segment whose endpoints are (0, 8) and (4, 0).

Altitude

Given three non-collinear points, draws the altitude of the triangle defined by the three points that passes through the first point. The triangle does not have to be drawn.

`altitude(point1, point2, point3)`

Example:

`altitude(A, B, C)` draws a line passing through point A that is perpendicular to line BC.

Bisector

Given three points, creates the bisector of the angle defined by the three points whose vertex is at the first point. The angle does not have to be drawn in the Plot view.

```
bisector(point1, point2, point3)
```

Examples:

`bisector(A,B,C)` draws the bisector of ∡BAC.

`bisector(0,-4i,4)` draws the line given by y=–x

Polygon

Triangle

Draws a triangle, given its three vertices.

```
triangle(point1, point2, point3)
```

Example:

`triangle(GA, GB, GC)` draws ΔABC.

Isosceles Triangle

Draws an isosceles triangle defined by two of its vertices and an angle. The vertices define one of the two sides equal in length and the angle defines the angle between the two sides of equal length. Like `equilateral_triangle`, you have the option of storing the coordinates of the third point into a CAS variable.

```
isosceles_triangle(point1, point2, angle)
```

Example:

`isosceles_triangle(GA, GB, angle(GC, GA, GB)` defines an isosceles triangle such that one of the two sides of equal length is AB, and the angle between the two sides of equal length has a measure equal to that of ∡ACB.

Right Triangle

Draws a right triangle given two points and a scale factor. One leg of the right triangle is defined by the two points, the vertex of the right angle is at the first point, and the scale factor multiplies the length of the first leg to determine the length of the second leg.

```
right_triangle(point1, point2, realk)
```

Example:

`right_triangle(GA, GB, 1)` draws an isosceles right triangles with its right angle at point A, and with both legs equal in length to segment AB.

Quadrilateral

Draws a quadrilateral from a set of four points.

```
quadrilateral(point1, point2, point3, point4)
```

Example:

`quadrilateral(GA, GB, GC, GD)` draws quadrilateral ABCD.

Parallelogram

Draws a parallelogram given three of its vertices. The fourth point is calculated automatically but is not defined symbolically. As with most of the other polygon commands, you can store the fourth point's coordinates into a CAS variable. The orientation of the parallelogram is counterclockwise from the first point.

`parallelogram(point1, point2, point3)`

Example:

`parallelogram(0,6,9+5i)` draws a parallelogram whose vertices are at (0, 0), (6, 0), (9, 5), and (3,5). The coordinates of the last point are calculated automatically.

Rhombus

Draws a rhombus, given two points and an angle. As with many of the other polygon commands, you can specify optional CAS variable names for storing the coordinates of the other two vertices as points.

`rhombus(point1, point2, angle)`

Example:

`rhombus(GA, GB, angle(GC, GD, GE))` draws a rhombus on segment AB such that the angle at vertex A has the same measure as ∡DCE.

Rectangle

Draws a rectangle given two consecutive vertices and a point on the side opposite the side defined by the first two vertices or a scale factor for the sides perpendicular to the first side. As with many of the other polygon commands, you can specify optional CAS variable names for storing the coordinates of the other two vertices as points.

`rectangle(point1, point2, point3)` or `rectangle(point1, point2, realk)`

Examples:

`rectangle(GA, GB, GE)` draws a rectangle whose first two vertices are points A and B (one side is segment AB). Point E is on the line that contains the side of the rectangle opposite segment AB.

`rectangle(GA, GB, 3, p, q)` draws a rectangle whose first two vertices are points A and B (one side is segment AB). The sides perpendicular to segment AB have length 3*AB. The third and fourth points are stored into the CAS variables p and q, respectively.

Polygon

Draws a polygon from a set of vertices.

`polygon(point1, point2, …, pointn)`

Example:

`polygon(GA, GB, GD)` draws ΔABD

Regular Polygon

Draws a regular polygon given the first two vertices and the number of sides, where the number of sides is greater than 1. If the number of sides is 2, then the segment is drawn. You can provide CAS variable names for storing the coordinates of the calculated points in the order they were created. The orientation of the polygon is counterclockwise.

`isopolygon(point1, point2, realn)`, where realn is an integer greater than 1.

Example:

`isopolygon(GA, GB, 6)` draws a regular hexagon whose first two vertices are the points A and B.

Square

Draws a square, given two consecutive vertices as points.

`square(point1, point2)`

Example:

`square(0, 3+2i, p, q)` draws a square with vertices at (0, 0), (3, 2), (1, 5), and (-2, 3). The last two vertices are computed automatically and are saved into the CAS variables p and q.

Curve

Circle

Draws a circle, given the endpoints of the diameter, or a center and radius, or an equation in x and y.

`circle(point1, point2)` or `circle(point1, point 2-point1)` or `circle(equation)`

Examples:

`circle(GA, GB)` draws the circle with diameter AB.

`circle(GA, GB-GA)` draws the circle with center at point A and radius AB.

`circle(x^2+y^2=1)` draws the unit circle.

This command can also be used to draw an arc.

`circle(GA, GB, 0, π/2)` draws a quarter-circle with diameter AB.

Circumcircle

Draws the circumcircle of a triangle; that is, the circle circumscribed about a triangle.

`circumcircle(point1, point2, point3)`

Example:

`circumcircle(GA, GB, GC)` draws the circle circumscribed about ΔABC.

Excircle

Given three points that define a triangle, draws the excircle of the triangle that is tangent to the side defined by the last two points and also tangent to the extensions of the two sides where the common vertex is the first point.

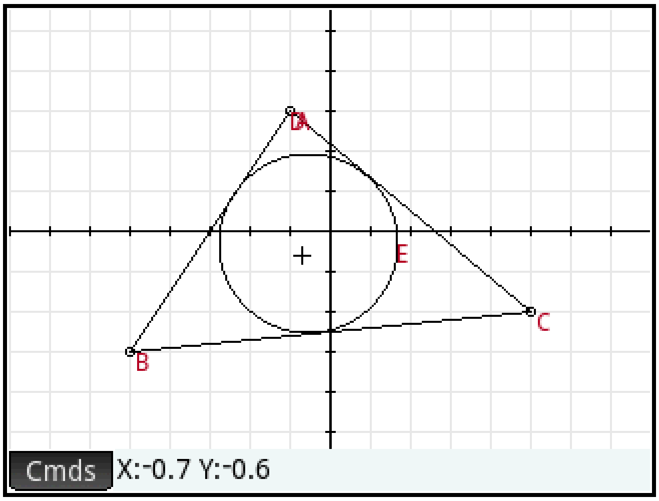
Example:

`excircle(GA, GB, GC)` draws the circle tangent to segment BC and to the rays AB and AC.

Incircle

An incircle is a circle that is tangent to each of a polygon’s sides. The HP Prime can draw an incircle that is tangent to the sides of a triangle.

Tap at each vertex of the triangle, pressing Enter after each tap.



Ellipse

Draws an ellipse, given the foci and either a point on the ellipse or a scalar that is one half the constant sum of the distances from a point on the ellipse to each of the foci.

`ellipse(point1, point2, point3)` or `ellipse(point1, point2, realk)`

Examples:

`ellipse(GA, GB, GC)` draws the ellipse whose foci are points A and B and which passes through point C.

`ellipse(GA, GB, 3)` draws an ellipse whose foci are points A and B. For any point P on the ellipse, AP +BP=6.

Hyperbola

Draws a hyperbola, given the foci and either a point on the hyperbola or a scalar that is one half the constant difference of the distances from a point on the hyperbola to each of the foci.

`hyperbola(point1, point2, point3)` or `hyperbola(point1, point2, realk)`

Examples:

`hyperbola(GA, GB, GC)` draws the hyperbola whose foci are points A and B and which passes through point C.

`hyperbola(GA, GB, 3)` draws a hyperbola whose foci are points A and B. For any point P on the hyperbola, |AP-BP|=6.

Parabola

Draws a parabola, given a focus point and a directrix line, or the vertex of the parabola and a real number that represents the focal length.

`parabola(point,line)` or `parabola(vertex,real)`

Examples:

`parabola(GA, GB)` draws a parabola whose focus is point A and whose directrix is line B.

`parabola(GA, 1)` draws a parabola whose vertex is point A and whose focal length is 1.

Conic

Plots the graph of a conic section defined by an expression in x and y.

`conic(expr)`

Example:

`conic(x^2+y^2-81)` draws a circle with center at (0,0) and radius of 9

Locus

Given a first point and a second point that is an element of (a point on) a geometric object, draws the locus of the first point as the second point traverses its object.

`locus(point,element)`

Plot

Function

Draws the plot of a function, given an expression in the independent variable x. Note the use of lowercase x.

Syntax: `plotfunc(Expr)`

Example:

`plotfunc(3*sin(x))` draws the graph of `y=3*sin(x)`

Parametric

Takes a complex expression in one variable and an interval for that variable as arguments. Interprets the complex expression f(t)+i*g(t) as x=f(t) and y=g(t) and plots the parametric equation over the interval specified in the second argument.

Syntax: `plotparam(f(Var)+i*g(Var), Var= Start..Stop, [tstep=Value])`

Examples:

`plotparam(cos(t)+ i*sin(t), t=0..2*π)` plots the unit circle

`plotparam(cos(t)+ i*sin(t), t=0..2*π, tstep=π/3)` plots a regular hexagon inscribed in the unit circle (note the tstep value)

Polar

Draws a polar plot.

Syntax: `plotpolar(Expr,Var=Interval, [Step])` or `plotpolar(Expr, Var, Min, Max, [Step])`

Example:

`plotpolar(f(x),x,a,b)` draws the polar curve `r=f(x)` for x in `[a,b]`

Sequence

Given an expression in x and a list containing three values, draws the line y=x, the plot of the function defined by the expression over the domain defined by the interval between the last two values, and draws the cobweb plot for the first n terms of the sequence defined recursively by the expression (starting at the first value).

Syntax: `plotseq(f(Var), Var={Start, Xmin, Xmax}, Integern)`

Example:

`plotseq(1-x/2, x={3 -1 6}, 5)` plots `y=x` and `y=1-x/2` (from `x=-1` to `x=6`), then draws the first 5 terms of the cobweb plot for `u(n)=1-(u(n-1)/2`, starting at `u(0)=3`

Implicit

Plots an implicitly defined curved from Expr (in x and y). Specifically, plots Expr=0. Note the use of lowercase x and y. With the optional x-interval and y-interval, plots only within those intervals.

Syntax: `plotimplicit(Expr, [XIntrvl, YIntrvl])`

Example:

`plotimplicit((x+5)^2+(y+4)^2-1)` plots a circle, centered at the point `(-5, -4)`, with a radius of `1`

Slopefield

Plots the graph of the slopefield for the differential equation y'=f(x,y), where f(x,y) is contained in Expr. VectorVar is a vector containing the variables. If VectorVar is of the form [x=Interval, y=Interval], then the slopefield is plotted over the specified x-range and y-range. Given xstep and ystep values, plots the slopefield segments using these steps. If Option is `normalize`, then the slopefield segments drawn are equal in length.

Syntax: `plotfield(Expr, VectorVar, [xstep=Val, ystep=Val, Option])`

Example:

`plotfield(x*sin(y), [x=-6..6, y=-6..6],normalize)` draws the slopefield for `y'=x*sin(y)`, from `-6` to `6` in both directions, with segments that are all of the same length.

ODE

Draws the solution of the differential equation y'=f(Va1, Var2, ...) that contains as initial condition for the variables Val1, Val2,... The first argument is the expression f(Var1, Var2,...), the second argument is the vector of variables, and the third argument is the vector of initial conditions.

Syntax: `plotode(Expr, [Var1, Var2, ...], [Val1, Val2. ...])`

Example:

`plotode(x*sin(y), [x,y], [-2, 2])` draws the graph of the solution to `y'=x*sin(y)` that passes through the point `(-2, 2)` as its initial condition

List

Plots a set of n points and connects them with segments. The points are defined by a 2xn matrix, with the abscissas in the first row and the ordinates in the second row.

Syntax: `plotlist(Matrix 2xn)`

Example:

`plotlist([[0,3],[2,1],[4,4],[0,3]])` draws a triangle

Slider

Creates a slider bar that can be used to control the value of a parameter. A dialog box displays the slider bar definition and any animation for the slider. When completed, the slider bar appears near the top left of Plot view. You can then move it to another location.

Transform

Translation

Translates a geometric object along a given vector. The vector is given as the difference of two points (head-tail).

```
translation(vector, object)
```

Examples:

`translation(0-i, GA)` translates object A down one unit.

`translation(GB-GA, GC)` translates object C along the vector AB.

Reflection

Reflects a geometric object over a line or through a point. The latter is sometimes referred to as a half-turn.

`reflection(line, object)` or `reflection(point, object)`

Examples:

`reflection(line(x=3),point(1,1))` reflects the point at (1, 1) over the vertical line x=3 to create a point at (5,1).

`reflection(1+i, 3-2i)` reflects the point at (3,–2) through the point at (1, 1) to create a point at (–1, 4).

Rotation

Rotates a geometric object, about a given center point, through a given angle.

```
rotate(point, angle, object)
```

Example:

`rotate(GA, angle(GB, GC, GD),GK)` rotates the geometric object labeled K, about point A, through an angle equal to ∡CBD.

Dilation

Dilates a geometric object, with respect to a center point, by a scale factor.

```
homothety(point, realk, object)
```

Example:

`homothety(GA, 2, GB)` creates a dilation centered at point A that has a scale factor of 2. Each point P on geometric object B has its image P’ on ray AP such that AP’=2AP.

Similarity

Dilates and rotates a geometric object about the same center point.

`similarity(point, realk, angle, object)`

Example:

`similarity(0, 3, angle(0,1,i),point(2,0))` dilates the point at (2,0) by a scale factor of 3 (a point at (6,0)), then rotates the result 90° counterclockwise to create a point at (0, 6).

Projection

Draws the orthogonal projection of a point onto a curve.

`projection(curve, point)`

Inversion

Draws the inversion of a point, with respect to another point, by a scale factor.

`inversion(point1, realk, point2)`

Example:

`inversion(GA, 3, GB)` draws point C on line AB such that AB*AC=3. In this case, point A is the center of the inversion and the scale factor is 3. Point B is the point whose inversion is created.

In general, the inversion of point A through center C, with scale factor k, maps A onto A', such that A' is on line CA and CA*CA'=k, where CA and CA' denote the lengths of the corresponding segments. If k=1, then the lengths CA and CA' are reciprocals.

Reciprocation

Given a circle and a vector of objects that are either points or lines, returns a vector where each point is replaced with its polar line and each line is replaced with its pole, with respect to the circle.

`reciprocation(Circle, [Obj1, Obj2,...Objn])`

Example:

`reciprocation(circle(0,1),[line(1+i,2),point(1+i*2)])` returns `[point(1/2, 1/2) line(y=-x/2+1/2)]`

Numeric view: Cmds menu

Cartesian

Abscissa

Returns the x coordinate of a point or the x length of a vector.

`abscissa(point) or abscissa(vector)`

Example:

`abscissa(GA)` returns the x-coordinate of the point A.

Ordinate

Returns the y coordinate of a point or the y length of a vector.

`ordinate(point)` or `ordinate(vector)`

Example:

`ordinate(GA)` returns the y-coordinate of the point A.

Coordinates

Given a vector of points, returns a matrix containing the x- and y-coordinates of those points. Each row of the matrix defines one point; the first column gives the x-coordinates and the second column contains the y-coordinates.

`coordinates([point1, point2, …, pointn]))`

Equation of

Returns the Cartesian equation of a curve in x and y, or the Cartesian coordinates of a point.

`equation(curve)` or `equation(point)`

Example:

If GA is the point at (0, 0), GB is the point at (1, 0), and GC is defined as circle(GA, GB-GA), then `equation(GC)` returns x2 + y2 = 1.

Parametric

Works like the **equation** command, but returns parametric results in complex form.

`parameq(GeoObj )`

Polar Coordinates

Returns a vector containing the polar coordinates of a point or a complex number.

`polar_coordinates(point)` or `polar_coordinates(complex)`

Example:

`polar_coordinates(√2, √2)` returns `[2, π/4]`

Measure

Distance

Returns the distance between two points or between a point and a curve.

`distance(point1, point2)` or `distance(point, curve)`

Examples:

`distance(1+i, 3+3i)` returns 2.828… or 2√2.

If GA is the point at (0, 0) and GB is defined as plotfunc(4–x^2/4), then `distance(GA, GB)` returns 3.464… or 2√3.

Radius

Returns the radius of a circle.

`radius(circle)`

Example:

If GA is the point at (0, 0), GB is the point at (1, 0), and GC is defined as circle(GA, GB-GA), then `radius(GC)` returns 1.

Perimeter

Returns the perimeter of a polygon or the circumference of a circle.

`perimeter(polygon)` or `perimeter(circle)`

Examples:

If GA is the point at (0, 0), GB is the point at (1, 0), and GC is defined as circle(GA, GB-GA), then `perimeter(GC)` returns 2p.

If GA is the point at (0, 0), GB is the point at (1, 0), and GC is defined as square(GA, GB-GA), then `perimeter(GC)` returns 4.

Slope

Returns the slope of a straight object (segment, ray, or line).

`slope(Object)`

Example:

`slope(line(point(1, 1), point(2, 2)))` returns 1.

Area

Returns the area of a circle or polygon.

`area(circle)` or `area(polygon)`

This command can also return the area under a curve between two points.

`area(expr, value1, value2)`

Examples:

If GA is defined to be the unit circle, then `area(GA)` returns p.

`area(4-x^2/4, -4,4)` returns 14.666…

Angle

Returns the measure of a directed angle. The first point is taken as the vertex of the angle as the next two points in order give the measure and sign.

`angle(vertex, point2, point3)`

Example:

`angle(GA, GB, GC)` returns the measure of ∡BAC.

Arc Length

Returns the length of the arc of a curve between two points on the curve. The curve is an expression, the independent variable is declared, and the two points are defined by values of the independent variable.

This command can also accept a parametric definition of a curve. In this case, the expression is a list of 2 expressions (the first for x and the second for y) in terms of a third independent variable.

`arcLen(expr, real1, real2)`

Examples:

`arcLen(x^2, x, -2, 2)` returns 9.29....

`arcLen({sin(t), cos(t)}, t, 0, π/2)` returns 1.57...

Tests

Collinear

Takes a set of points as argument and tests whether or not they are collinear. Returns 1 if the points are collinear and 0 otherwise.

`is_collinear(point1, point2, …, pointn)`

Example:

`is_collinear(point(0,0), point(5,0), point(6,1))` returns 0

On circle

Takes a set of points as argument and tests if they are all on the same circle. Returns 1 if the points are all on the same circle and 0 otherwise.

`is_concyclic(point1, point2, …, pointn)`

Example:

`is_concyclic(point(-4,-2), point(-4,2), point(4,-2), point(4,2))` returns 1

On object

Tests whether a point is on a geometric object. Returns a number (1 to n number of sides) representing the segment that contains the point if it is and 0 otherwise.

`is_element(point, object)`

Example:

`is_element(point(2/√2,2/√2), circle(0,1))` returns 1.

`is_element(point(0,-5), square(point(3,3),point(-5,3))` returns 3.

Parallel

Tests whether or not two lines are parallel. Returns 1 if they are and 0 otherwise.

`is_parallel(line1, line2)`

Example:

`is_parallel(line(2x+3y=7),line(2x+3y=9)` returns 1.

Perpendicular

Similar to **is_orthogonal**. Tests whether or not two lines are perpendicular.

`is_perpendicular(line1, line2)`

Isosceles

Takes three points and tests whether or not they are vertices of a single isosceles triangle. Returns 0 if they are not. If they are, returns the number order of the common point of the two sides of equal length (1, 2, or 3). Returns 4 if the three points form an equilateral triangle.

`is_isosceles(point1, point2, point3)`

Example:

`is_isoscelesl(point(0,0), point(4,0), point(2,4))` returns 3.

Equilateral

Takes three points and tests whether or not they are vertices of a single equilateral triangle. Returns 1 if they are and 0 otherwise.

`is_equilateral(point1, point2, point3)`

Example:

`is_equilateral(point(0,0), point(4,0), point(2,4))` returns 0.

Parallelogram

Tests whether or not a set of four points are vertices of a parallelogram. Returns 0 if they are not. If they are, then returns 1 if they form only a parallelogram, 2 if they form a rhombus, 3 if they form a rectangle, and 4 if they form a square.

`is_parallelogram(point1, point2, point3, point4)`

Example:

`is_parallelogram(point(0,0), point(2,4), point(0,8), point(-2,4))` returns 2.

Conjugate

Tests whether or not two points or two lines are conjugates for the given circle. Returns 1 if they are and 0 otherwise.

`is_conjugate(circle, point1, point2)` or `is_conjugate(circle, line1, line2)`

Other Geometry functions

The following functions are not available from a menu in the Geometry app, but are available from the Catlg menu.

affix

Returns the coordinates of a point or both the x- and y-lengths of a vector as a complex number.

`affix(point)` or `affix(vector)`

Example:

If GA is a point at (1, –2), then `affix(GA)` returns 1–2i.

barycenter

Calculates the hypothetical center of mass of a set of points, each with a given weight (a real number). Each point, weight pair is enclosed in square brackets as a vector.

`barycenter([[point1, weight1], [point2, weight2],…,[pointn, weightn]])`

Example:

$$\texttt{barycenter}\left(\begin{bmatrix} \text{point}(1) & 1 \\ \text{point}(1+i) & 2 \\ \text{point}(1-i) & 1 \end{bmatrix}\right) \text{ returns point (1, 1/4)}$$

convexhull

Returns a vector containing the points that serve as the convex hull for a given set of points.

`convexhull(point1, point2, …, pointn)`

Example:

`convexhull(0,1,1+i,1+2i,-1-i,1-3i,-2+i)` returns [1-3*i 1+2*i -2+ i -1- i]

distance2

Returns the square of the distance between two points or between a point and a curve.

`distance2(point1, point2)` or `distance2(point, curve)`

Examples:

`distance2(1+i, 3+3i)` returns 8.

If GA is the point at (0, 0) and GB is defined as plotfunc(4-x^2/4), then `distance2(GA, GB)` returns 12.

division_point

For two points A and B, and a numerical factor k, returns a point C such that C-B=k*(C-A).

`division_point(point1, point2, realk)`

Example:

`division_point(0,6+6*i,4)` returns point (8,8)

equilateral_triangle

Draws an equilateral triangle defined by one of its sides; that is, by two consecutive vertices. The third point is calculated automatically, but is not defined symbolically. If a lowercase variable is added as a third argument, then the coordinates of the third point are stored in that variable. The orientation of the triangle is counterclockwise from the first point.

`equilateral_triangle(point1, point2)` or `equilateral_triangle(point1, point2, var)`

Examples:

`equilateral triangle(0,6)` draws an equilateral triangle whose first two vertices are at (0, 0) and (6,0); the third vertex is calculated to be at (3,3*√3).

`equilateral triangle(0,6, v)` draws an equilateral triangle whose first two vertices are at (0, 0) and (6,0); the third vertex is calculated to be at (3,3*√3) and these coordinates are stored in the CAS variable v. In CAS view, entering v returns point(3*(√3*i+1)), which is equal to (3,3*√3).

exbisector

Given three points that define a triangle, creates the bisector of the exterior angles of the triangle whose common vertex is at the first point. The triangle does not have to be drawn in the Plot view.

`exbisector(point1, point2, point3)`

Examples:

`exbisector(A,B,C)` draws the bisector of the exterior angles of ΔABC whose common vertex is at point A.

`exbisector(0,-4i,4)` draws the line given by y=x.

extract_measure

Returns the definition of a geometric object. For a point, that definition consists of the coordinates of the point. For other objects, the definition mirrors their definition in Symbolic view, with the coordinates of their defining points supplied.

`extract_measure(Var)`

harmonic_conjugate

Returns the harmonic conjugate of 3 points. Specifically, returns the harmonic conjugate of point3 with respect to point1 and point2. Also accepts three parallel or concurrent lines; in this case, it returns the equation of the harmonic conjugate line.

`harmonic_conjugate(point1, point2, point3)` or `harmonic_conjugate(line1, line2, line3)`

Example:

`harmonic_conjugate(point(0, 0), point(3, 0), point(4, 0))` returns point(12/5, 0)

harmonic_division

Returns the harmonic conjugate of 3 points. Specifically, returns the harmonic conjugate of point3 with respect to point1 and point2 and stores the result in the variable var. Also accepts three parallel or concurrent lines; in this case, it returns the equation of the harmonic conjugate line.

`harmonic_division(point1, point2, point3, var)` or `harmonic_division(line1, line2, line3, var)`

Example:

`harmonic_division(point(0, 0), point(3, 0), point(4, 0), p)` returns point(12/5, 0) and stores it in the variable p

isobarycenter

Returns the hypothetical center of mass of a set of points. Works like barycenter but assumes that all points have equal weight.

```
isobarycenter(point1, point2, …,pointn)
```

Example:

`isobarycenter(-3,3,3*√3*i)` returns point(3*√3*i/3), which is equivalent to (0,√3).

is_harmonic

Tests whether or not 4 points are in a harmonic division or range. Returns 1 if they are or 0 otherwise.

```
is_harmonic(point1, point2, point3, point4)
```

Example:

`is_harmonic(point(0, 0), point(3, 0), point(4, 0), point(12/5, 0))` returns 1

is_harmonic_circle_bundle

Returns 1 if the circles build a beam, 2 if they have the same center, 3 if they are the same circle and 0 otherwise.

```
is_harmonic_circle_bundle({circle1, circle2, …, circlen})
```

is_harmonic_line_bundle

Returns 1 if the lines are concurrent, 2 if they are all parallel, 3 if they are the same line and 0 otherwise.

```
is_harmonic_line_bundle({line1, line2, …, linen}))
```

is_orthogonal

Tests whether or not two lines or two circles are orthogonal (perpendicular). In the case of two circles, tests whether or not the tangent lines at a point of intersection are orthogonal. Returns 1 if they are and 0 otherwise.

`is_orthogonal(line1, line2)` or `is_orthogonal(circle1, circle2)`

Example:

`is_orthogonal(line(y=x),line(y=-x))` returns 1.

is_rectangle

Tests whether or not a set of four points are vertices of a rectangle. Returns 0 if they are not, 1 if they are, and 2 if they are vertices of a square.

```
is_rectangle(point1, point2, point3, point4)
```

Examples:

`is_rectangle(point(0,0), point(4,2), point(2,6), point(-2,4))` returns 2.

With a set of only three points as argument, tests whether or not they are vertices of a right triangle. Returns 0 if they are not. If they are, returns the number order of the common point of the two perpendicular sides (1, 2, or 3).

`is_rectangle(point(0,0), point(4,2), point(2,6))` returns 2.

is_rhombus

Tests whether or not a set of four points are vertices of a rhombus. Returns 0 if they are not, 1 if they are, and 2 if they are vertices of a square.

`is_rhombus(point1, point2, point3, point4)`

Example:

`is_rhombus(point(0,0), point(-2,2), point(0,4), point(2,2))` returns 2

is_square

Tests whether or not a set of four points are vertices of a square. Returns 1 if they are and 0 otherwise.

`is_square(point1, point2, point3, point4)`

Example:

`is_square(point(0,0), point(4,2), point(2,6), point(-2,4))` returns 1.

LineHorz

Draws the horizontal line y=a.

`LineHorz(a)`

Example:

`LineHorz(-2)` draws the horizontal line whose equation is y = –2

LineVert

Draws the vertical line x=a.

`LineVert(a)`

Example:

`LineVert(-3)` draws the vertical line whose equation is x = –3

open_polygon

Connects a set of points with line segments, in the given order, to produce a polygon. If the last point is the same as the first point, then the polygon is closed; otherwise, it is open.

`open_polygon(point1, point2, …, point1)` or `open_polygon(point1, point2, …, pointn)`

orthocenter

Returns the orthocenter of a triangle; that is, the intersection of the three altitudes of a triangle. The argument can be either the name of a triangle or three non-collinear points that define a triangle. In the latter case, the triangle does not need to be drawn.

`orthocenter(triangle)` or `orthocenter(point1, point2, point3)`

Example:

`orthocenter(0,4i,4)` returns (0,0)

perpendicular bisector

Draws the perpendicular bisector of a segment. The segment is defined either by its name or by its two endpoints.

`perpen_bisector(segment)` or `perpen_bisector(point1, point2)`

Examples:

`perpen_bisector(GC)` draws the perpendicular bisector of segment C.

`perpen_bisector(GA, GB)` draws the perpendicular bisector of segment AB.

`perpen_bisector(3+2i, i)` draws the perpendicular bisector of a segment whose endpoints have coordinates (3, 2) and (0, 1); that is, the line whose equation is y=x/3+1.

point2d

Randomly redistributes a set of points such that, for each point, x ∈ [–5,5] and y ∈ [–5,5]. Any further movement of one of the points will randomly redistribute all of the points with each tap or direction key press.

`point2d(point1, point2, …, pointn)`

polar

Returns the polar line of the given point as pole with respect to the given circle.

`polar(circle, point)`

Example:

`polar(circle(x^2+y^2=1),point(1/3,0))` returns x=3

pole

Returns the pole of the given line with respect to the given circle.

`pole(circle, line)`

Example:

`pole(circle(x^2+y^2=1), line(x=3))` returns point(1/3, 0)

power_pc

Given a circle and a point, returns the difference between the square of the distance from the point to the circle's center and the square of the circle's radius.

`powerpc(circle, point)`

Example:

`powerpc(circle(point(0,0), point(1,1)-point(0,0)), point(3,1))` returns 8

radical_axis

Returns the line whose points all have the same powerpc values for the two given circles.

radical_axis(circle1, circle2)

Example:

radical_axis(circle(((x+2)^2+y^2) = 8),circle(((x−2)^2+y^2) = 8)) returns line(x=0)

vector

Creates a vector from point1 to point2. With one point as argument, the origin is used as the tail of the vector.

vector(point1, point2) or vector(point)

Example:

vector(point(1,1), point(3,0)) creates a vector from (1, 1) to (3, 0).

vertices

Returns a list of the vertices of a polygon.

vertices(polygon)

vertices_abca

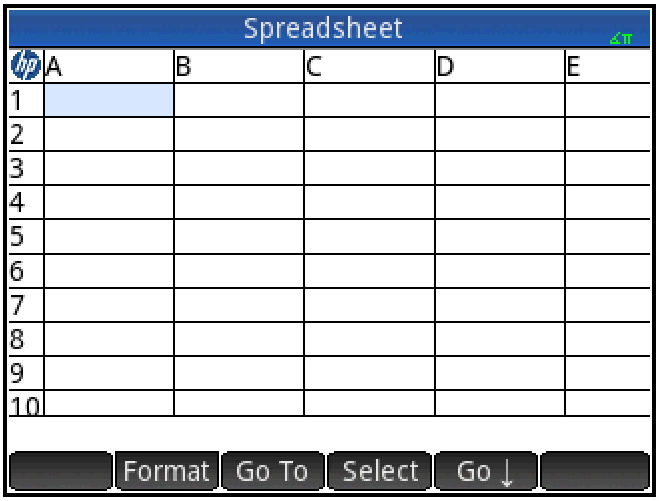
Returns the closed list of the vertices of a polygon.

vertices_abca(polygon)

10 Spreadsheet

The Spreadsheet app provides a grid of cells for you to enter content (such as numbers, text, expressions, and so on) and to perform certain operations on what you enter.

To open the Spreadsheet app, press Apps Info and select **Spreadsheet**.



	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

You can create any number of customized spreadsheets, each with its own name, much like creating an app. You open a customized spreadsheet in the same way: by pressing Apps Info and selecting the particular spreadsheet.

The maximum size of any one spreadsheet is 10,000 rows by 676 columns.

The app opens in Numeric view. There is no Plot or Symbolic view. There is a Symbolic Setup view (Shift Symb Setup) that enables you to override certain system-wide settings. (This is a common Symbolic Setup view operation.)

Getting started with the Spreadsheet app

Suppose you have a stall at a weekend market. You sell furniture on consignment for their owners, taking a 10% commission for yourself. You have to pay the landowner $100 a day to set up your stall and you will keep the stall open until you have made $250 for yourself.

1. Open the Spreadsheet app.

   Press Apps Info and select **Spreadsheet**.

2. Select column A. Either tap **A** or use the cursor keys to highlight the A cell (that is, the heading of the A column).

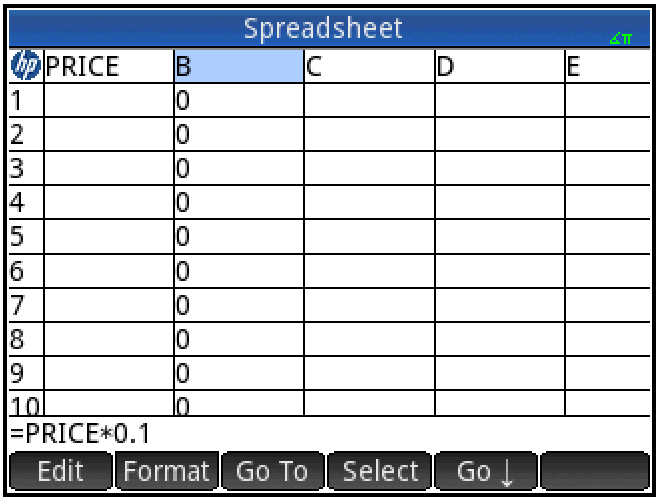
3. Enter `PRICE` and tap Name to name the entire first column PRICE.

4. Select column B. Either tap **B** or use the cursor keys to highlight the B cell.

5. Enter a formula for your commission (being 10% of the price of each item sold):

   Shift • = PRICE × 0.1 Enter

   Because you entered the formula in the heading of a column, it is automatically copied to every cell in that column. At the moment only 0 is shown, since there are no values in the PRICE column yet.



Spreadsheet

	PRICE	B	C	D	E
1		0			
2		0			
3		0			
4		0			
5		0			
6		0			
7		0			
8		0			
9		0			
10		0			

=PRICE*0.1

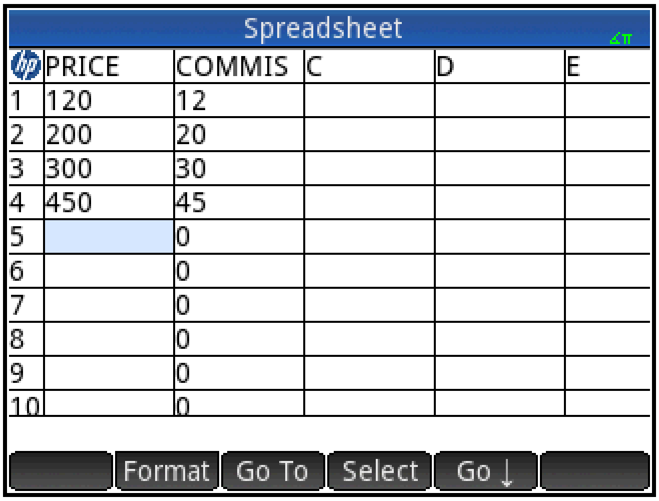
6. Select column B.

7. Tap Format and select **Name**.

8. Type `COMMIS` and tap OK. The heading of column B is now COMMIS.

9. It is always a good idea to check your formulas by entering some dummy values and noting whether the result is as expected. Select cell A1 and make sure that Go ↓ and not Go → is showing in the menu. (If not, tap the button.) This option means that your cursor automatically selects the cell immediately following the one you have just entered content into.

10. Add some values in the **PRICE** column and note the result in the **COMMIS** column. If the results do not look right, you can tap the **COMMIS** heading, tap Edit, and revise the formula.



Spreadsheet

	PRICE	COMMIS	C	D	E
1	120	12			
2	200	20			
3	300	30			
4	450	45			
5		0			
6		0			
7		0			
8		0			
9		0			
10		0			

11. To delete the dummy values, select cell **A1**, tap Select, press ▼ until all the dummy values are selected, and then press Del.

12. Select cell **C1**.

13. Enter a label for your takings as follows:

    Shift . ALPHA 0 TAKINGS Enter

NOTE: Text strings, but not names, need to be enclosed within quotation marks.

14. Select cell **D1**.

15. Enter a formula to add up your takings as follows:

    Shift . SUM ( ) PRICE Enter

    You can specify a range—such as `A1:A100`—but by specifying the name of the column, you can be sure that the sum includes all the entries in the column.

16. Select cell **C3**.

17. Enter a label for your total commission:

    Shift . ALPHA 0 TOTAL COMMIS Enter

18. To widen column C to see the entire label in C3, select the heading cell for column C, tap Format, and select Column ↔.

    An input form appears for you to specify the required width of the column.

19. Enter `100` and tap ↔.

    You might have to experiment until you get the column width exactly as you want it. The value you enter is the width of the column in pixels.

20. Select cell **D3**.

21. Enter a formula to add up your commission:

    Shift . SUM ( ) COMMIS Enter

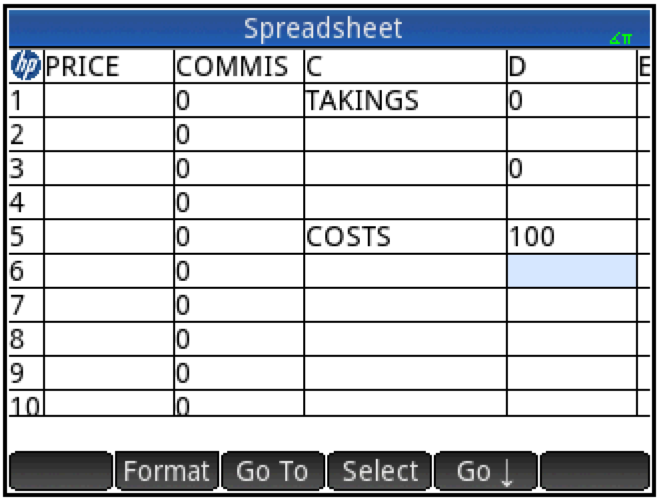
TIP: Instead of entering `SUM` by hand, you can choose it from the **Apps** menu (one of the Toolbox menus).

22. Select cell **C5**.

23. Enter a label for your fixed costs:

    Shift . ALPHA 0 COSTS Enter

24. In cell **D5**, enter `100`. This is what you have to pay the landowner for renting the space for your stall.



	PRICE	COMMIS	C	D	E
1		0	TAKINGS	0	
2		0			
3		0		0	
4		0			
5		0	COSTS	100	
6		0			
7		0			
8		0			
9		0			
10		0			

25. Enter the label `PROFIT` in cell **C7**.

26. In cell **D7**, enter a formula to calculate your profit:

Shift • D3 − D5 Enter

You can also name D3 and D5. For example, TOTCOM and COSTS respectively. Then, the formula in D7 is =TOTCOM–COSTS.

27. Enter the label `GOAL` in cell **E1**.

You can swipe the screen with a finger, or repeatedly press the cursor keys, to bring **E1** into view.

28. Enter `250` in cell **F1**.

This is the minimum profit you want to make on the day.

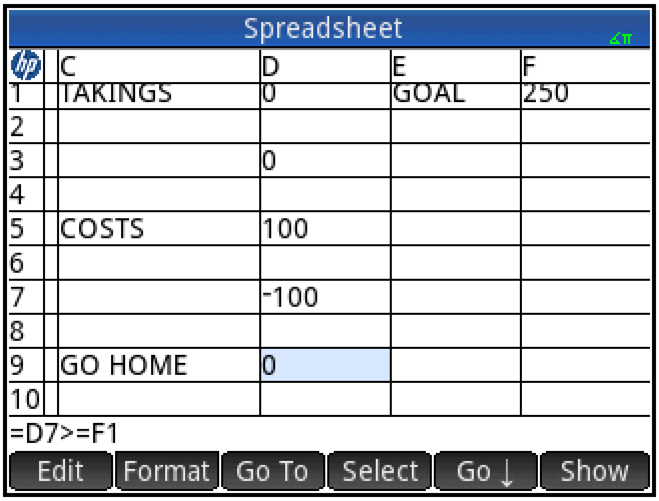
29. In cell **C9**, enter the label `GO HOME`.

30. In cell **D9**, enter the following formula:

Shift . = D7≥F1 Enter ≈

You can select ≥ from the relations palette (Shift 6 ≤,≥,≠ w).

This formula displays **0** in **D9** if you have not reached your goal profit, and **1** if you have. It provides a quick way for you to see when you have made enough profit and can go home.



Spreadsheet

	C	D	E	F
1	TAKINGS	0	GOAL	250
2				
3		0		
4				
5	COSTS	100		
6				
7		-100		
8				
9	GO HOME	0		
10				

=D7>=F1

31. Select **C9** and **D9**.

You can select both cells with a finger drag, or by highlighting **C9**, selecting Select, and pressing ▶.

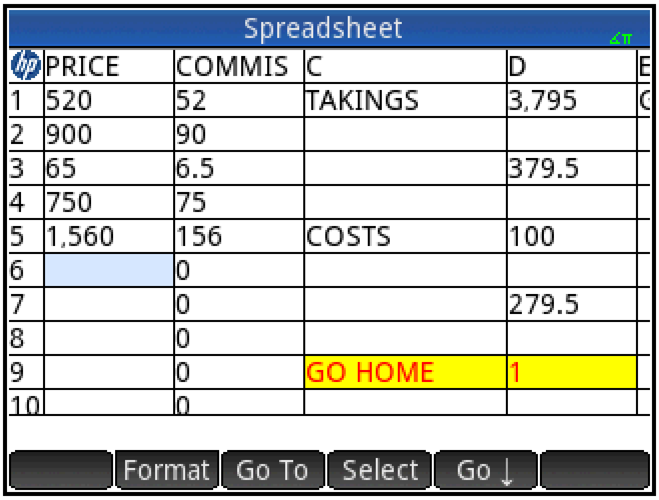
32. Tap Format and select **Color**.

33. Choose a color for the contents of the selected cells.

34. Tap Format and select **Fill**.

35. Choose a color for the background of the selected cells.

The most important cells in the spreadsheet will now stand out from the rest.



	PRICE	COMMIS	C	D	E
1	520	52	TAKINGS	3,795	C
2	900	90			
3	65	6.5		379.5	
4	750	75			
5	1,560	156	COSTS	100	
6		0			
7		0		279.5	
8		0			
9		0	GO HOME	1	
10		0			

The spreadsheet is complete, but you may want to check all the formulas by adding some dummy data to the **PRICE** column. When the profit reaches 250, you should see the value in **D9** change from **0** to **1**.

Basic operations

Navigation, selection, and gestures

You can move about a spreadsheet by using the cursor keys, by swiping, or by tapping Go To and specifying the cell you want to move to.

You select a cell simply by moving to it. You can also select an entire column—by tapping the column letter—and select an entire row (by tapping the row number). You can also select the entire spreadsheet: just tap on the unnumbered cell at the top-left corner of the spreadsheet. (It has the HP logo in it.)

A block of cells can be selected by pressing down on a cell that will be a corner cell of the selection and, after a second, dragging your finger to the diagonally opposite cell. You can also select a block of cells by moving to a corner cell, tapping Select, and using the cursor keys to move to the diagonally opposite cell. Tapping Sel• or another cell deselects the selection.

Cell references

You can refer to the value of a cell in formulas as if it were a variable. A cell is referenced by its column and row coordinates, and references can be absolute or relative. An absolute reference is written as $C$R (where C is the column number and R the row number). Thus $B$7 is an absolute reference. In a formula it will always refer to the data in cell B7 wherever that formula, or a copy of it, is placed. On the other hand, B7 is a relative reference. It is based on the relative position of cells. Thus a formula in, say, B8 that references B7 will reference C7 instead of B7 if it is copied to C8.

Ranges of cells can also be specified, as in C6:E12, as can entire columns (E:E) or entire rows ($3:$5). Note that the alphabetic component of column names can be uppercase or lowercase except for columns g, l, m, and z. (G, L, M, and Z are names reserved for graphic objects, lists, matrices, and complex numbers.) These must be in lowercase if not preceded by $. Thus cell B1 can be referred to as B1,b1,$B$1 or $b$1 whereas M1 can only be referred to as m1, $m$1, or $M$1.

Cell naming

Cells, rows, and columns can be named. The name can then be used in a formula. A named cell is given a blue border.

Method 1

To name an empty cell, row, or column, go the cell, row header, or column header, enter a name and tap Name.

Method 2

To name a cell, row, or column—whether it is empty or not:

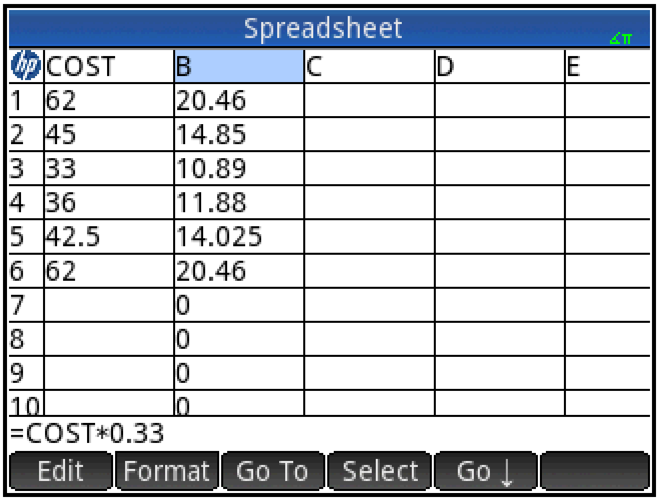
1. Select the cell, row, or column.
2. Tap Format and select **Name**.
3. Enter a name and tap OK.

Using names in calculations

The name you give a cell, row, or column can be used in a formula. For example, if you name a cell **TOTAL**, you could enter in another cell the formula `=TOTAL*1.1`.

The following is a more complex example involving the naming of an entire column.

1. Select cell **A** (that is the header cell for column A).
2. Enter `COST` and tap Name.
3. Select cell **B** (that is the header cell for column B).
4. Enter Shift = COST*0.33, and then tap OK.
5. Enter some values in column **A** and observe the calculated results in column **B**.



Spreadsheet

	COST	B	C	D	E
1	62	20.46			
2	45	14.85			
3	33	10.89			
4	36	11.88			
5	42.5	14.025			
6	62	20.46			
7		0			
8		0			
9		0			
10		0			

=COST*0.33

Entering content

You can enter content directly in the spreadsheet or import data from a statistics app.

Direct entry

A cell can contain any valid calculator object: a real number (3.14), a complex number (a + ib), an integer (#1Ah), a list ({1, 2}), a matrix or vector([1, 2]), a string ("text"), a unit (2_m) or an expression (that is, a formula). Move to the cell you want to add content to and start entering the content as you would in Home view. Press Enter when you have finished. You can also enter content into a number of cells with a single entry. Just select the cells, enter the content—for example, `=Row*3`—and press Enter.

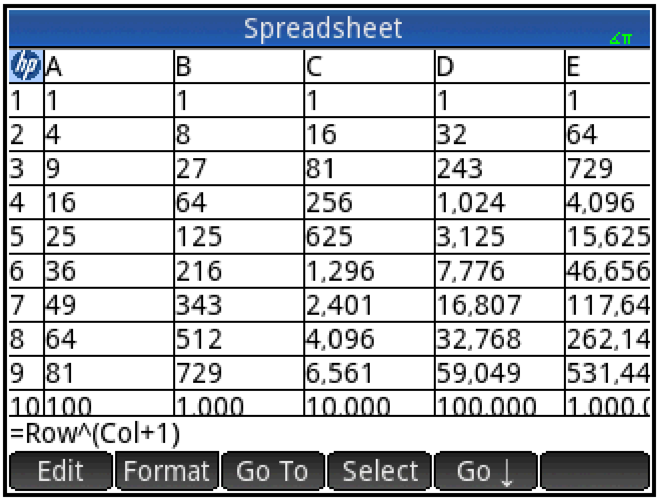
What you enter on the entry line is evaluated as soon as you press Enter, with the result placed in the cell or cells. However, if you want to retain the underlying formula, precede it with Shift ≐. For example, suppose that you want to add cell A1 (which contains 7) to cell B2 (which contains 12). Entering A1 + B2 Enter in, say, A4 yields19, as does entering Shift ≐ A1 + B2 in A5. However, if the value in A1 (or B2) changes, the value in A5 changes but not the value in A4. This is because the expression (or formula) was retained in A5. To see if a cell contains just the value shown in it or also an underlying formula that generates the value, move your cursor to the cell. The entry line shows a formula if there is one.

A single formula can add content to every cell in a column or row. For example, move to C (the heading cell for column C), enter Shift ≐ SIN(Row) and press Enter. Each cell in the column is populated with the sine of the cell’s row number. A similar process enables you to populate every cell in a row with the same formula. You can also add a formula once and have it apply to every cell in the spreadsheet. You do this by placing the formula in the cell at the top left (the cell with the HP logo in it). To see how this works, suppose you want to generate a table of powers (squares, cubes, and so on) starting with the squares:

1. Tap on the cell with the HP logo in it (at the top left corner). Alternatively, you can use the cursor keys to move to that cell (just as you can to select a column or row heading).

2. On the entry line, type Shift = Row x^y Col + 1

   Note that Row and Col are built-in variables. They are placeholders for the row number and column number of the cell that has a formula containing them.



	A	B	C	D	E
1	1	1	1	1	1
2	4	8	16	32	64
3	9	27	81	243	729
4	16	64	256	1,024	4,096
5	25	125	625	3,125	15,625
6	36	216	1,296	7,776	46,656
7	49	343	2,401	16,807	117,64
8	64	512	4,096	32,768	262,14
9	81	729	6,561	59,049	531,44
10	100	1,000	10,000	100,000	1,000,(

=Row^(Col+1)

3. Tap OK or press Enter.

   Note that each column gives the nth power of the row number starting with the squares. Thus 9^5 is 59,049.

Importing data

You can import data from the Statistics 1Var and Statistics 2Var apps (and from any app customized from a statistics app). In the procedure immediately below, dataset D1 from the Statistics 1Var app is being imported.

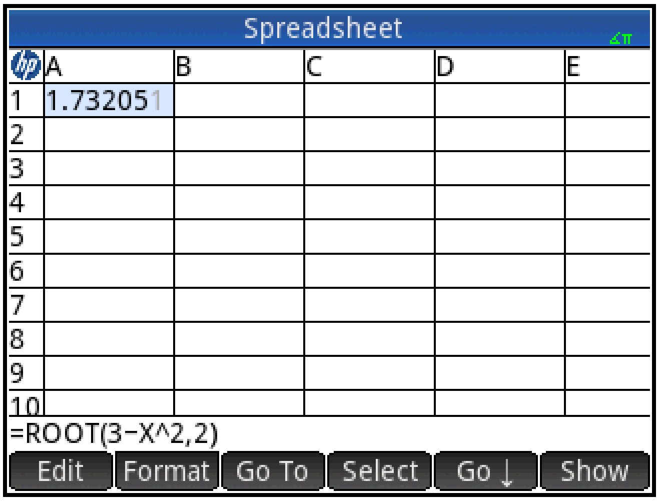
1. Select a cell.
2. Enter `Statistics_1Var.D1`.
3. Press Enter.

The column is filled with the data from the statistics app, starting with the cell selected at step 1. Any data in that column will be overwritten by the data being imported.

You can also export data from the Spreadsheet app to a statistics app, using the entering and editing statistical data procedure. This procedure can also be used in both the Statistics 1Var and Statistics 2Var apps.

External functions

You can use in a formula any function available on the Math, CAS, App, User or Catlg menus. For example, to find the root of $3 – x^2$ closest to x = 2, you could enter the following in a cell: Shift = ALPHA ALPHA ROOT ALPHA ( ) 3 − ALPHA × x^2 , 2 Enter. The answer displayed is 1.732.



	A	B	C	D	E
1	1.732051				
2					
3					
4					
5					
6					
7					
8					
9					
10					

=ROOT(3−X^2,2)

You can also select a function from a menu. For example, see the following procedure:

1. Press Shift .= .

2. Press 🧰 Mem B and tap CAS.

3. Select **Polynomial > Find Roots**.

   Your entry line now looks like this: **=CAS.proot()**.

4. Enter the coefficients of the polynomial, in descending order, separating each with a comma:

   +/− 1 , 0 , 3

5. Press Enter to see the result. Select the cell and tap Show to see a vector containing both roots: [1.732... –1.732...].

6. Tap OK to return to the spreadsheet.

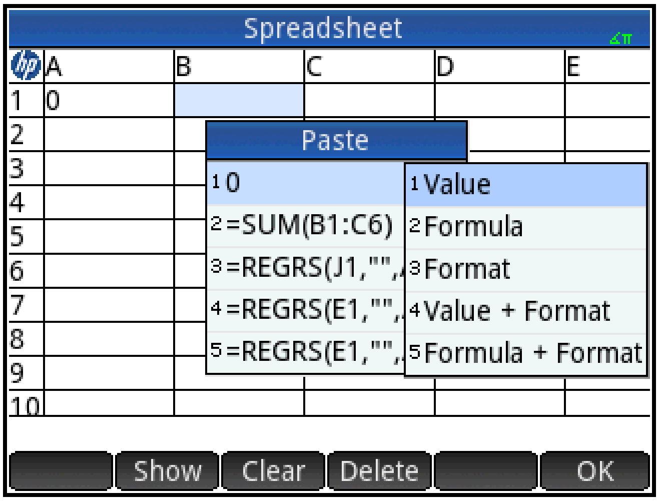
Note that the CAS prefix added to your function is to remind you that the calculation will be carried out by the CAS (and thus a symbolic result will be returned, if possible). You can also force a calculation to be handled by the CAS by tapping CAS in the spreadsheet.

There are additional spreadsheet functions that you can use (mostly related to finance and statistics calculations).

Copy and paste

1. To copy one or more cells, select them and press Shift View Copy .

2. Move to the desired location and press Shift Menu Paste .



You can choose to paste either the value, formula, format, both value and format, or both formula and format.

You can also copy data from the Spreadsheet app and paste it into the Statistics apps, the List Editor, or the Matrix Editor. Or, you can copy from one of those apps and paste into the Spreadsheet app. In these cases, only the values are pasted.

Using the CHOOSE command

The CHOOSE command defines a cell as a drop-down box in the spreadsheet. The name of a cell is used as the variable name.

For example, if you enter the command `=CHOOSE($B$1, "Favorite Color", {"Red", "Green", "Yellow", "Blue"})` in cell A1, cell A1 becomes a drop-down box. Tap this cell to open a list titled Favorite Color with entries Red, Green, Yellow, and Blue. If you tap Blue, then cell B1 contains the value 4 because Blue is the fourth entry. If you enter `2` in cell B1, the value selected in cell A1 changes to Green, because Green is the second entry.

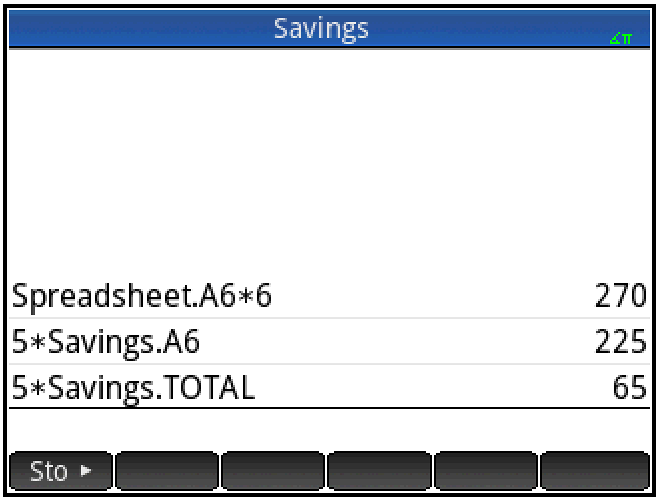
External references

You can refer to data in a spreadsheet from outside the Spreadsheet app by using the reference **SpreadsheetName.CR**. For example, in Home view you can refer to cell A6 in the built-in spreadsheet by entering `Spreadsheet.A6`. Thus the formula 6*Spreadsheet.A6 would multiply whatever value is currently in cell A6 in the built-in app by 6.

If you have created a customized spreadsheet called, say, Savings, you simply refer to it by its name, as in 5*Savings.A6.

An external reference can also be to a named cell, as in 5*Savings.TOTAL.

In the same way, you can also enter references to spreadsheet cells in the CAS.



Savings

```
Spreadsheet.A6*6              270
5*Savings.A6                  225
5*Savings.TOTAL                65
```

If you are working outside a spreadsheet, you can refer to a cell by its absolute reference. Thus, entering `Spreadsheet.$A$6` returns the contents of cell A6 in the Spreadsheet app.

NOTE: A reference to a spreadsheet name is case sensitive.

Referencing variables

Any variable can be inserted in a cell. This includes Home variables, App variables, CAS variables and user variables.

Variables can be referenced or entered. For example, if you have assigned 10 to P in Home view, you could enter `=P*5` in a spreadsheet cell, press Enter and get 50. If you subsequently change the value of P, the value in that cell automatically changes to reflect the new value. This is an example of a referenced variable.

If you just wanted the current value of P and not have the value change if P changes, just enter `P` and press Enter. This is an example of an entered variable.

Variables given values in other apps can also be referenced in a spreadsheet. The Solve app can be used to solve equations. An example used is $V^2 = U^2 + 2AD$. You could have four cells in a spreadsheet with =V, =U, =A, and =D as formulas. As you experiment with different values for these variables in the Solve app, the entered and the calculated values are copied to the spreadsheet (where further manipulation can be done).

The variables from other apps include the results of certain calculations. For example, if you have plotted a function in the Function app and calculated the signed area between two x-values, you can reference that value in a spreadsheet by pressing Vars, tapping App, and then selecting **Function > Results > SignedArea**.

Numerous system variables are also available. For example, you could enter Shift + Enter to get the last answer calculated in Home view. You could also enter Shift . Shift + Enter to get the last answer calculated in Home view and have the value automatically updated as new calculations are made in Home view. (Note that this works only with the Ans from Home view, not the Ans from CAS view.)

All the variables available to you are listed on the variables menus, displayed by pressing Vars Chars A .

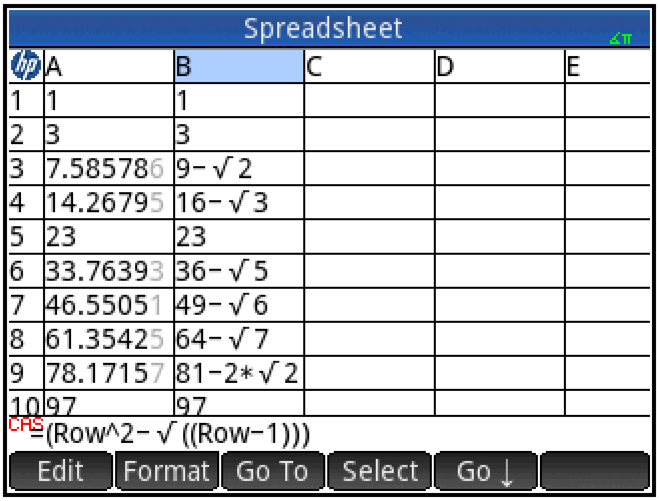
Using the CAS in spreadsheet calculations

You can force a spreadsheet calculation to be performed by the CAS, thereby ensuring that results are symbolic (and thus exact). For example, the formula =√Row in row 5 gives 2.2360679775 if not calculated by the CAS, and √5 if it is.

You choose the calculation engine when you are entering the formula. As soon as you begin entering a formula, the Format button changes to CAS or CAS• (depending on the last selection). This is a toggle key. Tap it to change it from one to the other.

When CAS is showing, the calculation will be numeric (with the number of significant digits limited by the precision of the calculator). When CAS• is showing, the calculation will be performed by CAS and be exact.

In the following figure, the formula in cell A is exactly the same as the formula in cell B: = Row2–√(Row–1). The only difference is that CAS• was showing (or selected) while the formula was being entered in B, thereby forcing the calculation to be performed by the CAS. Note that CAS appears in red on the entry line if the cell selected contains a formula that is being calculated by the CAS.



Spreadsheet

	A	B	C	D	E
1	1	1			
2	3	3			
3	7.585786	9−√2			
4	14.26795	16−√3			
5	23	23			
6	33.76393	36−√5			
7	46.55051	49−√6			
8	61.35425	64−√7			
9	78.17157	81−2*√2			
10	97	97			

CAS =(Row^2−√((Row−1)))

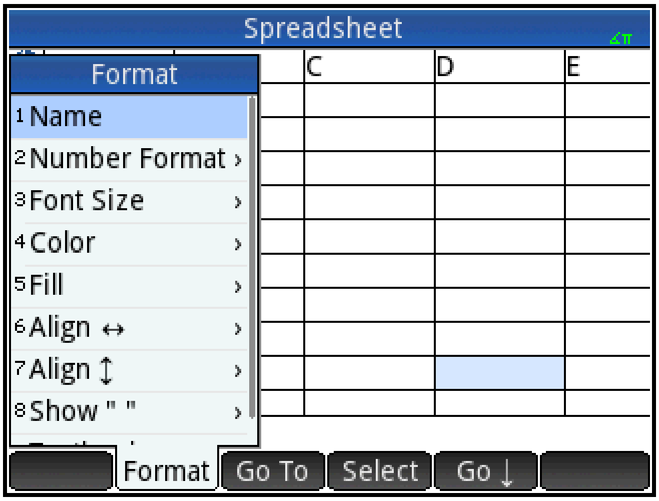
Buttons and keys

Button or key	Purpose
Edit	Activates the entry line for you to edit the object in the selected cell. This button is visible only if the selected cell has content.
Name	Converts the text you have entered on the entry line to a name. This button is visible only when the entry line is active.
CAS / CAS•	Toggles between options that force the expression to be handled by the CAS; however, only CAS evaluates it. This button is visible only when the entry line is active.
$	Enters the $ symbol. This button is a shortcut when entering absolute references and is visible only when the entry line is active.

Button or key	Purpose
Format	Displays formatting options for the selected cell, block, column, row, or the entire spreadsheet. See Formatting options on page 224.
Go To	Displays an input form for you to specify the cell you want to jump to.
Select	Sets the calculator to select mode so that you can easily select a block of cells using the cursor keys. It changes to Sel• to enable you to deselect cells. You can also press, hold, and drag to select a block of cells.
Go ↓ or Go →	Sets the direction the cursor moves after content has been entered in a cell.
Show	Displays the result in the selected cell in full-screen mode, with horizontal and vertical scrolling enabled. Only visible if the selected cell has content.
Sort	Enables you to select a column to sort by, and to sort it in ascending or descending order. Only visible if cells are selected.
Cancel	Cancel the input and clear the entry line.
OK	Accept and evaluate the input.
Shift Esc Clear	Clears the spreadsheet.

Formatting options

The formatting options appear when you tap Format. They apply to whatever is currently selected: a cell, block, column, row, or the entire spreadsheet.



The options are as follows:

- **Name**—Displays an input form for you to give a name to whatever is selected.
- **Number Format**—Auto, Standard, Fixed, Scientific, or Engineering. (This is similar to the settings in Home Settings.)
- **Font Size**—Auto or from 10 to 22 point.

- **Color**—Color for the content (text, number, etc.) in the selected cells; the gray-dotted option represents Auto.
- **Fill**—Background color that fills the selected cells; the gray-dotted option represents Auto.
- **Align** ↔—Auto, Left, Center, or Right horizontal alignment.
- **Align** ↕—Auto, Top, Center, or Bottom vertical alignment.
- **Column** ↔—Displays an input form for you to specify the required width of the selected columns; available only if you have selected the entire spreadsheet or one or more entire columns.

  You can also change the width of a selected column with an open or closed horizontal pinch gesture.
- **Row** ↕—Displays an input form for you to specify the required height of the selected rows; available only if you have selected the entire spreadsheet or one or more entire rows.

  You can also change the height of a selected row with an open or closed vertical pinch gesture.
- **show " "**—Show quote marks around strings in the body of the spreadsheet. Options are Auto, Yes, or No.
- **Textbook**—Display formulas in textbook format. Options are Auto, Yes, or No.
- **Caching**—Turn this option on to speed up calculations in spreadsheets with many formulas; available only if you have selected the entire spreadsheet.

Format parameters

Each format attribute is represented by a parameter that can be referenced in a formula. For example, =D1(1) returns the formula in cell D1 (or nothing if D1 has no formula). The attributes that can be retrieved in a formula by referencing its associated parameter are listed below.

Parameter	Attribute	Result
0	content	Contents (or empty)
1	formula	Formula
2	name	Name (or empty)
3	number format	Standard—0 Fixed—1 Scientific—2 Engineering—3
4	number of decimal places	1 to 11, or unspecified (–1)
5	font	0 to 6, or unspecified (-1) 0 equals 10 point and 6 equals 22 point
6	background color	Cell fill color, or 32768 if unspecified
7	foreground color	Cell contents color, or 32768 if unspecified
8	horizontal alignment	Left—0 Center—1 Right—2

Parameter	Attribute	Result
		Unspecified—–1
9	vertical alignment	Top—0 Center—1 Bottom—2 Unspecified—–1
10	show strings in quotes	Yes—0 No—1 Unspecified—–1
11	textbook mode (as opposed to algebraic mode)	Yes—0 No—1 Unspecified—–1

In addition to retrieving format attributes, you can set a format attribute (or cell content) by specifying it in a formula in the relevant cell. For example, wherever it is placed g5(1):=6543 enters 6543 in cell g5. Any previous content in g5 is replaced. Similarly, entering `B3(5):=2` forces the contents of B3 to be displayed in medium font size.

Spreadsheet functions

In addition to the functions on the **Math**, **CAS**, and **Catlg** menus, you can use special spreadsheet functions. These can be found on the **App** menu, one of the Toolbox menus. Press Mem, tap App, and select **Spreadsheet**.

Remember to precede a function by an equal sign (Shift =) if you want the result to automatically update as its dependent values change. Without an equal sign you are entering just the current value.

11 Statistics 1Var app

The Statistics 1Var app can store up to ten data sets at one time. It can perform one-variable statistical analysis of one or more sets of data.

The Statistics 1Var app starts with the Numeric view which is used to enter data. The Symbolic view is used to specify which columns contain data and which column contains frequencies.

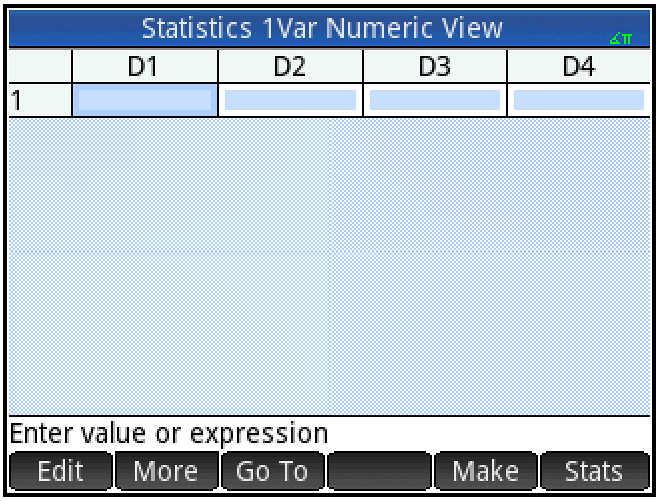
You can also compute statistics in Home and recall the values of specific statistics variables.

The values computed in the Statistics 1Var app are saved in variables, and can be re-used in Home view and in other apps.

Getting started with the Statistics 1Var app

Suppose that you are measuring the heights of students in a classroom to find the mean height. The first five students have the following measurements: 160 cm, 165 cm, 170 cm, 175 cm and180 cm.

1. Press Apps Info , and the select Statistics 1Var to open the Statistics 1Var app.



Statistics 1Var Numeric View

	D1	D2	D3	D4
1				

Enter value or expression

2. Enter the measurement data in column D1:

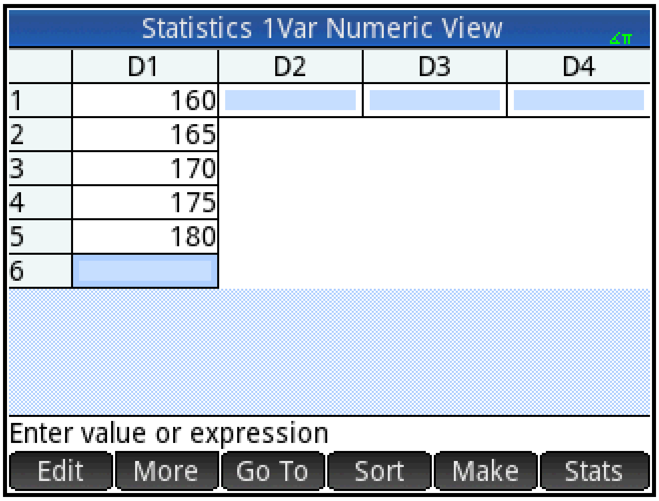
160 Enter

165 Enter

170 Enter

175 Enter

180 Enter



Statistics 1Var Numeric View

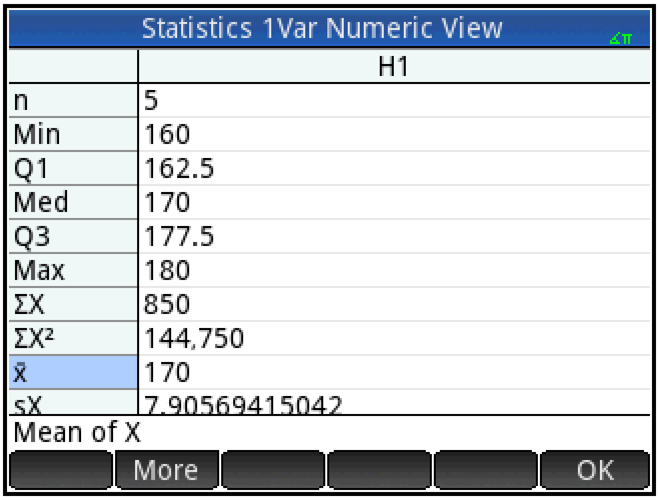
	D1	D2	D3	D4
1	160			
2	165			
3	170			
4	175			
5	180			
6				

Enter value or expression

3. Find the mean of the sample.

Tap Stats to see the statistics calculated from the sample data in D1. The mean (ẋ) is 170. There are more statistics than can be displayed on one screen. Thus you may need to scroll to see the statistic you are after.

Note that the title of the column of statistics is H1. There are 5 data-set definitions available for one-variable statistics: H1–H5. If data is entered in D1, H1 is automatically set to use D1 for data, and the frequency of each data point is set to 1. You can select other columns of data from the Symbolic view of the app.



Statistics 1Var Numeric View

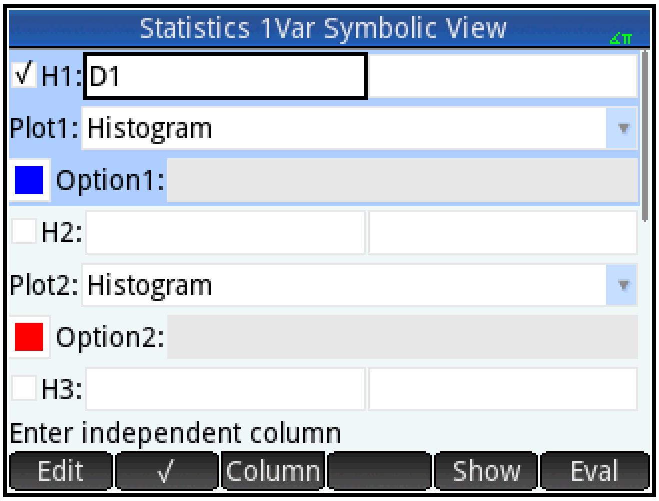
	H1
n	5
Min	160
Q1	162.5
Med	170
Q3	177.5
Max	180
ΣX	850
ΣX²	144,750
x̄	170
sX	7.90569415042

Mean of X

4. Tap OK to close the statistics window.

5. Press Symb Setup to see the data-set definitions.

The first field in each set of definitions is where you specify the column of data that is to be analyzed, the second field is where you specify the column that has the frequencies of each data point, and the third field (Plotn) is where you choose the type of plot that will represent the data in Plot view: Histogram, Box and whisker, Normal probability, Line, Bar, Pareto, Control, Dot, Stem and leaf, or Pie chart.



Statistics 1Var Symbolic View

√ H1: D1

Plot1: Histogram

🟦 Option1:

□ H2:

Plot2: Histogram

🟥 Option2:

□ H3:

Enter independent column

Symbolic view: menu items

The menu items you can tap in Symbolic view are as follows:

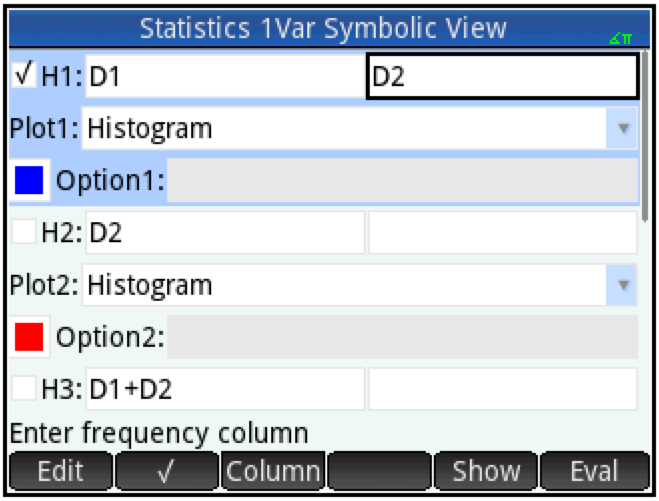
Menu item	Purpose
Edit	Copies the column variable (or variable expression) to the entry line for editing. Tap OK when done.
√	Selects (or deselects) a statistical analysis (H1–H5) for exploration.
Column	Selects the name of a column from Numeric view.
Show	Displays the current expression in textbook format in full-screen view. Tap OK when done.
Eval	Evaluates the highlighted expression, resolving any references to other definitions.

To continue our example, suppose that the heights of the rest of the students in the class are measured and that each one is rounded to the nearest of the five values first recorded. Instead of entering all the new data in D1, we simply add another column, D2, that holds the frequencies of our five data points in D1.

Height (cm)	Frequency
160	5
165	3
170	8
175	2
180	1

1. Tap **Freq** to the right of H1 (or press ▶ to highlight the second H1 field).

2. Tap Column to display the available D*n* lists, and then select **D2**.



Statistics 1Var Symbolic View

√ H1: D1 D2

Plot1: Histogram ▾

🟦 Option1:

□ H2: D2

Plot2: Histogram ▾

🟥 Option2:

□ H3: D1+D2

Enter frequency column

3. Optionally, select a color for the graph.

4. If you have more than one analysis defined in Symbolic view, deselect any analysis you are not currently interested in.

5. Return to Numeric view.

Num ↵Setup

6. In column D2, enter the frequency data shown in the previous table:

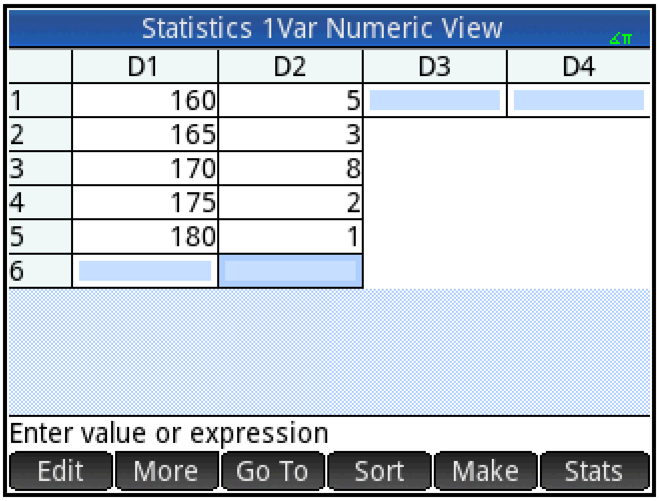
▶ 5 Enter

3 Enter

8 Enter

2 Enter

1 Enter



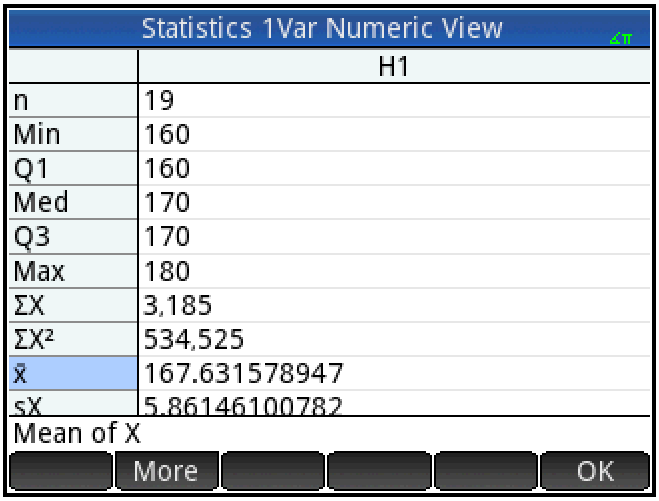
Statistics 1Var Numeric View

	D1	D2	D3	D4
1	160	5		
2	165	3		
3	170	8		
4	175	2		
5	180	1		
6				

Enter value or expression

7. To recalculate the statistics, tap Stats.

The mean height now is approximately 167.631 cm.



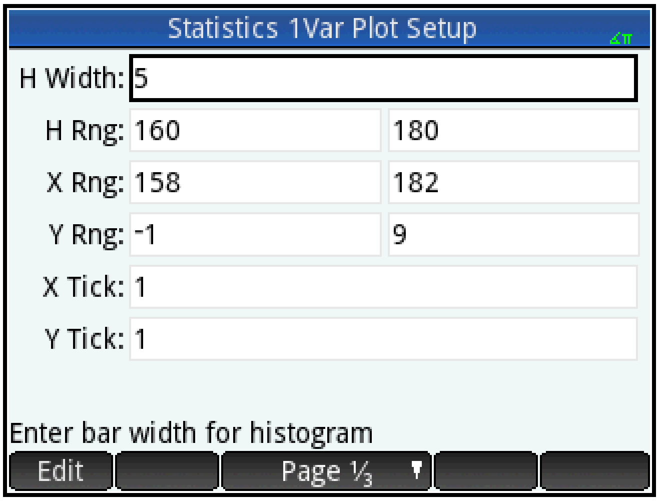
Statistics 1Var Numeric View

	H1
n	19
Min	160
Q1	160
Med	170
Q3	170
Max	180
ΣX	3,185
ΣX²	534,525
x̄	167.631578947
sX	5.86146100782

Mean of X

8. Configure a histogram plot for the data. Tap OK, and then press Shift Plot Setup.

Enter parameters appropriate to your data. Those shown in the following figure ensure that all the data in this particular example are displayed in Plot view.



Statistics 1Var Plot Setup

H Width: 5

H Rng: 160 180

X Rng: 158 182

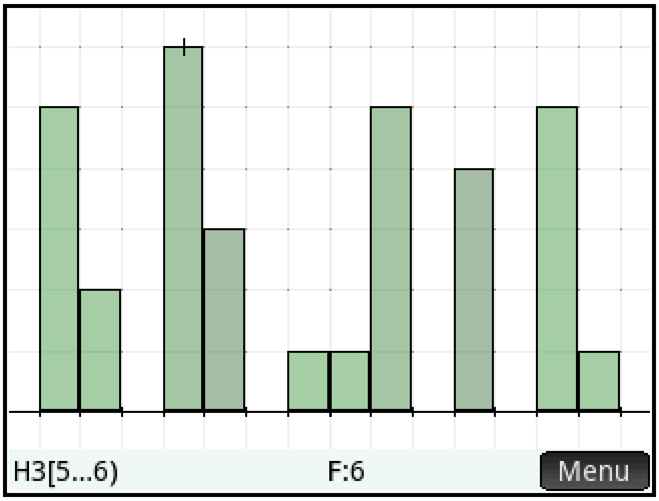
Y Rng: -1 9

X Tick: 1

Y Tick: 1

Enter bar width for histogram

9. To plot a histogram of the data, press Plot Setup.



Press ◀ and ▶ to move the tracer and see the interval and frequency of each bin. You can also tap to select a bin. Tap and drag to scroll the Plot view. You can also zoom in or out on the cursor by pressing + or −, respectively. Finally, you can use a 2-finger pinch zoom gesture performed vertically, horizontally, or diagonally to zoom.

Entering and editing statistical data

Each column in Numeric view is a dataset and is represented by a variable named D0 to D9. There are three ways to get data into a column as follows:

- Go to Numeric view and enter the data directly. See Getting started with the Statistics 1Var app on page 227 for an example.
- Go to Home view and copy the data from a list. For example, if you enter `L1` Sto ▸ `D1` in Home view, the items in list L1 are copied into column D1 in the Statistics 1Var app.
- Go to Home view and copy the data from the Spreadsheet app. For example, suppose the data of interest is in A1:A10 in the Spreadsheet app and you want to copy it into column D7. With the Statistics 1Var app open, return to Home view and enter `Spreadsheet.A1:A10` Sto ▸ `D7` Enter ≈ .

Whichever method you use, the data you enter is automatically saved. You can leave this app and come back to it later. You will find that the data you last entered is still available.

After entering the data, you must define data sets—and the way they are to be plotted—in Symbolic view.

Numeric view: menu items

The menu items you can tap in Numeric view are as follows:

Edit	Copies the highlighted item into the entry line to enable editing.
More	Displays an editing options menu. See More menu on page 234.
Go To	Moves the cursor to the specified item in a list.
Sort	Sorts the data in various ways. See Sorting data values on page 236.
Make	Displays an input form for you to enter a formula that will generate a list of values for a specified column. See Generating data on page 235.
Stats	Calculates statistics for each data set selected in Symbolic view. See Computed statistics on page 236.

More menu

The More menu contains options for editing lists of data. The options are described in the following table.

Option	Sub-option	Purpose
Insert	Row	Inserts a new row in the selected list. The new row contains 0 as its element.
Delete	Column	Deletes the contents of the selected list. To delete a single item, select it and then press ⌫ Del .
Select	Row	Selects the row that contains the currently selected cell; the entire row can then be copied.

Option	Sub-option	Purpose
	Box	Opens a dialog box where you can select a rectangular array defined by a starting location and a final location. You can also tap and hold a cell to select it as the starting location, and then drag your finger to select the rectangular array of elements. After it is selected, the array can be copied.
	Column	Selects the current list. After it is selected, the list can be copied.
Selection		Turns selection mode on and off. If selection mode is off, you can tap and hold a cell and then drag your finger to select a rectangular array.
Swap	Column	Transposes the contents of two columns (or lists).

Editing a data set

In Numeric view, highlight the data to change, type a new value, and press Enter. You can also highlight the data, tap Edit to copy it to the entry line, make your change, and press Enter.

Deleting data

- To delete a data item, highlight it and press Del. The values below the deleted cell will scroll up one row.
- To delete a column of data, highlight an entry in that column and press Shift Esc. Select the column and tap OK.
- To delete all data in every column, press Shift Esc, select **All columns**, and tap OK.

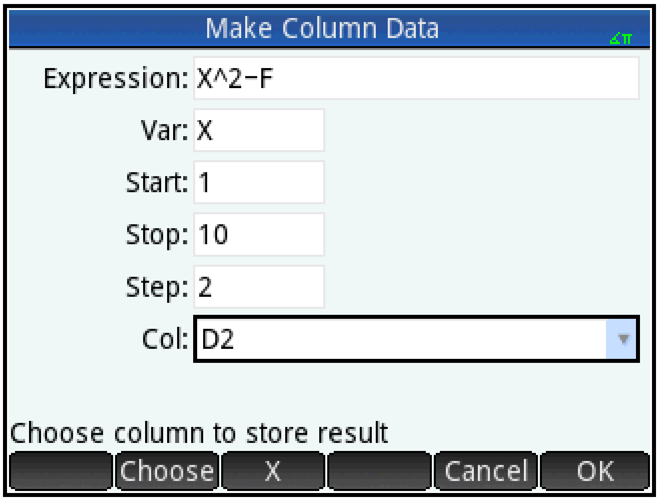
Inserting data

1. Highlight the cell below where you want to insert a value.
2. Tap More, select **Insert**, and then select **Row**.
3. Enter the value or expression and then press Enter.

If you just want to add more data to the data set and it is not important where it goes, select the last cell in the data set and start entering the new data.

Generating data

You can enter a formula to generate a list of data points for a specified column by tapping Make. In the following example, 5 data-points are placed in column D2. They are generated by the expression $X^2 – F$ where X comes from the set {1, 3, 5, 7, 9}. These are the values between 1 and 10 that differ by 2. F is whatever value has been assigned to it elsewhere (such as in Home view). If F happened to be 5, column D2 is populated with {–4, 4, 20, 44, 76}.



Make Column Data

Expression: X^2–F

Var: X

Start: 1

Stop: 10

Step: 2

Col: D2

Choose column to store result

Sorting data values

You can sort up to three columns of data at a time, based on a selected independent column.

1. In Numeric view, place the highlight in the column you want to sort, and tap Sort.
2. Specify the sort order: **Ascending** or **Descending**.
3. Specify the independent and dependent data columns. Sorting is by the independent column. For instance, if ages are in C1 and incomes in C2 and you want to sort by income, then you make C2 the independent column and C1 the dependent column.
4. Specify any frequency data column.
5. Tap OK.

The independent column is sorted as specified and any other columns are sorted to match the independent column. To sort just one column, choose **None** for the **Dependent** and **Frequency** columns.

Computed statistics

Tapping Stats displays the following results for each dataset selected in Symbolic view.

Statistic	Definition
n	Number of data points
Min	Minimum value
Q1	First quartile: median of values to left of median
Med	Median value
Q3	Third quartile: median of values to right of median
Max	Maximum value
ΣX	Sum of data values (with their frequencies)

Statistic	Definition
ΣX^2	Sum of the squares of the data values
$\dot{x}$	Mean
sX	Sample standard deviation
σX	Population standard deviation
serrX	Standard error
ssX	Sum of the squared deviations of X

When the data set contains an odd number of values, the median value is not used when calculating Q1 and Q3. For example, for the data set {3,5,7,8,15,16,17}only the first three items—3, 5, and 7—are used to calculate Q1, and only the last three terms—15, 16, and 17—are used to calculate Q3.

Plotting

You can plot the following:

- Histograms
- Box-and-whisker plots (with and without outliers)
- Normal probability plots
- Line plots
- Bar graphs
- Pareto charts
- Control charts
- Dot plots
- Stem and leaf plots
- Pie charts

After you have entered your data and defined your data set, you can plot your data. You can plot up to five graphs simultaneously. If you are plotting more than one graph, press View Copy and then select **Autoscale** to set up the initial window. Then, you can pan and zoom with your fingers to get an optimal view of the graphs.

Plotting statistical data

1. In the Symbolic view, select the data sets you want to plot.
2. From the **Plotn** menu, select the plot type.

3. For any plot, but especially for a histogram, adjust the plotting scale and range in the Plot Setup view. If you find histogram bars too fat or too thin, you can adjust them by changing the **H Width** setting. (See Setting up the plot on page 243.)

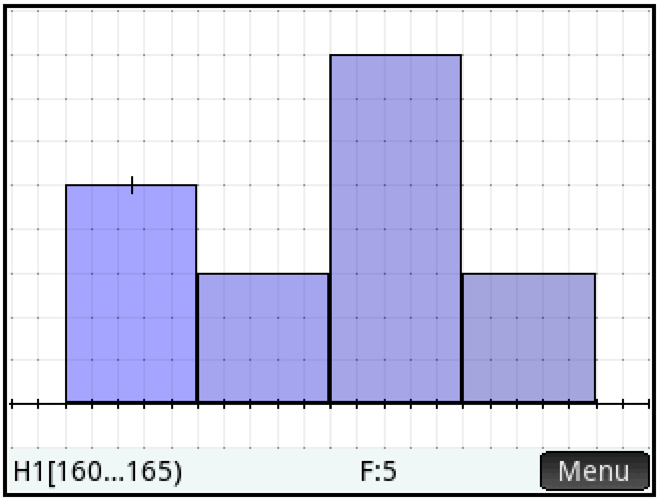
4. Press Plot Setup. If the scaling is not to your liking, press View Copy and select **Autoscale**.

   Autoscale can be relied upon to give a good starting scale which can then be adjusted, either directly in the Plot view or in the Plot Setup view.

Plot types

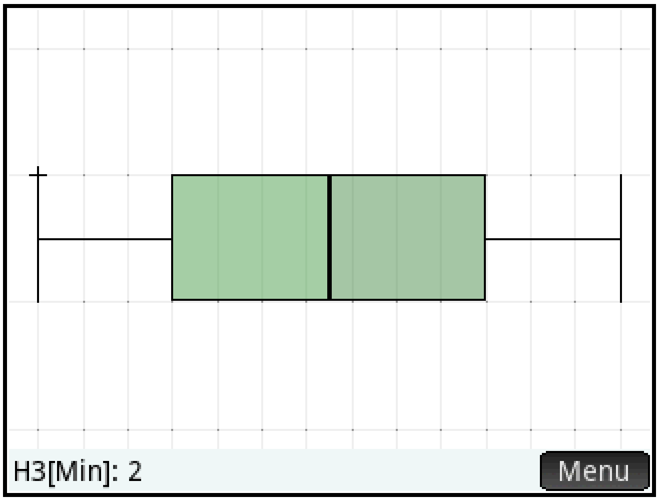
Histogram

The first set of numbers below the plot indicate where the cursor is. In the following example, the cursor is in the bin for data between 5 and 6 (but not including 6), and the frequency for that bin is 6. The data set is defined by H3 in Symbolic view. You can see information about other bins by pressing ◀ or ▶.



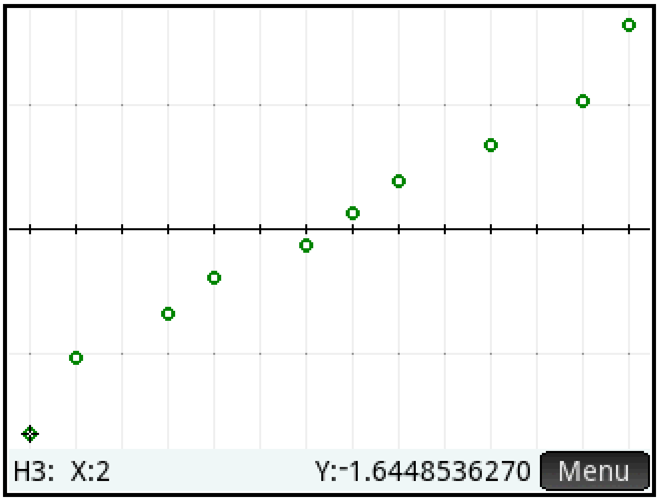
Box-and-whisker plot

The left whisker marks the minimum data value. The box marks the first quartile, the median, and the third quartile. The right whisker marks the maximum data value. The numbers following the plot give the statistic at the cursor. You can see other statistics by pressing ◀ or ▶. In Symbolic view, you can either include or exclude outliers. In the **Option** field, select **Show outliers** to display outliers outside the plot or select **No Outliers** to include any outliers in the data set.


H3[Min]: 2

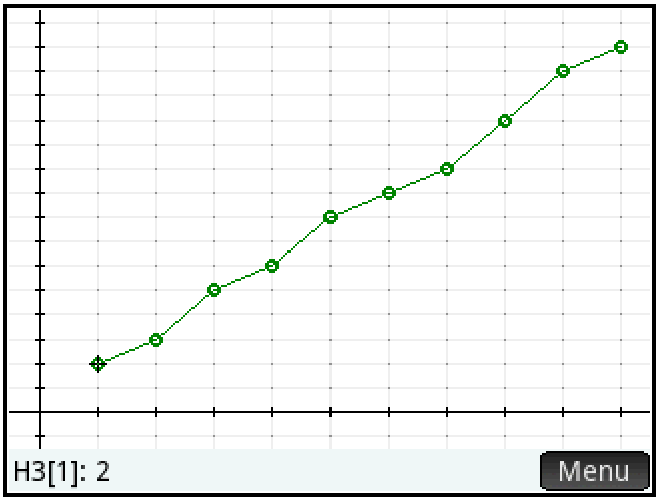
Normal probability plot

The normal probability plot is used to determine whether or not sample data is more or less normally distributed. The more linear the data appear, the more likely that the data are normally distributed.


H3: X:2 Y:-1.6448536270

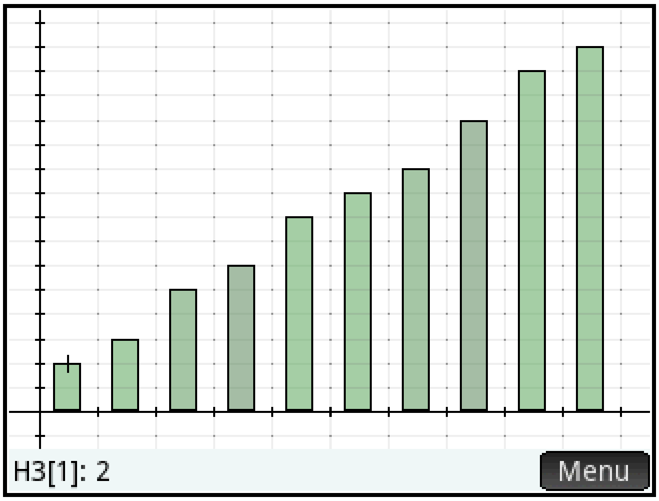
Line plot

The line plot connects points of the form (x, y), where x is the row number of the data point and y is its value.



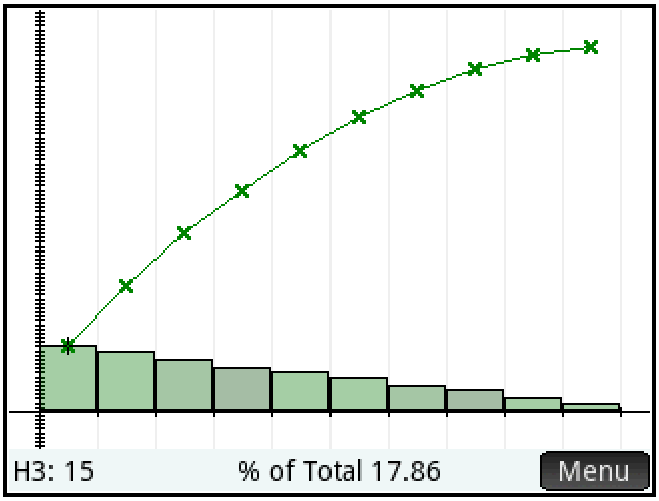
Bar graph

The bar graph shows the value of a data point as a vertical bar placed along the x-axis at the row number of the data point.



Pareto chart

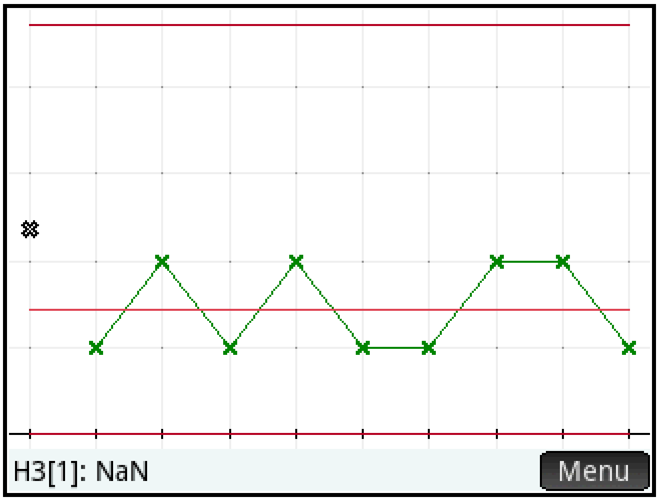
A pareto chart places the data in descending order and displays each with its percentage of the whole.



Control chart

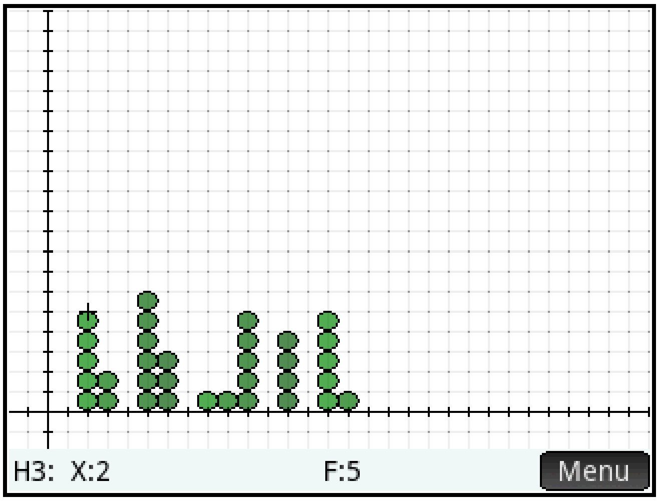
A control chart draws horizontal lines at the mean and both the upper and lower confidence levels. It then plots the data in order and connects the data points with line segments. This plot type has an option for plotting the moving range (the difference between pairs of data points) rather than individual data points.

In the **Option** box, you can select either **Individuals** or **Moving Range**.



Dot plot

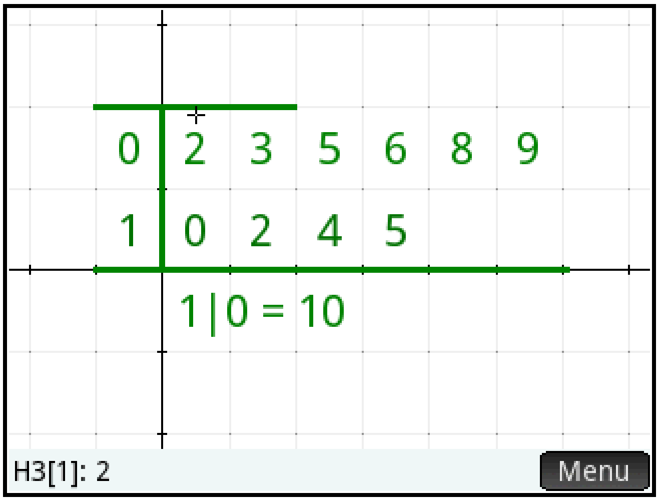
The dot plot draws a dot for each data point and stacks identical data points vertically.



Stem and leaf plot

The stem and leaf plot separates values by powers of ten, with the stem showing the highest power of ten and the leaves displaying the next lower power of ten for each data point. A legend is included at the base of the plot.

In the **Option** box, you can select either **Split stem** or the default **Single stem**. The split stem option splits each stem into two parts at 5, 50, and so on.

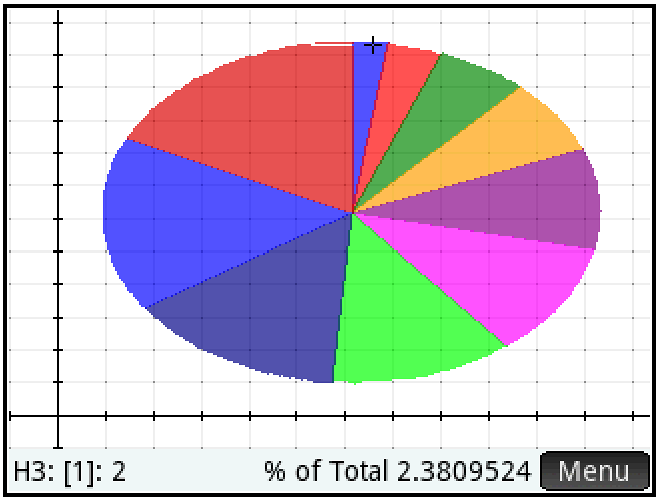


0	2	3	5	6	8	9
1	0	2	4	5		

1|0 = 10

Pie chart

The pie chart displays each data point as a sector of a circle, where the sector's area corresponds to the percentage of the whole data set that the individual data point represents.



H3: [1]: 2 % of Total 2.3809524

Setting up the plot

The Plot Setup view (Shift Plot Setup) enables you to specify many of the same plotting parameters as other apps (such as X Rng and Y Rng). There are two settings unique to the Statistics 1Var app, as follows:

- **Histogram width—H Width** enables you to specify the width of a histogram bin. This determines how many bins will fit in the display, as well as how the data is distributed (that is, how many data points each bin contains).
- **Histogram range—H Rng** enables you to specify the range of values for a set of histogram bins. The range runs from the left edge of the leftmost bin to the right edge of the rightmost bin.

Exploring the graph

The Plot view (Plot Setup) has zooming and tracing options, as well as coordinate display. The Autoscale option is available from the View menu (View Copy) as well as the Zoom menu. The View menu also enables you to view graphs in split-screen mode.

For all plot types, you can tap and drag to scroll the Plot view. You can use a 2-finger pinch zoom performed horizontally to zoom on the x-axis, vertically to zoom on the y-axis, or diagonally to zoom on both axes. You can also zoom in or out on the cursor by pressing + Ans or − Base respectively.

Plot view: menu items

The menu items you can tap in Plot view are as follows:

Button	Purpose
Zoom	Displays the Zoom menu.
Trace•	Turns trace mode on or off.
Defn	Displays the definition of the current statistical plot.
Menu	Shows or hides the menu.

12 Statistics 2Var app

The Statistics 2Var app can store up to ten data sets at one time. It can perform two-variable statistical analysis of one or more sets of data.

The Statistics 2Var app starts with the Numeric view which is used to enter data. The Symbolic view is used to specify which columns contain data and which column contains frequencies.

You can also compute statistics in Home and in the Spreadsheet app.

The values computed in the Statistics 2Var app are saved in variables. These can be referenced in Home view and in other apps.

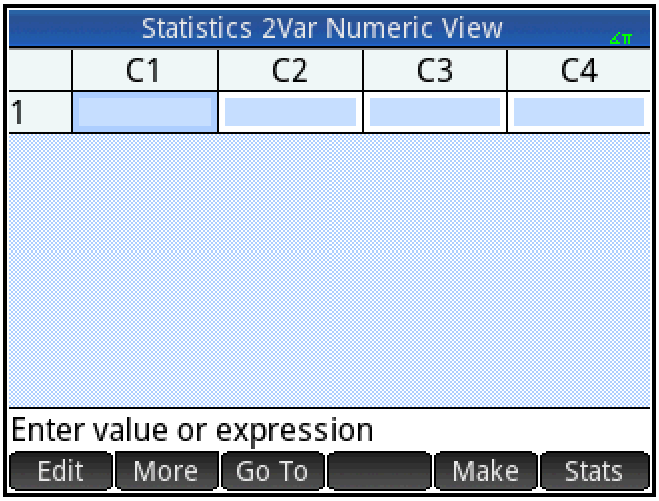
Getting started with the Statistics 2Var app

The following example uses the advertising and sales data in the table below. In the example, you will enter the data, compute summary statistics, fit a curve to the data, and predict the effect of more advertising on sales.

Advertising minutes	**Resulting sales ($)**
(independent, x)	**(dependent, y)**
2	1400
1	920
3	1100
5	2265
5	2890
4	2200

Opening the Statistics 2Var app

▲ Press Apps Info and then select **Statistics 2Var** to open the Statistics 2Var app.


Statistics 2Var Numeric View

	C1	C2	C3	C4
1				

Enter value or expression

Entering data

1. Enter the advertising minutes data in column C1:

   2 Enter 1 Enter 3 Enter 5 Enter 5 Enter 4 Enter

2. Enter the resulting sales data in column C2:

   1400 Enter

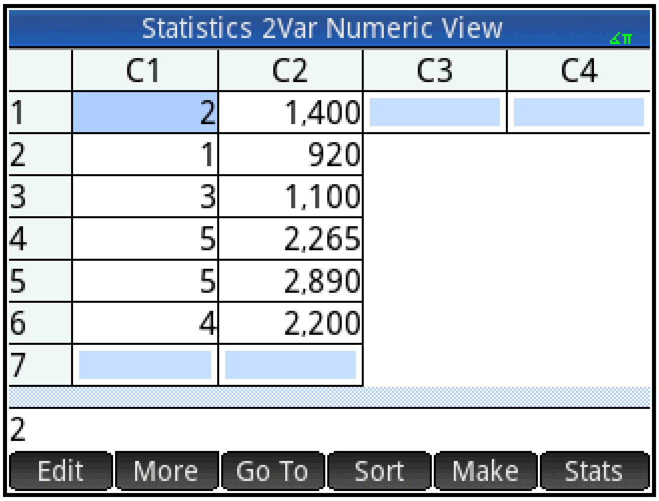
   920 Enter

   1100 Enter

   2265 Enter

   2890 Enter

   2200 Enter



	C1	C2	C3	C4
1	2	1,400		
2	1	920		
3	3	1,100		
4	5	2,265		
5	5	2,890		
6	4	2,200		
7				

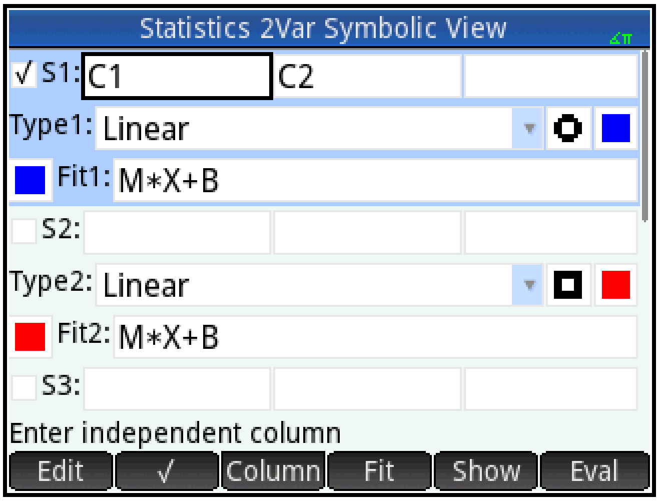
2

Choosing data columns and fit

In Symbolic view, you can define up to five analyses of two-variable data, named S1 to S5. In this example, we will define just one: S1. The process involves choosing data sets and a fit type.

1. Press Symb Setup to specify the columns that contain the data you want to analyze.

   In this case, C1 and C2 appear by default. But you could have entered your data into columns other than C1 and C2.



☑ S1: C1 C2

Type1: Linear ○ 🟦

🟦 Fit1: M*X+B

□ S2:

Type2: Linear □ 🟥

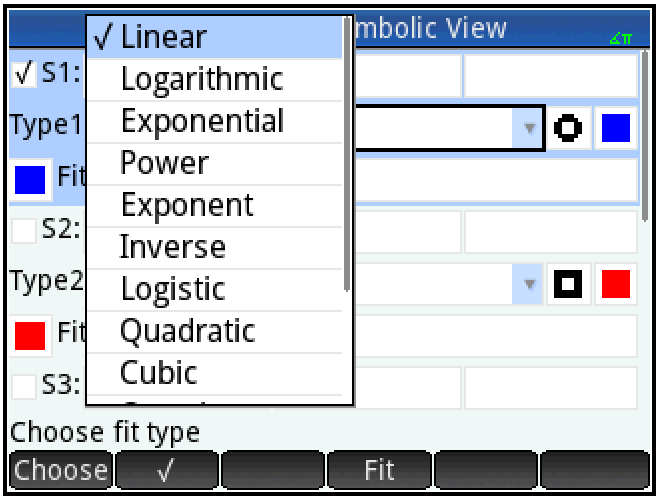
🟥 Fit2: M*X+B

□ S3:

Enter independent column

2. Select a fit:

   From the **Type 1** box select a fit. In this example, select **Linear**.



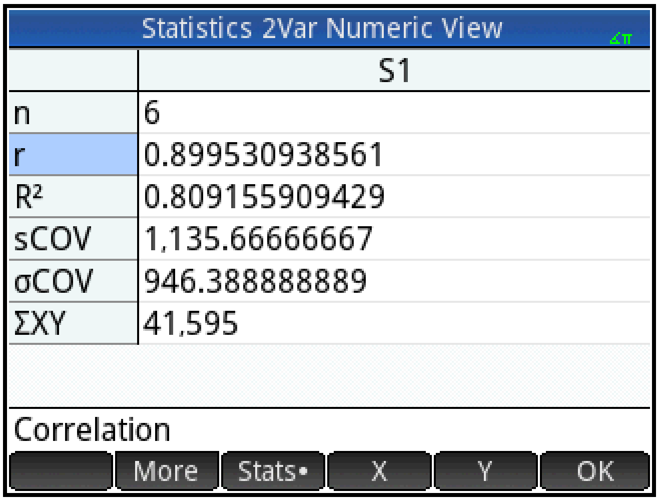
3. Optionally, select a point type and color for the scatter plot.
4. Optionally, select a color for the graph of the fit using the color menu to the left of **Fit**.
5. If you have more than one analysis defined in Symbolic view, deselect any analysis you are not currently interested in.

Exploring statistics

1. Find the correlation, r, between advertising time and sales:

   Num Setup Stats

   The correlation is r=0.8995...



Statistics 2Var Numeric View

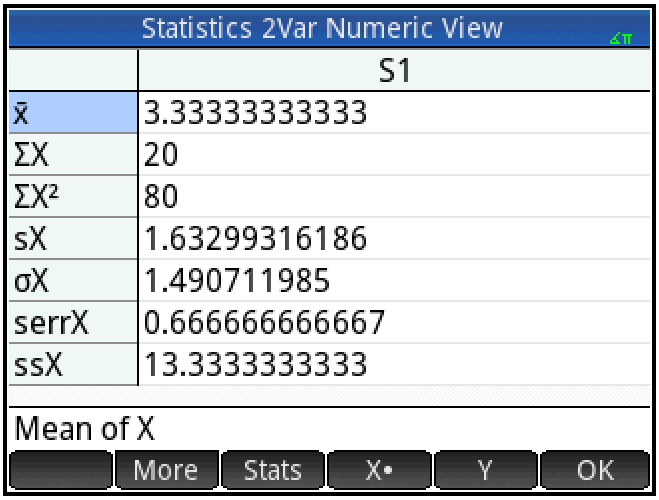
	S1
n	6
r	0.899530938561
R²	0.809155909429
sCOV	1,135.66666667
σCOV	946.388888889
ΣXY	41,595

Correlation

2. Find the mean advertising time (ẋ).

X

The mean advertising time, ẋ, is 3.33333... minutes.



Statistics 2Var Numeric View

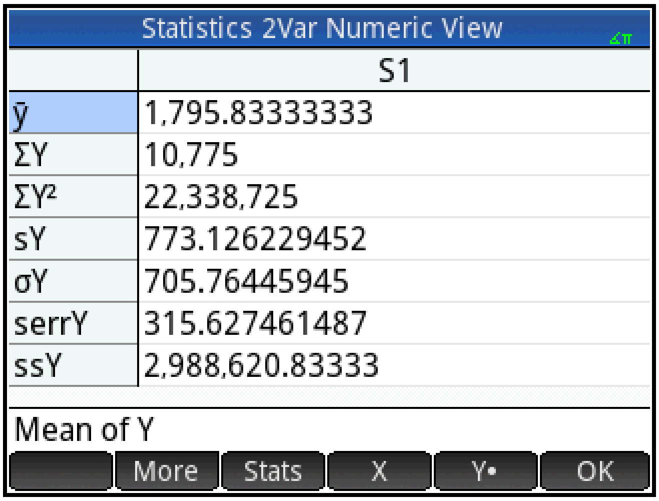
	S1
x̄	3.33333333333
ΣX	20
ΣX²	80
sX	1.63299316186
σX	1.490711985
serrX	0.666666666667
ssX	13.3333333333

Mean of X

3. Find the mean sales (ẏ).

Y

The mean sales, ẏ, is approximately $1,796.



Statistics 2Var Numeric View

	S1
ȳ	1,795.83333333
ΣY	10,775
ΣY²	22,338,725
sY	773.126229452
σY	705.76445945
serrY	315.627461487
ssY	2,988,620.83333

Mean of Y

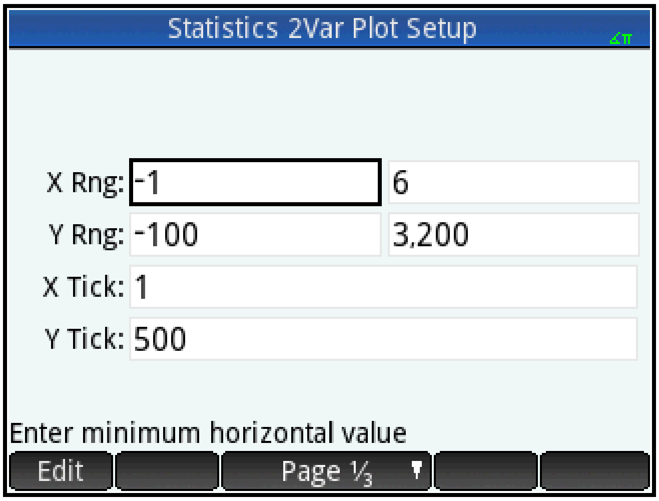
Press OK to return to Numeric view.

Setting up the plot

▲ Change the plotting range to ensure that all the data points are plotted.

Shift Plot Setup +/− 1 Enter 6 Enter +/− 100 Enter 3200 Enter ▼

500 Enter



Statistics 2Var Plot Setup

X Rng: -1 6

Y Rng: -100 3,200

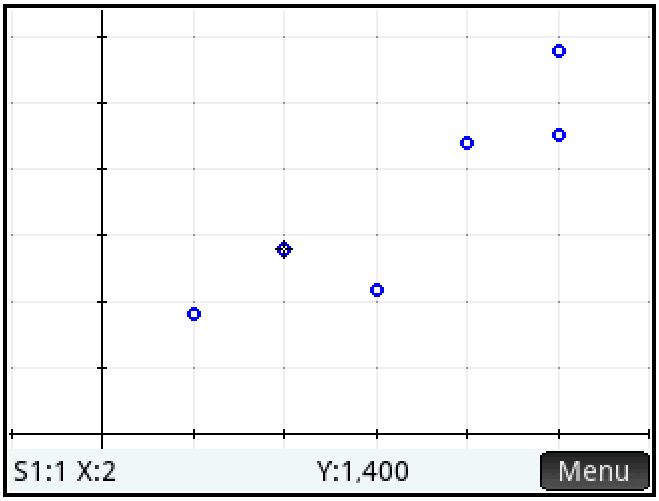
X Tick: 1

Y Tick: 500

Enter minimum horizontal value

Plotting the graph

1. Press Plot ↳Setup to plot the graph.



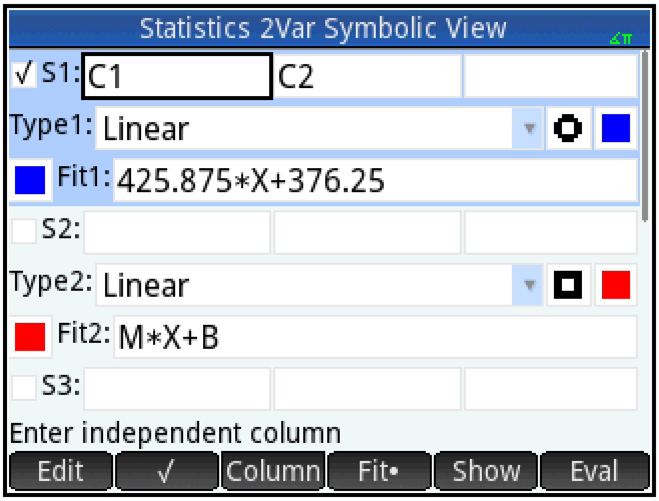
S1:1 X:2 Y:1,400

2. Tap Menu and then tap Fit to plot the fit.

Displaying the equation

▲ Press Symb Setup to return to Symbolic view.

Note the expression in the **Fit1** field. It shows that the slope (m) of the regression line is 425.875 and the y-intercept (b) is 376.25.



Statistics 2Var Symbolic View

√ S1: C1 | C2

Type1: Linear ▾ ● 🟦

🟦 Fit1: 425.875*X+376.25

□ S2:

Type2: Linear ▾ ■ 🟥

🟥 Fit2: M*X+B

□ S3:

Enter independent column

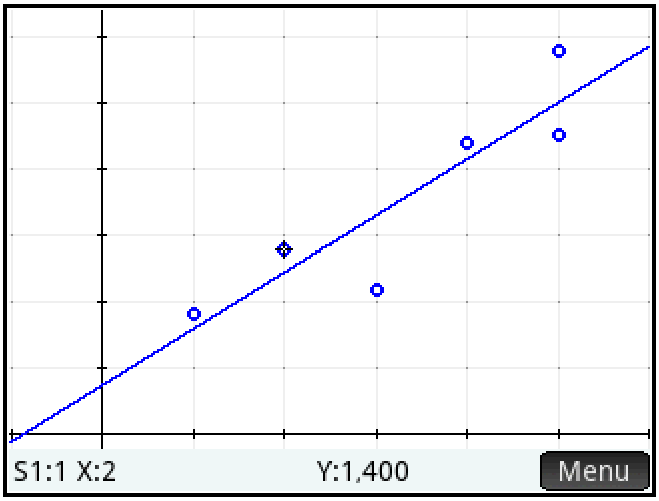
Predicting values

Let's now predict the sales figure if advertising were to go up to 6 minutes.

1. Press Plot Setup to return to Plot view.

   The trace option is active by default. This option will move the cursor from data point to data point as you press ◀ or ▶. As you move from data point to data point, the corresponding x- and y-values appear at the bottom of the screen. In this example, the x-axis represents minutes of advertising and the y-axis represents sales.

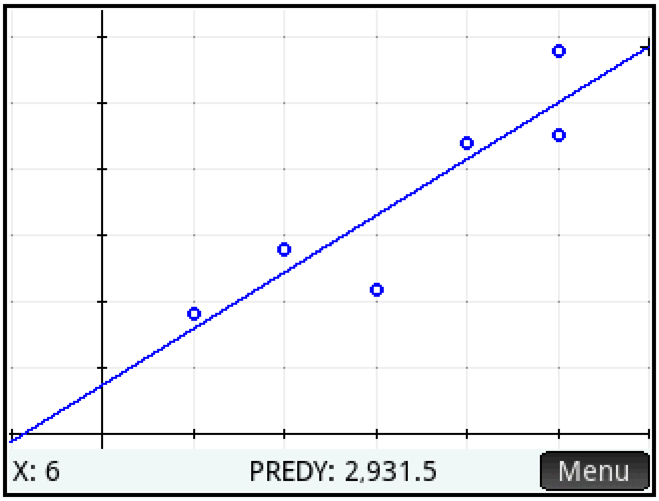
   However, there is no data point for 6 minutes. Thus we cannot move the cursor to x = 6. Instead, we need to predict what y will be when x = 6, based on the data we do have. To do that, we need to trace the regression curve, not the data points we have.



S1:1 X:2 Y:1,400

2. Press ▲ or ▼ to set the cursor to trace the regression line rather than the data points.

The cursor jumps from whatever data point it was on to the regression curve.



X: 6 PREDY: 2,931.5

3. Tap on the regression line near x = 6 (near the right edge of the display). Then press ▶ until x = 6. If the x-value is not shown at the bottom left of the screen, tap Menu Paste . When you reach x = 6, you will see that the **PREDY** value (also displayed at the bottom of the screen) reads 2931.5. Thus the model predicts that sales would rise to $2,931.50 if advertising were increased to 6 minutes.

TIP: You could use the same tracing technique to predict—although roughly—how many minutes of advertising you would need to gain sales of a specified amount. However, a more accurate method is available: return to Home view and enter `Predx(s)` where s is the sales figure. Predy and Predx are app functions.

Entering and editing statistical data

Each column in Numeric view is a dataset and is represented by a variable named C0 to C9. There are three ways to get data into a column, as follows:

- Go to Numeric view and enter the data directly. See Getting started with the Statistics 2Var app on page 245 for an example.
- Go to Home view and copy the data from a list. For example, if you enter `L1`, tap Sto ▸, and then enter `C1` in Home view, the items in list L1 are copied into column C1 in the Statistics 1Var app.
- Go to Home view and copy the data from the Spreadsheet app. For example, suppose the data of interest is in A1:A10 in the Spreadsheet app and you want to copy it into column C7. With the Statistics 2Var app open, return to Home view and enter `Spreadsheet.A1:A10`, tap Sto ▸, enter `C7`, and then press Enter ≈ .

NOTE: A data column must have at least four data points to provide valid two-variable statistics.

Whichever method you use, the data you enter is automatically saved. You can leave this app and come back to it later. You will find that the data you last entered is still available.

After entering the data, you must define data sets—and the way they are to be plotted—in Symbolic view.

Numeric view: menu items

The menu items you can tap in Numeric view are as follows:

Edit	Copies the highlighted item into the entry line to enable editing.
More	Displays an editing options menu. See More menu on page 253.
Go To	Moves the cursor to the specified item in a list.
Sort	Sorts the data in various ways.
Make	Displays an input form for you to enter a formula that will generate a list of values for a specified column.
Stats	Calculates statistics for each data set selected in Symbolic view.

More menu

The More menu contains options for editing lists of data. The options are described in the following table.

Option	Sub-option	Purpose
Insert	Row	Inserts a new row in the selected list. The new row contains 0 as its element.
Delete	Column	Deletes the contents of the selected list. To delete a single item, select it and then press Del .
Select	Row	Selects the row that contains the currently selected cell; the entire row can then be copied.
	Box	Opens a dialog box where you can select a rectangular array defined by a starting location and a final location. You can also tap and hold a cell to select it as the starting location, and then drag your finger to select the rectangular array of elements. After it is selected, the array can be copied.
	Column	Selects the current list. After it is selected, the list can be copied.
Selection		Turns selection mode on and off. If selection mode is off, you can tap and hold a cell and then drag your finger to select a rectangular array.
Swap	Column	Transposes the contents of two columns (or lists).

Defining a regression model

You define a regression model in Symbolic view. There are three ways to do so, as follows:

- Accept the default option to fit the data to a straight line.
- Choose a predefined fit type (logarithmic, exponential, and so on).
- Enter your own mathematical expression. The expression will be plotted so that you can see how closely it fits the data points.

Choosing a fit

1. Press Symb Setup to display the Symbolic view.
2. For the analysis you are interested in (S1 through S5), select the **Type** field.
3. Tap the field again to see the menu of fit types.
4. Select your preferred fit type from the menu. (See Fit types on page 254.)

Fit types

Twelve fit types are available, as follows:

Fit type	Meaning
Linear	(Default.) Fits the data to a straight line: y = mx + b. Uses a least-squares fit.
Logarithmic	Fits the data to a logarithmic curve: y = m lnx + b.
Exponential	Fits the data to the natural exponential curve: y = b * e^{mx}
Power	Fits the data to a power curve: y = b * x^m
Exponent	Fits the data to an exponential curve: y = b * m^x
Inverse	Fits the data to an inverse variation: y = m/x + b
Logistic	Fits the data to a logistic curve: $y = \dfrac{L}{1 + ae^{(-bx)}}$ where L is the saturation value for growth. You can store a positive real value in L, or—if L = 0—let L be computed automatically.
Quadratic	Fits the data to a quadratic curve: y = $ax^2 + bx + c$. Needs at least three points.
Cubic	Fits the data to a cubic polynomial: y = $ax^3 + bx^2 + cx + d$
Quartic	Fits to a quartic polynomial: y = $ax^4 + bx^3 + cx^2 + dx + e$
Trigonometric	Fits the data to a trigonometric curve: y = a * sin(bx + c) + d. Needs at least three points.
User Defined	Define your own fit (see the following).

Defining your own fit

1. Press Symb Setup to display the Symbolic view.
2. For the analysis you are interested in (S1 through S5), select the **Type** field.
3. Tap the field again to see a menu of fit types.
4. Select **User Defined** from the menu.

5. Select the corresponding fit field.

6. Enter an expression and press Enter ≈. The independent variable must be X, and the expression must not contain any unknown variables. For example, 1.5 * cos(x) + 0.3 * sin(x). Note that in this app, variables must be entered in uppercase.

Computed statistics

When you tap Stats, three sets of statistics become available. By default, the statistics involving both the independent and dependent columns are shown. Tap X to see the statistics involving just the independent column or Y to display the statistics derived from the dependent column. Tap Stats to return to the default view. The tables below describe the statistics displayed in each view.

The statistics computed when you tap Stats are as follows:

Statistic	**Definition**
n	The number of data points.
r	Correlation coefficient of the independent and dependent data columns, based only on the linear fit (regardless of the fit type chosen). Returns a value between –1 and 1, where 1 and –1 indicate best fits.
R^2	The coefficient of determination, that is, the square of the correlation coefficient. The value of this statistics is dependent on the Fit type chosen. A measure of 1 indicates a perfect fit.
sCOV	Sample covariance of independent and dependent data columns.
σCOV	Population covariance of independent and dependent data columns.
ΣXY	Sum of all the individual products of x and y.

The statistics displayed when you tap X are as follows:

Statistic	**Definition**
ẋ	Mean of x- (independent) values.
ΣX	Sum of x-values.
ΣX^2	Sum of x^2-values.
sX	The sample standard deviation of the independent column.
σX	The population standard deviation of the independent column.
serrX	The standard error of the independent column.
ssX	Sum of the squared deviations of X.

The statistics displayed when you tap Y are as follows:

Statistic	Definition
ẏ	Mean of y- (dependent) values.
ΣY	Sum of y-values.
$ΣY^2$	Sum of y^2-values.
sY	The sample standard deviation of the dependent column.
σY	The population standard deviation of the dependent column.
serrY	The standard error of the dependent column.
ssY	Sum of the squared deviations of Y.

Plotting statistical data

Once you have entered your data, selected the data set to analyze and specified your fit model, you can plot your data. You can plot up to five scatter plots at a time.

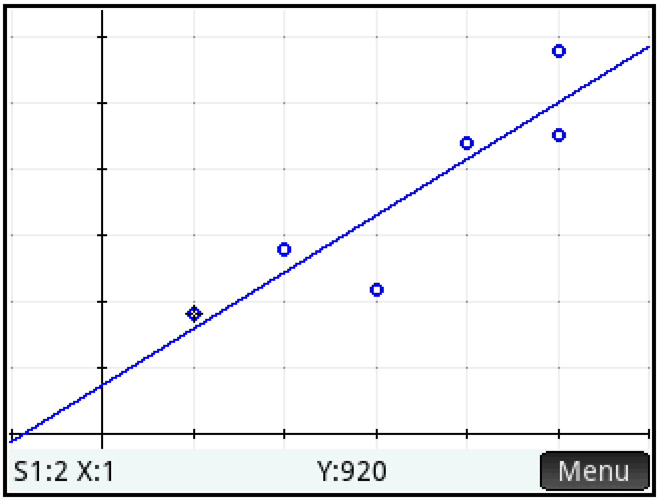
1. In Symbolic view, select the data sets you want to plot.
2. Make sure that the full range of your data will be plotted. You do this by reviewing (and adjusting, if necessary), the **X Rng** and **Y Rng** fields in Plot Setup view. (Shift Plot Setup).
3. Press Plot Setup .

   If the data set and regression line are not ideally positioned, press View Copy and select **Autoscale**.

   Autoscale can be relied upon to give a good starting scale which can then be adjusted later in the Plot Setup view.

Tracing a scatter plot

The figures below the plot indicate that the cursor is at the second data point of S1, at ((1, 920). Press ▶ to move to the next data point and display information about it.



S1:2 X:1 Y:920

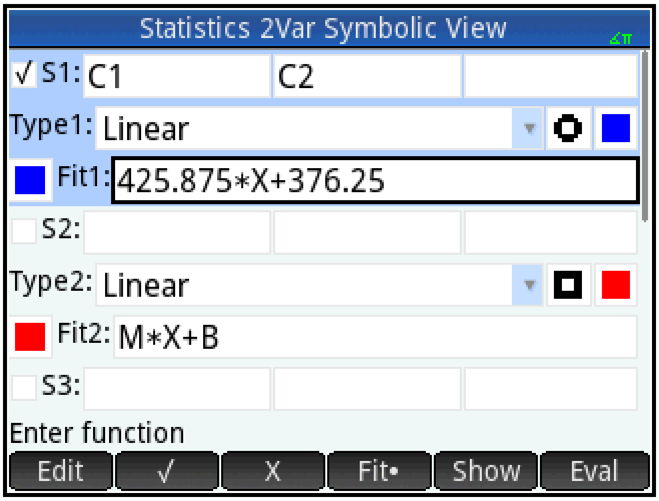
Tracing a curve

If the regression line is not showing, tap Fit. The coordinates of the tracer cursor are shown at the bottom of the screen. (If they are not visible, tap Menu Paste.)

Press Symb Setup to see the equation of the regression line in Symbolic view.

If the equation is too wide for the screen, select it and tap Show.

The following example shows that the slope of the regression line (m) is 425.875 and the y-intercept (b) is 376.25.



Statistics 2Var Symbolic View

√ S1: C1 C2

Type1: Linear ○ 🟦

🟦 Fit1: 425.875*X+376.25

□ S2:

Type2: Linear □ 🟥

🟥 Fit2: M*X+B

□ S3:

Enter function

Tracing order

While ▶ and ◀ move the cursor along a fit or from point to point in a scatter plot, use ▲ and ▼ to choose the scatter plot or fit you wish to trace. For each active analysis (S1–S5), the tracing order is the scatter plot first and the fit second. So if both S1 and S2 are active, the tracer is by default on the S1 scatter plot when you press Plot ↳Setup . Press ▼ to trace the S1 fit. At this point, press ▲ to return to the S1 scatter plot or ▼ again to trace the S2 scatter plot. Press ▼ a third time to trace the S2 fit. If you press ▼ a fourth time, you will return to the S1 scatter plot. If you are confused as to what you are tracing, just tap Defn to see the definition of the object (scatter plot or fit) currently being traced.

Plot view: menu items

The menu items in Plot view are as follows:

Button	Purpose
Zoom	Displays the Zoom menu.
Trace•	Turns trace mode on or off.
Go To	Enables you to specify an x-value on the curve of best fit to jump to (or a data point to jump to if your cursor is on a data point rather than on the curve of best fit).
Fit	Shows or hides a curve that best fits the analysis active in Symbolic view.
Fcn	Opens the Function menu. See Function menu on page 258.
Menu	Shows or hides the menu buttons.

Function menu

The menu items in the Function menu are as follows:

Option	Purpose
Fit	Shows or hides a curve that best fits the analysis active in Symbolic view. You can also tap the button in the Plot view menu.
Sketch	Enables you to sketch a function fit curve for the scatter plot with your finger.
Definition	Displays the definition of the current scatter plot or fit curve. Press ▲ or ▼ to switch between the scatter plot and the fit curve and to scroll through each plot active in Symbolic view.

Sketch

The Sketch option opens the Plot view, with a message displayed at the bottom of the screen to sketch a function fit with your finger. You can sketch a new function if you do not like your previous sketch. After you finish sketching a function, tap OK. The fit type for the first available dataset in Symbolic view (S1–S5) is changed to **User-Defined** and the expression (in X) of your fit is saved as the user-defined fit definition.

Plot Setup view

As with all the apps that provide a plotting feature, the Plot Setup view—Shift Plot Setup—enables you to set the range and appearance of Plot view. The settings are common to other Plot Setup view operations. Page 2 of the Plot Setup view has a **Connect** field. If you choose this option, straight line segments join the data points in Plot view.

Predicting values

PredX is a function that predicts a value for X given a value for Y. Likewise, PredY is a function that predicts a value for Y given a value for X. In both cases, the prediction is based on the equation that best fits the data according to the specified fit type.

You can predict values in the Plot view of the Statistics 2Var app and also in Home view.

Plot view

1. In the Plot view, tap Fit to display the regression curve for the data set (if it is not already displayed).
2. Make sure the trace cursor is on the regression curve. (Press ▲ or ▼ if it is not.)
3. Press ▶ or ◀ The cursor moves along the regression curve and the corresponding X and Y values are displayed across the bottom of the screen. (If these values are not visible, tap Menu Paste.)

You can force the cursor to a specific X value by tapping Go To, entering the value and tapping OK. The cursor jumps to the specified point on the curve.

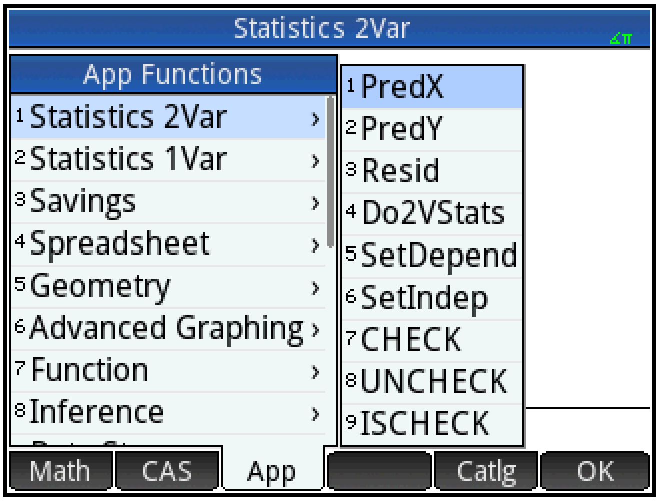
Home view

If the Statistics 2Var app is the active app, you can also predict X and Y values in the Home view.

- Enter `PredX(Y)` and then press Enter ≈ to predict the X value for the specified Y value.
- Enter `PredY(X)` and then press Enter ≈ to predict the Y value for the specified X value.

NOTE: In cases where more than one fit curve is displayed, the PredX and PredY functions use the first active fit defined in Symbolic view.

You can type PredX and PredY directly on the entry line, or select them from the App functions menu (under the Statistics 2Var category). The App functions menu is one of the Toolbox menus (Mem B).



Troubleshooting a plot

If you have problems plotting, check the following:

- The fit (that is, regression model) that you intended to select is the one selected.
- Only those data sets you want to analyze or plot are selected in Symbolic view.
- The plotting range is suitable. Try pressing View Copy and selecting **Autoscale**, or adjust the plotting parameters in Plot Setup view.
- Ensure that both paired columns contain data, and are of the same length.

13 Inference app

The Inference app calculates hypothesis tests, confidence intervals, and chi-square tests, in addition to both tests and confidence intervals based on inference for linear regression. In addition to the Inference app, the Math menu has a full set of probability functions based on various distributions (chi-square, F, binomial, poisson, and so on).

Based on statistics from one or two samples, you can test hypotheses and find confidence intervals for the following quantities:

- Mean
- Proportion
- Difference between two means
- Difference between two proportions

You can perform goodness of fit tests and tests on two-way tables based on the chi-square distribution. You can also perform the following calculations based on inference for linear regression:

- Linear t-test
- Confidence interval for slope
- Confidence interval for the intercept
- Confidence interval for mean response
- Prediction interval for a future response

You can also perform a one-way analysis of variance (ANOVA) on lists of data.

Sample data

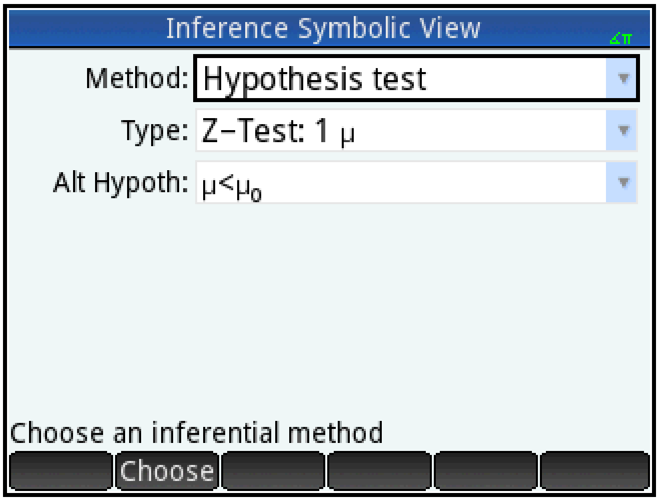
For many of the calculations, the Numeric view of the Inference app comes with sample data (which you can restore by resetting the app). This sample data is useful in helping you gain an understanding of the app.

Getting started with the Inference app

Use the following sections to conduct a Z-Test on one mean using the sample data.

Opening the Inference app

▲ Press Apps Info and then select **Inference**.



Inference Symbolic View

Method: Hypothesis test

Type: Z–Test: 1 μ

Alt Hypoth: $\mu<\mu_0$

Choose an inferential method

The Inference app opens in Symbolic view.

Symbolic view options

The following tables summarize the options available in Symbolic view.

Table 13-1 Hypothesis tests

Test	Description
Z-Test: 1 μ	The Z-Test on one mean
Z-Test: $\mu_1 - \mu_2$	The Z-Test on the difference between two means
Z-Test: 1 π	The Z-Test on one proportion
Z-Test: $\pi_1 - \pi_2$	The Z-Test on the difference between two proportions
T-Test: 1 μ	The T-Test on one mean
T-Test: $\mu_1 - \mu_2$	The T-Test on the difference between two means

Table 13-2 Confidence intervals

Test	Description
Z-Int: 1 μ	The confidence interval for one mean, based on the Normal distribution
Z-Int: $\mu_1 - \mu_2$	The confidence interval for the difference between two means, based on the Normal distribution
Z-Int: 1 π	The confidence interval for one proportion, based on the Normal distribution
Z-Int: $\pi_1 - \pi_2$	The confidence interval for the difference between two proportions, based on the Normal distribution

Table 13-2 Confidence intervals (continued)

Test	Description
T-Int: 1 μ	The confidence interval for one mean, based on the Student's t-distribution
T-Int: $\mu_1 - \mu_2$	The confidence interval for the difference between two means, based on the Student's t-distribution

Table 13-3 X^2 test

Test	Description
Goodness of fit	The chi-square goodness of fit test, based on categorical data
2-way test	The chi-square test, based on categorical data in a two-way table

Table 13-4 Regression

Test	Description
Linear t-test	The t-test for linear regression
Interval: Slope	The confidence interval for the slope of the true linear regression line, based on the t-distribution
Interval: Intercept	The confidence interval for the y-intercept of the true linear regression line, based on the t-distribution
Interval: Mean response	The confidence interval for a mean response, based on the t-distribution
Prediction interval	The prediction interval for a future response, based on the t-distribution

Table 13-5 ANOVA

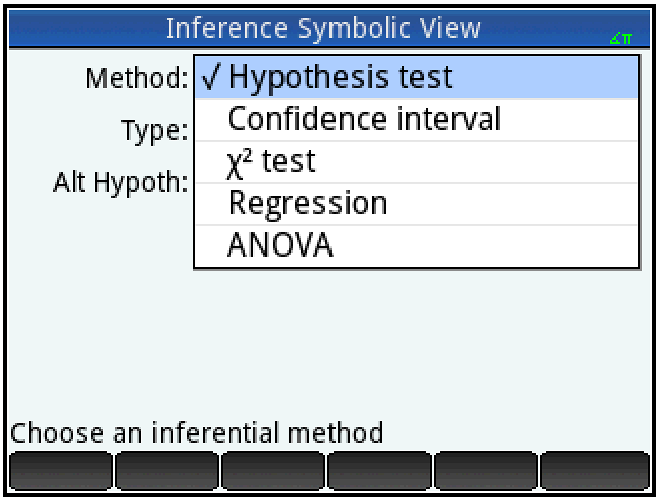
Test	Description
1-way ANOVA	One-way analysis of variance, based on the F-distribution

If you choose one of the hypothesis tests, you can choose an alternative hypothesis to test against the null hypothesis. For each test, there are three possible choices for an alternative hypothesis based on a quantitative comparison of two quantities. The null hypothesis is always that the two quantities are equal. Thus, the alternative hypotheses cover the various cases for the two quantities being unequal: <, >, and ≠.

In this section, we will conduct a Z-Test on one mean on the example data to illustrate how the app works.

Selecting the inference method

1. **Hypothesis test** is the default inference method. If it is not selected, tap **Method** and then select it.



Inference Symbolic View

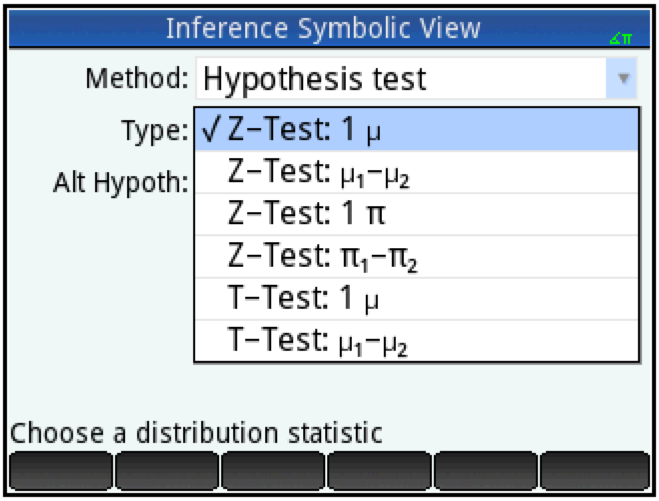
Method: ✓ Hypothesis test

Type:

Alt Hypoth:

Choose an inferential method

2. Choose the type of test. In this case, select **Z-Test: 1 µ** from the **Type** menu.



Inference Symbolic View

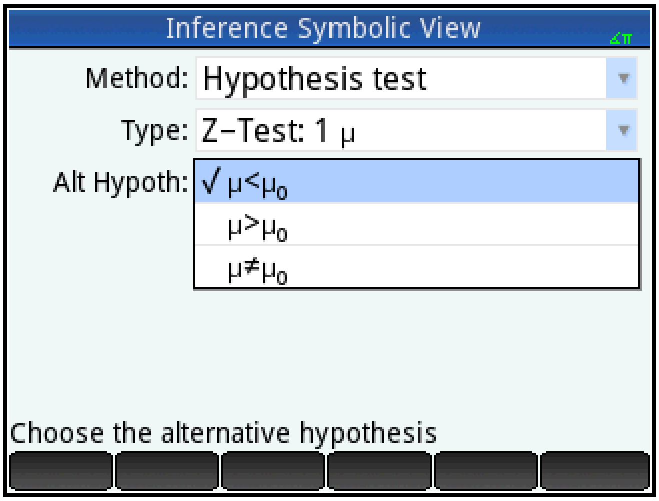
Method: Hypothesis test

Type: ✓ Z–Test: 1 µ

Alt Hypoth:

Choose a distribution statistic

3. Select an alternative hypothesis. In this case, select **μ <** μ_0 from the **Alt Hypoth** menu.



Inference Symbolic View

Method: Hypothesis test

Type: Z–Test: 1 μ

Alt Hypoth: ✓ $\mu<\mu_0$

$\mu>\mu_0$

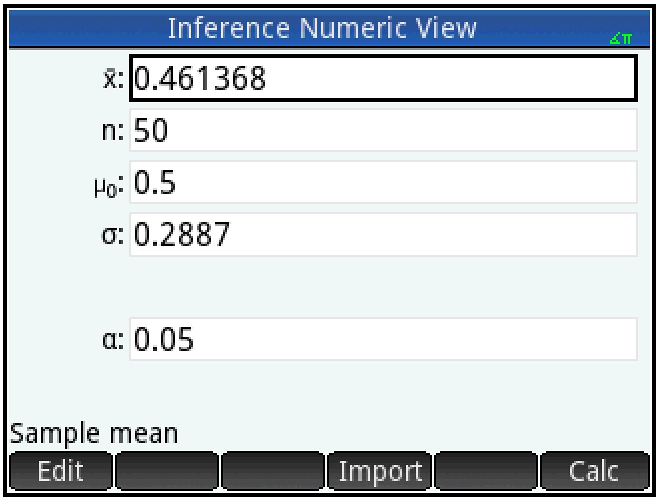
$\mu\neq\mu_0$

Choose the alternative hypothesis

Entering data

▲ Go to Numeric view to see the sample data.

Num ↳Setup



Inference Numeric View

x̄: 0.461368

n: 50

μ_0: 0.5

σ: 0.2887

α: 0.05

Sample mean

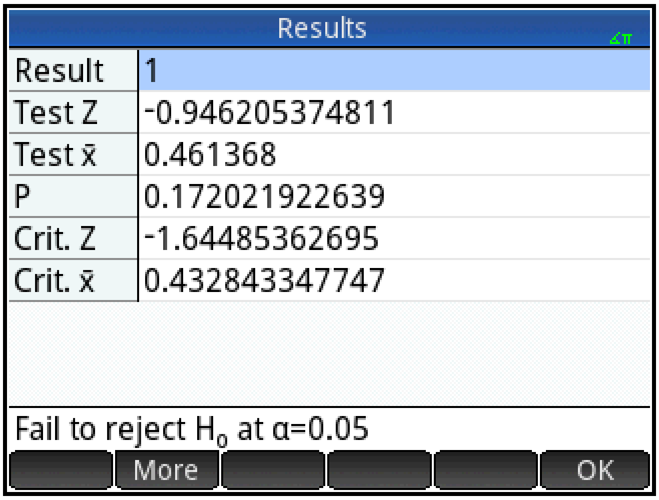
The following table describes the fields in this view for the sample data.

Field name	Description
ẋ	Sample mean
n	The confidence interval for the slope of the true linear regression line, based on the t-distribution
μ_0	Assumed population mean
σ	Population standard deviation
α	Alpha level for the test

The Numeric view is where you enter the sample statistics and population parameters for the situation you are examining. The sample data supplied here belong to the case in which a student has generated 50 pseudo-random numbers on his graphing calculator. If the algorithm is working properly, the mean would be near 0.5 and the population standard deviation is known to be approximately 0.2887. The student is concerned that the sample mean (0.461368) seems a bit low and it testing the less than alternative hypothesis against the null hypothesis.

Displaying the test results

▲ Tap Calc.



Results

Result	1
Test Z	-0.946205374811
Test x̄	0.461368
P	0.172021922639
Crit. Z	-1.64485362695
Crit. x̄	0.432843347747

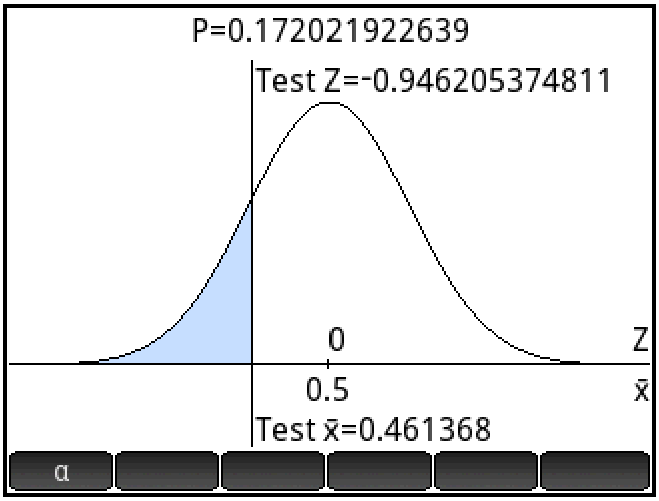
Fail to reject H_0 at α=0.05

The test distribution value and its associated probability are displayed, along with the critical value(s) of the test and the associated critical value(s) of the statistic. In this case, the test indicates that one should not reject the null hypothesis.

Tap OK to return to Numeric view.

Plotting the test results

▲ Press Plot ↳Setup.



The graph of the distribution is displayed, with the test Z-value marked. The corresponding X-value is also shown.

Tap α to see the critical Z-value. With the alpha level showing, you can press ▲ or ▼ to decrease or increase the α-level.

Importing statistics

For many of the calculations, the Inference app can import summary statistics from data in the Statistics 1Var and Statistics 2Var apps. For the others, the data can be manually imported. The following example illustrates the process.

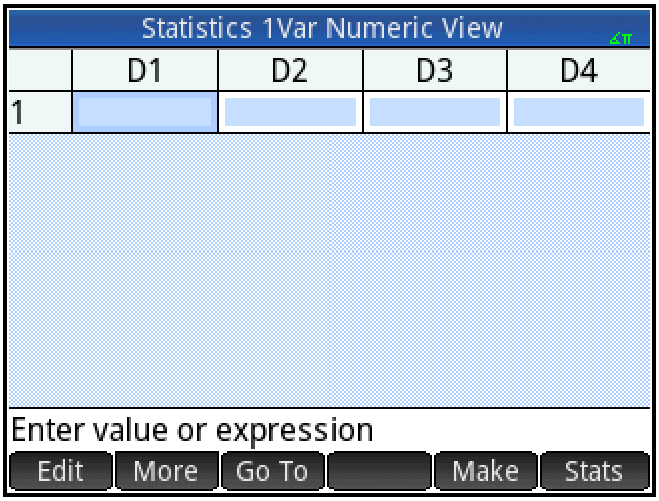
A series of six experiments gives the following values as the boiling point of a liquid:

82.5, 83.1, 82.6, 83.7, 82.4, and 83.0

Based on this sample, we want to estimate the true boiling point at the 90% confidence level.

Opening the Statistics 1Var app

- Press Apps Info and then select **Statistics 1Var**.



Statistics 1Var Numeric View

	D1	D2	D3	D4
1				

Enter value or expression

Clearing unwanted data

- If there is unwanted data in the app, clear it:

  Press Shift Esc Clear , and then select **All columns**.

Entering data

▲ In column D1, enter the boiling points found during the experiments.

82 . 5 Enter

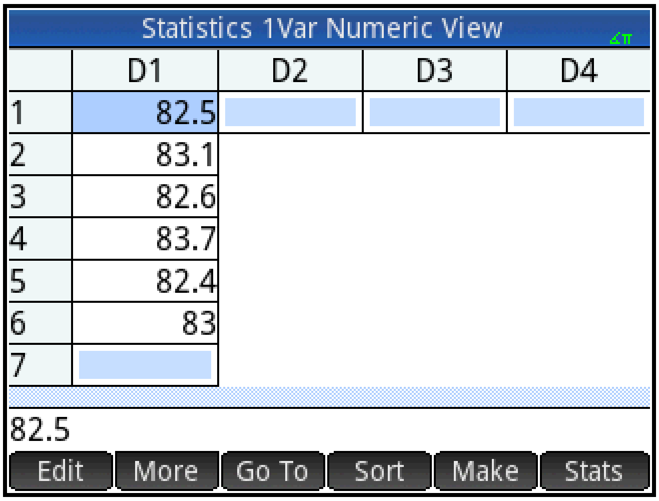
83 . 1 Enter

82 . 6 Enter

83 . 7 Enter

82 . 4 Enter

83 Enter



Statistics 1Var Numeric View

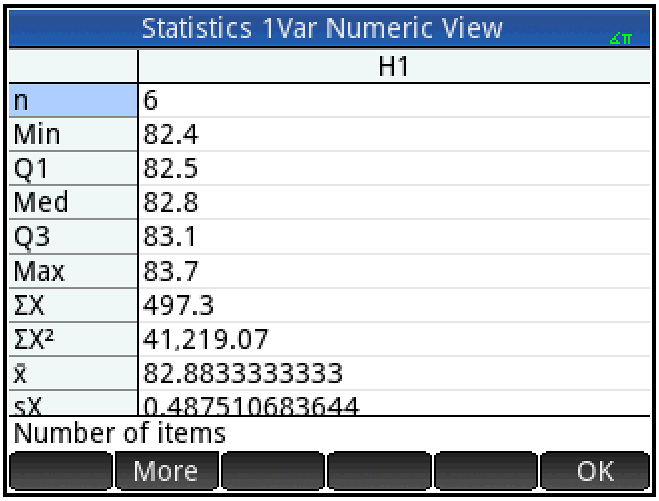
	D1	D2	D3	D4
1	82.5			
2	83.1			
3	82.6			
4	83.7			
5	82.4			
6	83			
7				

82.5

Calculating the statistics

1. Tap Stats.

   The statistics calculated will now be imported into the Inference app.



Statistics 1Var Numeric View

	H1
n	6
Min	82.4
Q1	82.5
Med	82.8
Q3	83.1
Max	83.7
ΣX	497.3
ΣX²	41,219.07
x̄	82.8833333333
sX	0.487510683644

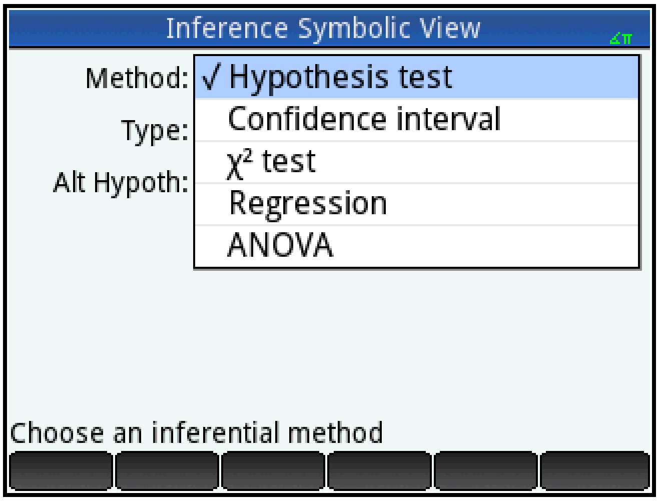
Number of items

2. Tap OK to close the statistics window.

Opening the Inference app

▲ Open the Inference app and clear the current settings.

   Press Apps Info, select **Inference**, and then press Shift Esc Clear .



Inference Symbolic View

Method: ✓ Hypothesis test

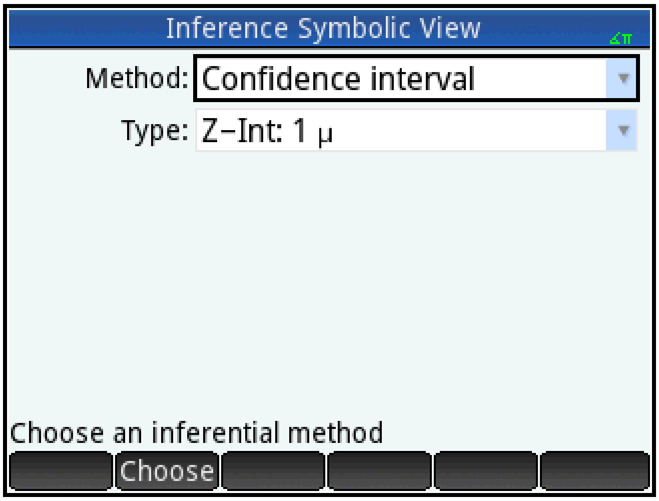
Type:

Alt Hypoth:

Choose an inferential method

Selecting the inference method and type

1. Select **Method**, and then select **Confidence interval**.



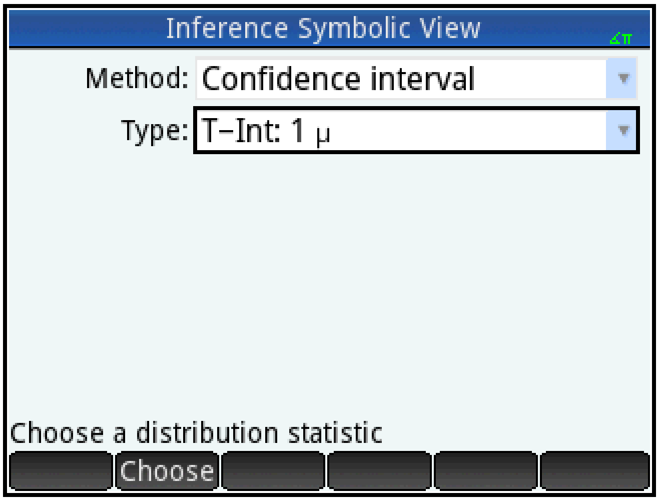
Inference Symbolic View

Method: Confidence interval

Type: Z–Int: 1 μ

Choose an inferential method

2. Select **Type**, and then select **T-Int: 1 μ**.



Inference Symbolic View

Method: Confidence interval

Type: T–Int: 1 μ

Choose a distribution statistic

Importing the data

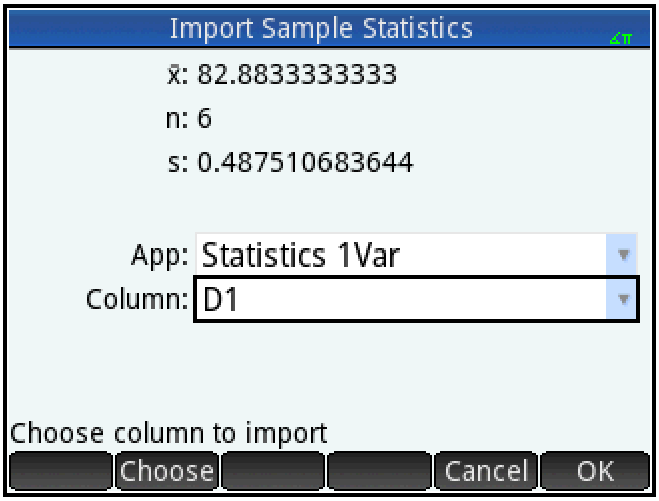
1. Press Num ↳Setup .

2. Specify the data you want to import:

   Tap Import .

3. In the **App** field select the statistics app that has the data you want to import.

4. In the **Column** field specify the column in that app where the data is stored. (D1 is the default.)



Import Sample Statistics

x̄: 82.8833333333

n: 6

s: 0.487510683644

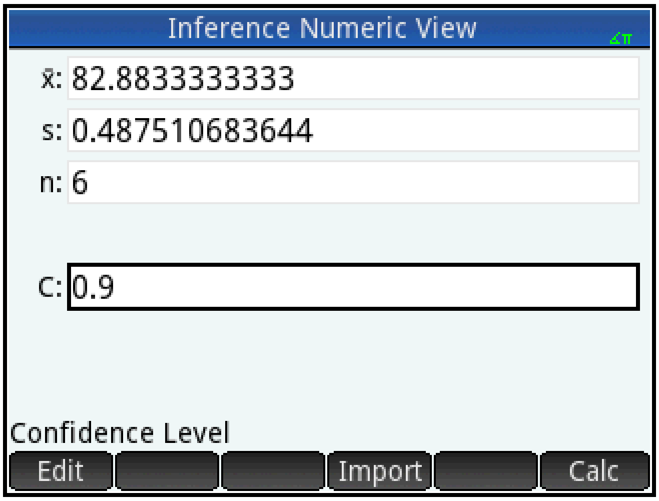
App: Statistics 1Var

Column: D1

Choose column to import

5. Tap OK.

6. Specify a 90% confidence interval in the **C** field.



Inference Numeric View

x̄: 82.8833333333

s: 0.487510683644

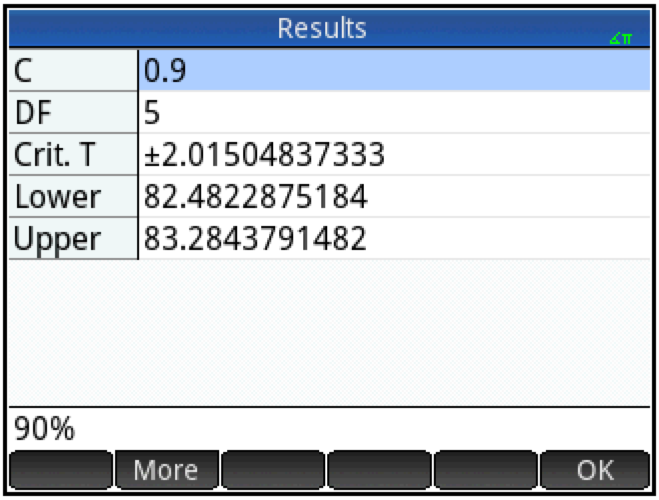
n: 6

C: 0.9

Confidence Level

Displaying results numerically

1. To display the confidence interval in Numeric view, tap Calc.



Results

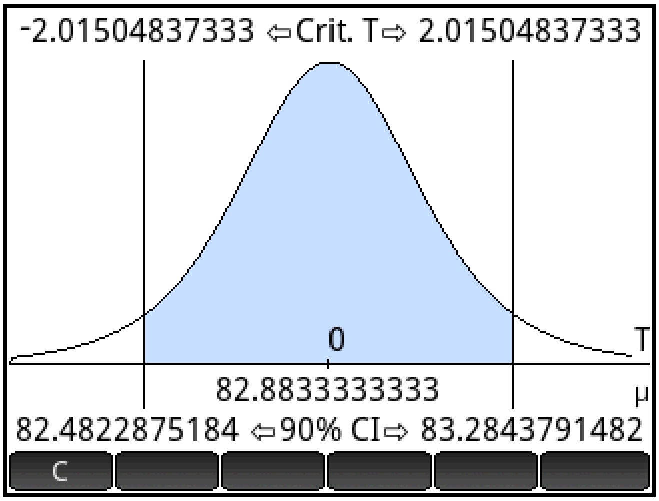
C	0.9
DF	5
Crit. T	±2.01504837333
Lower	82.4822875184
Upper	83.2843791482

90%

2. Tap OK to return to Numeric view.

Displaying results graphically

- To display the confidence interval in Plot view, press Plot Setup.



-2.01504837333 ⇦Crit. T⇨ 2.01504837333

82.8833333333

82.4822875184 ⇦90% CI⇨ 83.2843791482

The 90% confidence interval is [82.48..., 83.28...].

Hypothesis tests

You use hypothesis tests to test the validity of hypotheses about the statistical parameters of one or two populations. The tests are based on statistics of samples of the populations.

The HP Prime hypothesis tests use the Normal Z-distribution or the Student's t-distribution to calculate probabilities. If you wish to use other distributions, please use the Home view and the distributions found within the Probability category of the Math menu.

One-Sample Z-Test

Menu name

Z-Test: 1 μ

On the basis of statistics from a single sample, this test measures the strength of the evidence for a selected hypothesis against the null hypothesis. The null hypothesis is that the population mean equals a specified value, H_0: $\mu = \mu_0$.

You select one of the following alternative hypotheses against which to test the null hypothesis:

- H_0: $\mu < \mu_0$
- H_0: $\mu > \mu_0$
- H_0: $\mu \neq \mu_0$

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}$	Sample mean
n	Sample size
μ_0	Hypothetical population mean
σ	Population standard deviation
α	Significance level

Results

The results are as follows:

Result	Description
Test Z	Z-test statistic
Test $\dot{x}$	Value of $\dot{x}$ associated with the test Z-value
P	Probability associated with the Z-Test statistic
Critical Z	Boundary value(s) of Z associated with the α level that you supplied
Critical $\dot{x}$	Boundary value(s) of $\dot{x}$ required by the α value that you supplied

Two-Sample Z-Test

Menu name

Z-Test: $\mu_1 - \mu_2$

On the basis of two samples, each from a separate population, this test measures the strength of the evidence for a selected hypothesis against the null hypothesis. The null hypothesis is that the means of the two populations are equal, H_0: $\mu_1 = \mu_2$.

You can select one of the following alternative hypotheses to test against the null hypothesis:

- H_0: $\mu_1 < \mu_2$
- H_0: $\mu_1 > \mu_2$
- H_0: $\mu_1 \neq \mu_2$

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}_1$	Sample 1 mean
$\dot{x}_2$	Sample 2 mean
n_1	Sample 1 size
n_2	Sample 2 size
σ_1	Population 1 standard deviation
σ_2	Population 2 standard deviation
α	Significance level

Results

The results are as follows:

Result	Description
Test Z	Z-Test statistic
Test $\Delta\dot{x}$	Difference in the means associated with the test Z-value
P	Probability associated with the Z-Test statistic
Critical Z	Boundary value(s) of Z associated with the α level that you supplied
Critical $\Delta\dot{x}$	Difference in the means associated with the α level you supplied

One-Proportion Z-Test

Menu name

Z-Test: 1 π

On the basis of statistics from a single sample, this test measures the strength of the evidence for a selected hypothesis against the null hypothesis. The null hypothesis is that the proportion of successes is an assumed value, H_0: $\pi = \pi_0$.

You select one of the following alternative hypotheses against which to test the null hypothesis:

- H_0: $\pi < \pi_0$
- H_0: $\pi > \pi_0$
- H_0: $\pi \neq \pi_0$

Inputs

The inputs are as follows:

Field name	Description
x	Number of successes in the sample
n	Sample size
π_0	Population proportion of successes
α	Significance level

Results

The results are as follows:

Result	Description
Test Z	Z-Test statistic
Test $\hat{p}$	Proportion of successes in the sample
P	Probability associated with the Z-Test statistic
Critical Z	Boundary value(s) of Z associated with the α level that you supplied
Critical $\hat{p}$	Proportion of successes associated with the level you supplied

Two-Proportion Z-Test

Menu name

Z-Test: $\pi_1 - \pi_2$

On the basis of statistics from two samples, each from a different population, this test measures the strength of the evidence for a selected hypothesis against the null hypothesis. The null hypothesis is that the proportions of successes in the two populations are equal, H_0: $\pi_1 = \pi_2$.

You select one of the following alternative hypotheses against which to test the null hypothesis:

- H_0: $\pi_1 < \pi_2$
- H_0: $\pi_1 > \pi_2$
- H_0: $\pi_1 \neq \pi_2$

Inputs

The inputs are as follows:

Field name	Description
x_1	Sample 1 success count
x_2	Sample 2 success count
n_1	Sample 1 size
n_2	Sample 2 size
α	Significance level

Results

The results are as follows:

Results	Description
Test Z	Z-Test statistic
Test $\Delta\hat{p}$	Difference between the proportions of successes in the two samples that is associated with the test Z-value
P	Probability associated with the Z-Test statistic
Critical Z	Boundary value(s) of Z associated with the α level that you supplied
Critical $\Delta\hat{p}$	Difference in the proportion of successes in the two samples associated with the α level you supplied

One-Sample T-Test

Menu name

T-Test: 1 μ

This test is used when the population standard deviation is not known. On the basis of statistics from a single sample, this test measures the strength of the evidence for a selected hypothesis against the null hypothesis. The null hypothesis is that the sample mean has some assumed value, H_0: $\mu = \mu_0$.

You select one of the following alternative hypotheses against which to test the null hypothesis:

- H_0: $\mu < \mu_0$
- H_0: $\mu > \mu_0$
- H_0: $\mu \neq \mu_0$

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}$	Sample mean
s	Sample standard deviation
n	Sample size
μ_0	Hypothetical population mean
α	Significance level

Results

The results are as follows:

Results	Description
Test T	T-Test statistic
Test $\dot{x}$	Value of $\dot{x}$ associated with the test t-value
P	Probability associated with the T-Test statistic
DF	Degrees of freedom
Critical T	Boundary value(s) of T associated with the α level that you supplied
Critical $\dot{x}$	Boundary value(s) of $\dot{x}$ required by the α value that you supplied

Two-Sample T-Test

Menu name

T-Test: $\mu_1 - \mu_2$

This test is used when the population standard deviation is not known. On the basis of statistics from two samples, each sample from a different population, this test measures the strength of the evidence for a selected hypothesis against the null hypothesis. The null hypothesis is that the two populations means are equal, H_0: $\mu_1 = \mu_2$.

You select one of the following alternative hypotheses against which to test the null hypothesis:

- H_0: $\mu_1 < \mu_2$
- H_0: $\mu_1 > \mu_2$
- H_0: $\mu_1 \neq \mu_2$

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}_1$	Sample 1 mean
$\dot{x}_2$	Sample 2 mean
s_1	Sample 1 standard deviation
s_2	Sample 2 standard deviation
n_1	Sample 1 size
n_2	Sample 2 size
α	Significance level
Pooled	Select this option to pool samples based on their standard deviations

Results

The results are as follows:

Results	Description
Test T	T-Test statistic
Test Δẋ	Difference in the means associated with the test t-value
P	Probability associated with the T-Test statistic
DF	Degrees of freedom
Critical T	Boundary values of T associated with the α level that you supplied
Critical Δẋ	Difference in the means associated with the α level you supplied

Confidence intervals

The confidence interval calculations that the HP Prime can perform are based on the Normal Z-distribution or Student’s t-distribution.

One-Sample Z-Interval

Menu name

Z-Int: 1 μ

This option uses the Normal Z-distribution to calculate a confidence interval for μ, the true mean of a population, when the true population standard deviation, σ, is known.

Inputs

The inputs are as follows:

Field name	Description
ẋ	Sample mean
n	Sample size
σ	Population standard deviation
C	Confidence level

Results

The results are as follows:

Result	Description
C	Confidence level
Critical Z	Critical values for Z

Result	Description
Lower	Lower bound for μ
Upper	Upper bound for μ

Two-Sample Z-Interval

Menu name

Z-Int: $\mu_1 - \mu_2$

This option uses the Normal Z-distribution to calculate a confidence interval for the difference between the means of two populations, $\mu_1 - \mu_2$, when the population standard deviations, σ_1 and σ_2, are known.

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}_1$	Sample 1 mean
$\dot{x}_2$	Sample 2 mean
n_1	Sample 1 size
n_2	Sample 2 size
σ_1	Population 1 standard deviation
σ_2	Population 2 standard deviation
C	Significance level

Results

The results are as follows:

Result	Description
C	Confidence level
Critical Z	Critical values for Z
Lower	Lower bound for Δμ
Upper	Upper bound for Δμ

One-Proportion Z-Interval

Menu name

Z-Int: 1 π

This option uses the Normal Z-distribution to calculate a confidence interval for the proportion of successes in a population for the case in which a sample of size n has a number of successes x.

Inputs

The inputs are as follows:

Field name	Description
x	Sample success count
n	Sample size
C	Confidence level

Results

The results are as follows:

Result	Description
C	Confidence level
Critical Z	Critical values for Z
Lower	Lower bound for π
Upper	Upper bound for π

Two-Proportion Z-Interval

Menu name

Z-Int: $\pi_1 - \pi_2$

This option uses the Normal Z-distribution to calculate a confidence interval for the difference between the proportions of successes in two populations.

Inputs

The inputs are as follows:

Field name	Description
x_1	Sample 1 success count
x_2	Sample 2 success count
n_1	Sample 1 size

Field name	Description
n_2	Sample 2 size
C	Confidence level

Results

The results are as follows:

Results	Description
C	Confidence level
Critical Z	Critical values for Z
Lower	Lower bound for $\Delta\pi$
Upper	Upper bound for $\Delta\pi$

One-Sample T-Interval

Menu name

T-Int: 1 μ

This option uses the Student’s t-distribution to calculate a confidence interval for μ, the true mean of a population, for the case in which the true population standard deviation, σ, is unknown.

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}$	Sample mean
s	Sample standard deviation
n	Sample size
C	Confidence level

Results

The results are as follows:

Results	Description
C	Confidence level
DF	Degrees of freedom
Critical	Critical values for T

Results	Description
Lower	Lower bound for μ
Upper	Upper bound for μ

Two-Sample T-Interval

Menu name

T-Int: $\mu_1 - \mu_2$

This option uses the Student's t-distribution to calculate a confidence interval for the difference between the means of two populations, $\mu_1 - \mu_2$, when the population standard deviations, σ_1 and σ_2, are unknown.

Inputs

The inputs are as follows:

Field name	Description
$\dot{x}_1$	Sample 1 mean
$\dot{x}_2$	Sample 2 mean
s_1	Sample 1 standard deviation
s_2	Sample 2 standard deviation
n_1	Sample 1 size
n_2	Sample 2 size
C	Confidence level
Pooled	Whether or not to pool the samples based on their standard deviations

Results

The results are as follows:

Results	Description
C	Confidence level
DF	Degrees of freedom
Critical T	Critical values for T
Lower	Lower bound for Δμ
Upper	Upper bound for Δμ

Chi-square tests

An HP Prime calculator can perform tests on categorical data based on the chi-square distribution. Specifically, HP Prime calculators support both goodness of fit tests and tests on two-way tables.

Goodness of fit test

Menu name

Goodness of Fit

This option uses the chi-square distribution to test the goodness of fit of categorical data on observed counts against either expected probabilities or expected counts. In the Symbolic view, make your selection in the **Expected** box: choose either **Probability** (the default) or **Count**.

Inputs

With **Expected Probability** selected, the Numeric view inputs are as follows:

Field name	Description
ObsList	The list of observed count data
ProbList	The list of expected probabilities

Results

When Calc is tapped, the results are as follows:

Results	Description
x^2	The value of the chi-square test statistic
P	The probability associated with the chi-square value
DF	The degrees of freedom

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
Stats	Displays the default test results, as listed previously.
Exp	Displays the expected counts.
Cont	Displays the list of contributions of each category to the chi-square value.
OK	Returns to the Numeric view.

With Expected Count selected, the Numeric view inputs include ExpList for the expected counts instead of ProbList and the menu key labels in the Results screen do not include Exp.

Two-way table test

Menu name

2-way test

This option uses the chi-square distribution to test the goodness of fit of categorical data of observed counts contained in a two-way table.

Inputs

The Numeric view inputs are as follows:

Field name	Description
ObsMat	The matrix of the observed count data in the two-way table

Results

When Calc is tapped, the results are as follows:

Results	Description
x^2	The value of the chi-square test statistic
P	The probability associated with the chi-square value
DF	The degrees of freedom

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
Exp	Displays the matrix of expected counts. Press OK to exit.
Cont	Displays the matrix of contributions of each category to the chi-square value. Press OK to exit.
OK	Returns to the Numeric view.

Inference for regression

An HP Prime calculator can perform tests and calculate intervals based on inference for linear regression. These calculations are based on the t-distribution.

Linear t-test

Menu name

Linear t test

This option performs a t-test on the true linear regression equation, based on a list of explanatory data and a list of response data. You must choose an alternative hypothesis in Symbolic view using the **Alt Hypoth** field.

Inputs

The Numeric view inputs are as follows:

Field name	Description
Xlist	The list of explanatory data
Ylist	The list of response data

Results

When Calc is tapped, the results are as follows:

Results	Description
Test T	The value of the t-test statistic
P	The probability associated with the t-test statistic
DF	The degrees of freedom
β_0	The intercept of the calculated regression line
β_1	The slope of the calculated regression line
serrLine	The standard error of the calculated regression line
serrSlope	The standard error of the slope of the calculated regression line
serrInter	The standard error of the intercept of the calculated regression line
r	The correlation coefficient of the data
R^2	The coefficient of determination of the data

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
OK	Returns to the Numeric view.

Confidence interval for slope

Menu name

Interval: Slope

This option calculates a confidence interval for the slope of the true linear regression equation, based on a list of explanatory data, a list of response data, and a confidence level. After you enter your data in Numeric view and tap Calc, enter the confidence level in the prompt that appears.

Inputs

The Numeric view inputs are as follows:

Field name	Description
Xlist	The list of explanatory data
Ylist	The list of response data
C	The confidence level (0 < C < 1)

Results

When Calc is tapped, the results are as follows:

Results	Description
C	The input confidence level
Crit. T	The critical value of t
DF	The degrees of freedom
β_1	The slope of the calculated regression line
serrSlope	The standard error of the slope of the calculated regression line
Lower	The lower bound of the confidence interval for the slope
Upper	The upper bound of the confidence interval for the slope

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
OK	Returns to the Numeric view.

Confidence interval for intercept

Menu name

Interval: Intercept

This option calculates a confidence interval for the intercept of the true linear regression equation, based on a list of explanatory data, a list of response data, and a confidence level. After you enter your data in Numeric view and tap Calc, enter the confidence level in the prompt that appears.

Inputs

The Numeric view inputs are as follows:

Field name	Description
Xlist	The list of explanatory data
Ylist	The list of response data
C	The confidence level (0 < C < 1)

Results

When Calc is tapped, the results are as follows:

Results	Description
C	The input confidence level
Crit. T	The critical value of t
DF	The degrees of freedom
β_0	The intercept of the calculated regression line
serrInter	The standard error of the y-intercept of the regression line
Lower	The lower bound of the confidence interval for the intercept
Upper	The upper bound of the confidence interval for the intercept

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
OK	Returns to the Numeric view.

Confidence interval for a mean response

Menu name

Interval: Mean response

This option calculates a confidence interval for the mean response (ŷ), based on a list of explanatory data, a list of response data, a value of the explanatory variable (X), and a confidence level. After you enter your data in Numeric view and tap Calc, enter the confidence level and the value of the explanatory variable (X) in the prompt that appears.

Inputs

The Numeric view inputs are as follows:

Field name	Description
Xlist	The list of explanatory data
Ylist	The list of response data
X	The value of the explanatory variable for which you want a mean response and a confidence interval
C	The confidence level (0 < C < 1)

Results

When Calc is tapped, the results are as follows:

Results	Description
C	The input confidence level
Crit. T	The critical value of t
DF	The degrees of freedom
ŷ	The mean response for the input X-value
serrŷ	The standard error of ŷ
Lower	The lower bound of the confidence interval for the mean response
Upper	The upper bound of the confidence interval for the mean response

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
OK	Returns to the Numeric view.

Prediction interval

Menu name

Prediction interval

This option calculates a prediction interval for a future response, based on a list of explanatory data, a list of response data, a value of the explanatory variable (X), and a confidence level. After you enter your data in Numeric view and tap Calc, enter the confidence level and the value of the explanatory variable (X) in the prompt that appears.

Inputs

The Numeric view inputs are as follows:

Field name	Description
Xlist	The list of explanatory data
Ylist	The list of response data
X	The value of the explanatory variable for which you want a future response and a confidence interval
C	The confidence level (0 < C < 1)

Results

When Calc is tapped, the results are as follows:

Results	Description
C	The input confidence level
Crit. T	The critical value of t
DF	The degrees of freedom
ŷ	The future response for the input X-value
serrŷ	The standard error of ŷ
Lower	The lower bound of the confidence interval for the mean response
Upper	The upper bound of the confidence interval for the mean response

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
OK	Returns to the Numeric view.

ANOVA

Menu name

ANOVA

This option performs a one-way analysis of variance (ANOVA) using an F-test, based on lists of numerical data.

Inputs

The inputs for the one-way ANOVA are lists of data in l1-l4. You can add additional lists in l5 and so on.

Results

When Calc is tapped, the results are as follows:

Results	Description
F	The F-value of the test
P	The probability associated with the F-value of the test
DF	The degrees of freedom for the test
SS	The sum of the squares of the treatments
MS	The mean square of the treatments
DFerr	The degrees of freedom of the errors
SSerr	The sum of the squares of the errors
MSerr	The mean square of the errors

Menu keys

The menu key options are as follows:

Menu key	Description
More	Opens a menu that enables you to select multiple cells to copy and paste.
OK	Returns to the Numeric view.

Use the cursor keys or tap to move about the table. In addition to tapping More, you can tap and hold on a cell, and then drag your finger to select a rectangular array of cells to copy and paste.

14 Solve app

The Solve app enables you to define up to ten equations or expressions each with as many variables as you like. You can solve a single equation or expression for one of its variables, based on a seed value. You can also solve a system of equations (linear or non-linear), again using seed values.

Note the following differences between an equation and an expression:

- An equation contains an equal sign. Its solution is a value for the unknown variable that makes both sides of the equation have the same value.
- An expression does not contain an equal sign. Its solution is a root, a value for the unknown variable that makes the expression have a value of zero.

For brevity, the term equation in this chapter will cover both equations and expressions.

Solve works only with real numbers.

Getting started with the Solve app

The Solve app uses the customary app views: Symbolic, Plot, and Numeric view; however, the Numeric view is significantly different from the other apps as it is dedicated to numerical solving rather than to displaying a table of values.

The Symbolic view and Plot view menu buttons are available in this app.

One equation

Suppose you want to find the acceleration needed to increase the speed of a car from 16.67 m/s (60 kph) to 27.78 m/s (100 kph) over a distance of 100 m.

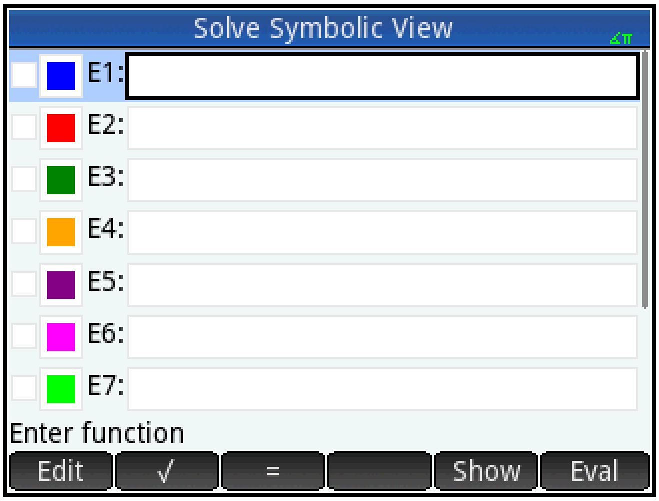
The following is the equation to solve is:

$V^2 = U^2 + 2AD$

In this equation, V = final speed, U = initial speed, A = acceleration needed, and D = distance.

Opening the Solve app

▲ Press Apps Info and then select **Solve**.



The Solve app starts in Symbolic view, where you can specify the equation to solve.

NOTE: In addition to the built-in variables, you can use one or more variables you created yourself (either in Home view or in the CAS). For example, if you have created a variable called ME, you could include it in an equation such as this: $Y^2 = G^2 + ME$.

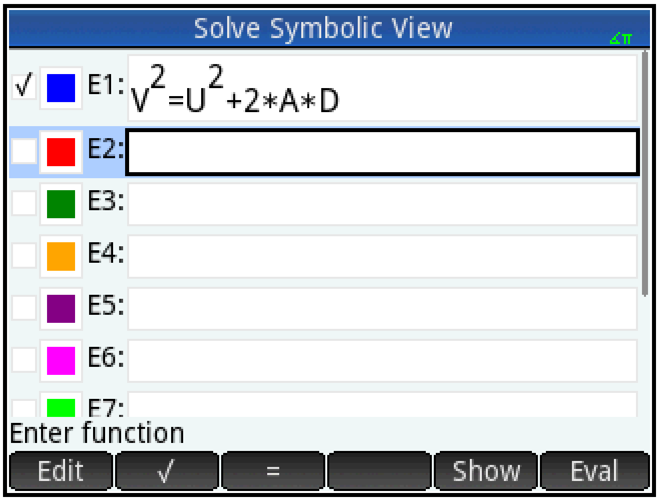
Functions defined in other apps can also be referenced in the Solve app. For example, if you have defined F1(X) to be $X^2 + 10$ in the Function app, you can enter F1(X) = 50 in the Solve app to solve the equation $X^2 + 10 = 50$.

Clearing the app and defining the equation

1. If you have no need for any equations or expressions already defined, press Shift Esc Clear . Tap OK to confirm your intention to clear the app.

2. Define the equation.

ALPHA V x^2 Shift . ALPHA U x^2 + 2 ALPHA A ALPHA D Enter



Solve Symbolic View

✓ 🟦 E1: $V^2=U^2+2*A*D$

□ 🟥 E2:

□ 🟩 E3:

□ 🟧 E4:

□ 🟪 E5:

□ 🟪 E6:

□ 🟩 E7:

Enter function

Edit | √ | = | | Show | Eval

Entering known variables

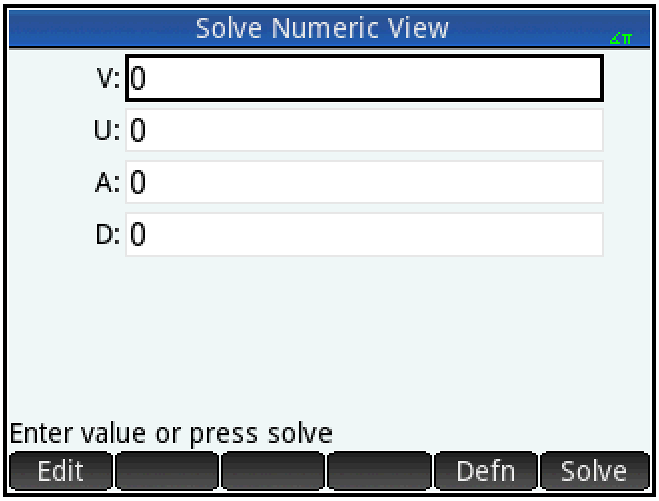
1. Display the Numeric view.

Num Setup

Here you specify the values of the known variables, highlight the variable that you want to solve for, and then tap Solve.

2. Enter the values for the known variables.

27 . 78 Enter 16 . 67 Enter ▼ 100 Enter



Solve Numeric View

V: 0

U: 0

A: 0

D: 0

Enter value or press solve

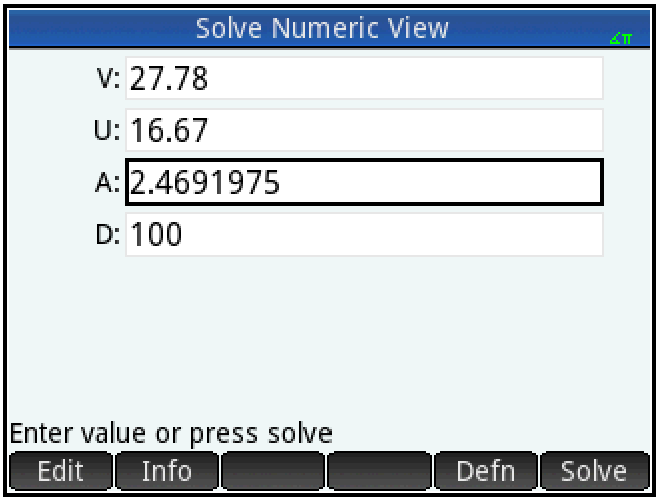
Edit | | | | Defn | Solve

NOTE: Some variables may already have values against them when you display the Numeric view. This occurs if the variables have been assigned values elsewhere. For example, in Home view you might have assigned 10 to variable U by entering `10`, tapping Sto ▸, and then entering `U`. Then when you open the Numeric view to solve an equation with U as a variable, 10 will be the default value for U. This also occurs if a variable has been given a value in some previous calculation (in an app or program).

To reset all prepopulated variables to zero, press Shift Esc Clear .

Solving for the unknown variable

▲ To solve for the unknown variable A, move the cursor to the **A** box and tap Solve.



Solve Numeric View

V: 27.78

U: 16.67

A: 2.4691975

D: 100

Enter value or press solve

Therefore, the acceleration needed to increase the speed of a car from 16.67 m/s (60 kph) to 27.78 m/s (100 kph) over a distance of 100 m is approximately 2.4692 m/s^2.

The equation is linear with respect to the variable A. Hence you can conclude that there are no further solutions for A. This is also visible if you plot the equation.

Plotting the equation

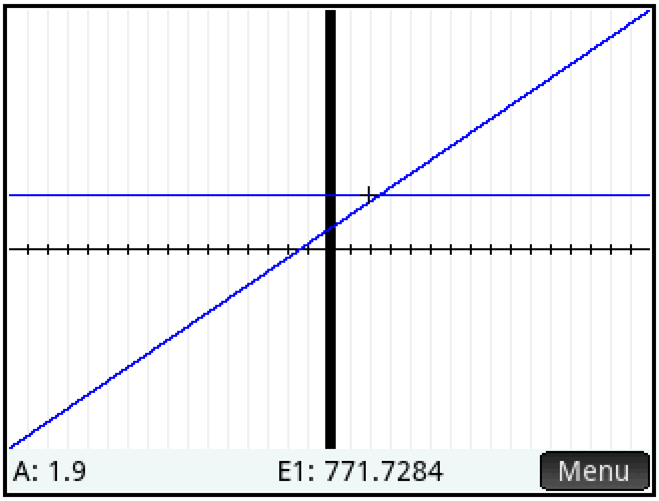
The Plot view shows one graph for each side of the solved equation. You can choose any of the variables to be the independent variable by selecting it in Numeric view. So in this example make sure that A is highlighted.

The current equation is $V^2 = U^2 + 2AD$. The plot view will plot two equations, one for each side of the equation. One of these is $Y = V^2$, with V = 27.78, making Y = 771.7284. This graph is a horizontal line. The other graph is $Y = U^2 + 2AD$ with U = 16.67 and D = 100, making, Y = 200A + 277.8889. This graph is also a line. The desired solution is the value of A where these two lines intersect.

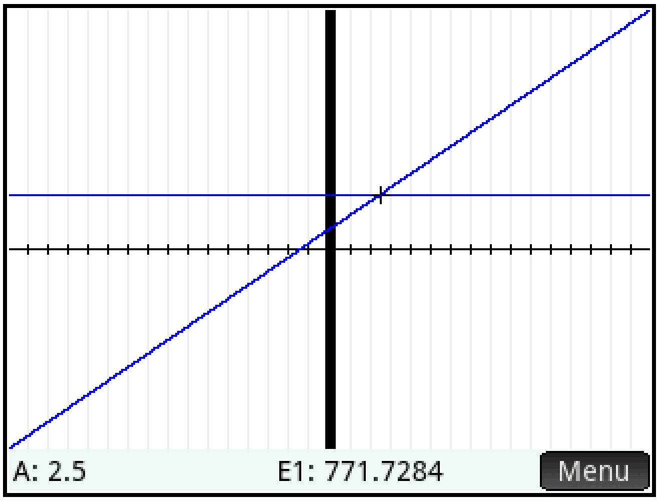
1. To plot the equation for variable A, press View Copy .

2. Select **Auto Scale**.

3. Select **Both sides of En** (where n is the number of the selected equation).



4. By default, the tracer is active. Using the cursor keys, move the trace cursor along either graph until it nears the intersection. Note that the value of A displayed near the bottom left corner of the screen closely matches the value of A that you calculated.



The Plot view provides a convenient way to find an approximation to a solution when you suspect that there are a number of solutions. Move the trace cursor close to the solution (that is, the intersection) of interest to you and then open Numeric view. The solution given in Numeric view is the solution nearest the trace cursor.

NOTE: By dragging a finger horizontally or vertically across the screen, you can quickly see parts of the plot that are initially outside the x and y ranges you set.

Several equations

You can define up to ten equations and expressions in Symbolic view and select those you want to solve together as a system. For example, suppose you want to solve the system of equations consisting of the following:

- $X^2 + Y^2 = 16$
- $X – Y = –1$

Opening the solve app

1. Press Apps and then select **Solve**.

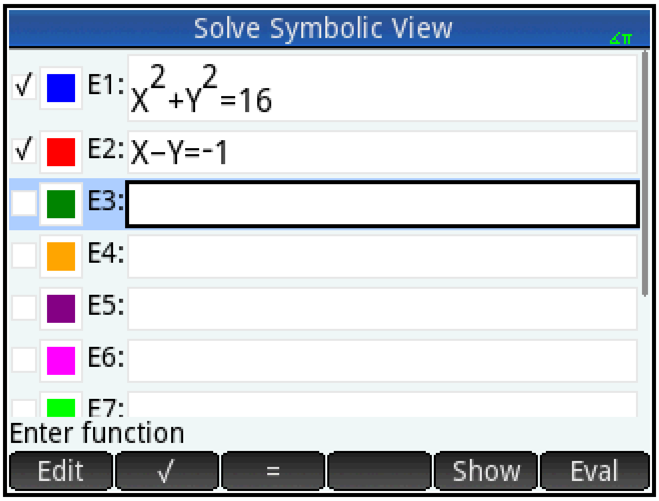
2. If you have no need for any equations or expressions already defined, press Shift Esc. Tap OK to confirm your intention to clear the app.

Defining the equations

▲ Define the equations.

ALPHA X x^2 + ALPHA Y x^2 Shift = 16 Enter

ALPHA X − ALPHA Y Shift = +/− 1 Enter



Solve Symbolic View

✓	🟦	E1: $X^2+Y^2=16$
✓	🟥	E2: X−Y=-1
□	🟩	E3:
□	🟧	E4:
□	🟪	E5:
□	■	E6:
□	🟩	E7:

Enter function

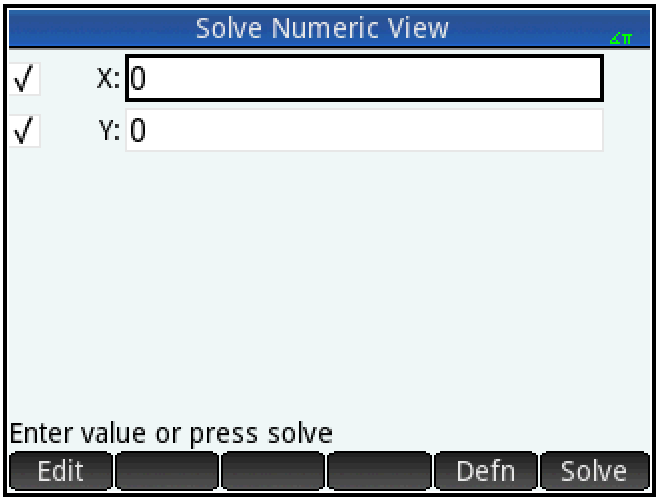
Make sure that both equations are selected, as we are looking for values of X and Y that satisfy both equations.

Entering a seed value

1. Display Numeric view.

   Num

   Unlike the one-equation example, in this example there are no given values for any variable. You can either enter a seed value for one of the variables, or let the calculator provide a solution. (Typically, a seed value is a value that directs the calculator to provide, if possible, a solution that is closest to it rather than some other value.) In this example, look for a solution in the vicinity of X = 2.


Solve Numeric View

√ X: 0

√ Y: 0

Enter value or press solve

2. Enter the seed value in the X field.

   For example, enter 2 and then tap OK.

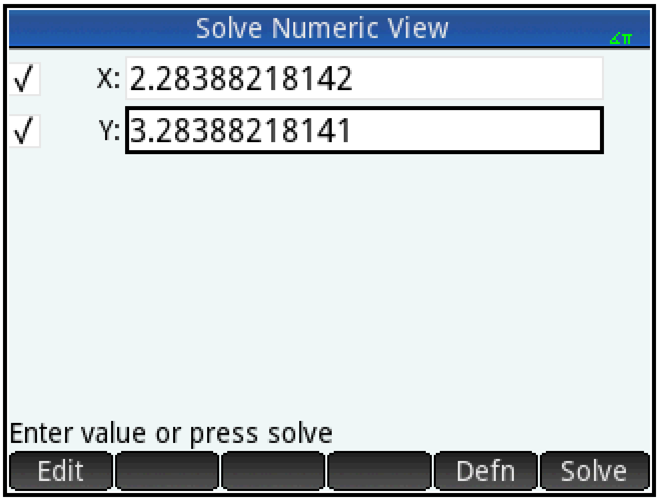
   The calculator provides one solution (if there is one) and you will not be alerted if there are multiple solutions. Vary the seed values to find other potential solutions.

3. Select the variables you want solutions for. In this example, you want to find values for both X and Y, so make sure that both variables are selected.

NOTE: If you have more than two variables, you can enter seed values for more than one of them.

Solving the unknown variables

▲ Tap Solve to find a solution near X = 2 that satisfies each selected equation.


Solve Numeric View

√ X: 2.28388218142

√ Y: 3.28388218141

Enter value or press solve

Solutions, if found, are displayed beside each selected variable.

Limitations

You cannot plot equations if more than one is selected in Symbolic view.

The HP Prime calculator does not alert you to the existence of multiple solutions. If you suspect that another solution exists close to a particular value, repeat the exercise using that value as a seed. (In the example just discussed, you will find another solution if you enter –4 as the seed value for X.)

In some situations, the Solve app uses a random number seed in its search for a solution. This means that it is not always predictable which seed leads to which solution when there are multiple solutions.

Solution information

When you are solving a single equation, the Info button appears on the menu after you tap Solve. Tapping Info displays a message giving you some information about the solutions found (if any). Tap OK to clear the message.

Message	Meaning
Zero	The Solve app found a point where both sides of the equation were equal, or where the expression was zero (a root), within the calculator's 12-digit accuracy.
Sign Reversal	Solve found two points where the two sides of the equation have opposite signs, but it cannot find a point in between where the value is zero. Similarly, for an expression, where the value of the expression has different signs but is not precisely zero. Either the two values are neighbors (they differ by one in the twelfth digit) or the equation is not real-valued between the two points. Solve returns the point where the value or difference is closer to zero. If the equation or expression is continuously real, this point is Solve’s best approximation of an actual solution.
Extremum	Solve found a point where the value of the expression approximates a local minimum (for positive values) or maximum (for negative values). This point may or may not be a solution. — or — Solve stopped searching at 9.99999999999E499, the largest number the calculator can represent. **NOTE:** The **Extremum** message indicates that it is highly likely that there is no solution. Use Numeric view to verify this (and note that any values shown are suspect).
Cannot find solution	No values satisfy the selected equation or expression.
Bad Guess(es)	The initial guess lies outside the domain of the equation. Therefore, the solution was not a real number or it caused an error.
Constant?	The value of the equation is the same at every point sampled.

15 Linear Solver app

The Linear Solver app enables you to solve a set of linear equations. The set can contain two or three linear equations.

In a two-equation set, each equation must be in the form ax + by = k. In a three-equation set, each equation must be in the form ax + by + cz = k.

You provide values for a, b, and k (and c in three-equation sets) for each equation, and the app will attempt to solve for x and y (and z in three-equation sets).

The HP Prime calculator alerts you if no solution can be found, or if there is an infinite number of solutions.

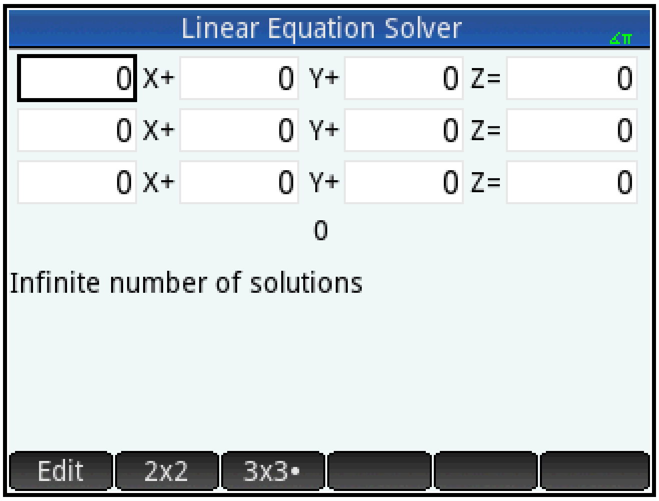
Getting started with the Linear Solver app

The following example defines the following set of equations and then solves for the unknown variables:

- 6x + 9y + 6z = 5
- 7x + 10y + 8z = 10
- 6x + 4y = 6

Opening the Linear Solver app

▲ Press Apps Info, and then select **Linear Solver**.



Linear Equation Solver

0 X+ 0 Y+ 0 Z= 0
0 X+ 0 Y+ 0 Z= 0
0 X+ 0 Y+ 0 Z= 0

0

Infinite number of solutions

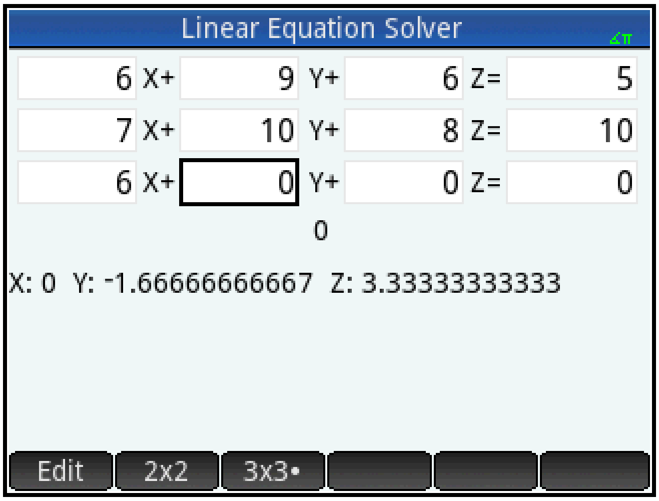
The app opens in Numeric view.

NOTE: If the last time you used the Linear Solver app you solved for two equations, the two-equation input form is displayed. To solve a three-equation set, tap 3x3; now the input form displays three equations.

Defining and solving the equations

1. You define the equations you want to solve by entering the coefficients of each variable in each equation and the constant term. Notice that the cursor is positioned immediately to the left of x in the first equation, ready for you to insert the coefficient of x (6). Enter the coefficient and either tap OK or press Enter.

2. The cursor moves to the next coefficient. Enter that co-efficient and either tap OK or press Enter. Continue doing likewise until you have defined all the equations.



Linear Equation Solver

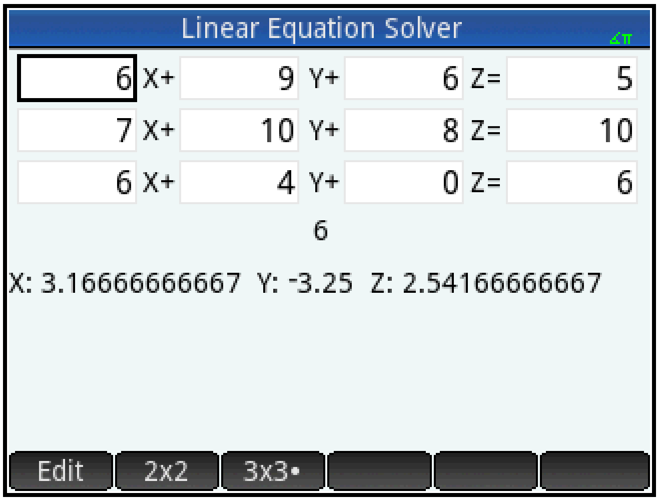
6 X+ 9 Y+ 6 Z= 5
7 X+ 10 Y+ 8 Z= 10
6 X+ 0 Y+ 0 Z= 0

0

X: 0 Y: -1.66666666667 Z: 3.33333333333

Once you have entered enough values for the solver to be able to generate solutions, those solutions appear near the bottom of the display. In this example, the solver was able to find solutions for x, y, and z as soon as the first coefficient of the last equation was entered.

As you enter each of the remaining known values, the solution changes. The following figure shows the final solution once all the coefficients and constants had been entered.



Linear Equation Solver

6 X+ 9 Y+ 6 Z= 5
7 X+ 10 Y+ 8 Z= 10
6 X+ 4 Y+ 0 Z= 6

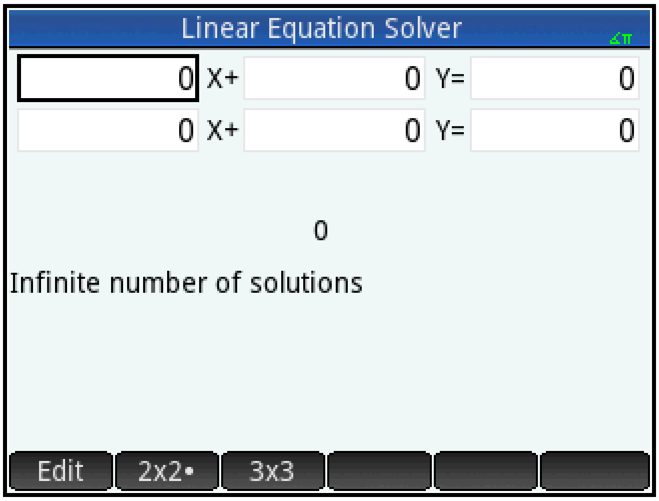
6

X: 3.16666666667 Y: -3.25 Z: 2.54166666667

Solving a two-by-two system

If the three-equation input form is displayed and you want to solve a two-equation set, do the following:

▲ Tap 2x2 .



0 X+ 0 Y= 0
0 X+ 0 Y= 0

0

Infinite number of solutions

NOTE: You can enter any expression that resolves to a numerical result, including variables. Just enter the name of a variable.

Menu items

The menu items are as follows:

Menu item	Description
Edit	Moves the cursor to the entry line where you can add or change a value. You can also highlight a field, enter a value, and press Enter . The cursor automatically moves to the next field, where you can enter the next value and press Enter .
2x2	Displays the page for solving a system of 2 linear equations in 2 variables; changes to 2x2• when active.
3x3	Displays the page for solving a system of 3 linear equations in 3 variables; changes to 3x3• when active.

16 Parametric app

The Parametric app enables you to explore parametric equations. These are equations in which both x and y are defined as functions of t. They take the forms x = f(t) and y = g(t).

Getting started with the Parametric app

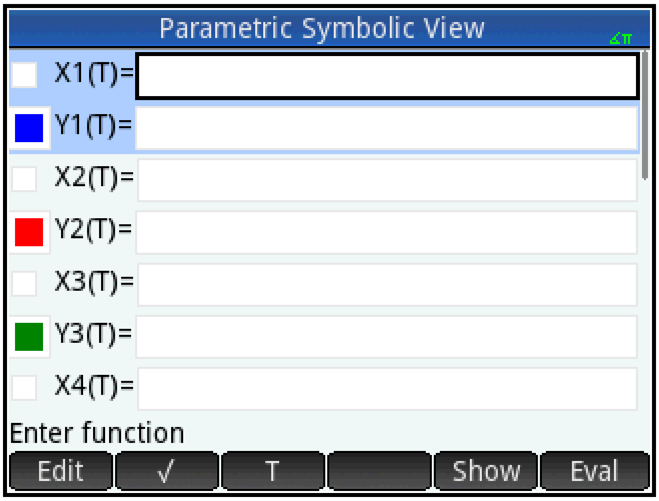
The Parametric app uses the customary app views: Symbolic, Plot, and Numeric view.

The Symbolic view, Plot view, and Numeric view menu buttons are available in this app.

Throughout this chapter, we will explore the parametric equations x(T) = 8sin(T) and y(T) = 8cos(T). These equations produce a circle.

Opening the Parametric app

▲ Press Apps Info , and then select **Parametric**.



Enter function

The Parametric app starts in Symbolic view. This is the defining view. It is where you symbolically define (that is, specify) the parametric expressions you want to explore.

The graphical and numerical data you see in Plot view and Numeric view are derived from the symbolic functions defined here.

Defining the functions

There are 20 fields for defining functions. These are labelled X1(T) through X9(T) and X0(T), and Y1(T) through Y9(T) and Y0(T). Each X function is paired with a Y function.

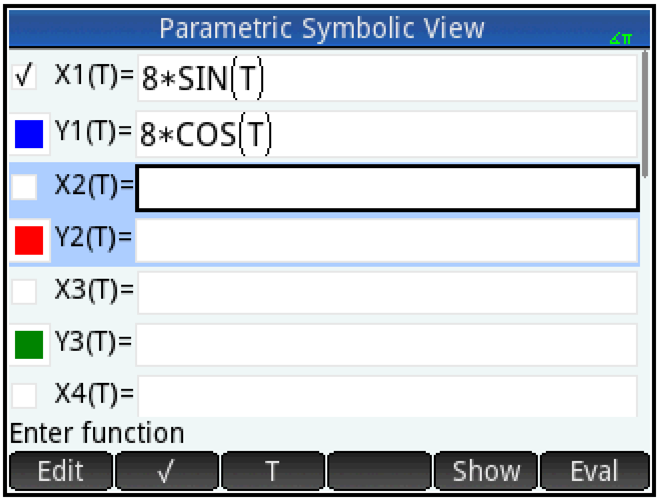
1. Highlight which pair of functions you want to use, either by tapping on, or scrolling to, one of the pair. If you are entering a new function, just start typing. If you are editing an existing function, tap Edit and make your changes. When you have finished defining or changing the function, press Enter ≈ .

2. Define the two expressions.

   8 SIN ASIN G xtθn Define D Enter ≈

   8 COS ACOS H xtθn Define D Enter ≈

   The xtθn Define D key enters whatever variable is relevant to the current app. In this app, it enters a T.



Parametric Symbolic View

✓ X1(T)= 8*SIN(T)

🟦 Y1(T)= 8*COS(T)

□ X2(T)=

🟥 Y2(T)=

□ X3(T)=

🟩 Y3(T)=

□ X4(T)=

Enter function

3. Decide to do one of the following:

   - Give one or more function a custom color when it is plotted
   - Evaluate a dependent function
   - Clear a definition that you don’t want to explore
   - Incorporate variables, math commands, and CAS commands in a definition

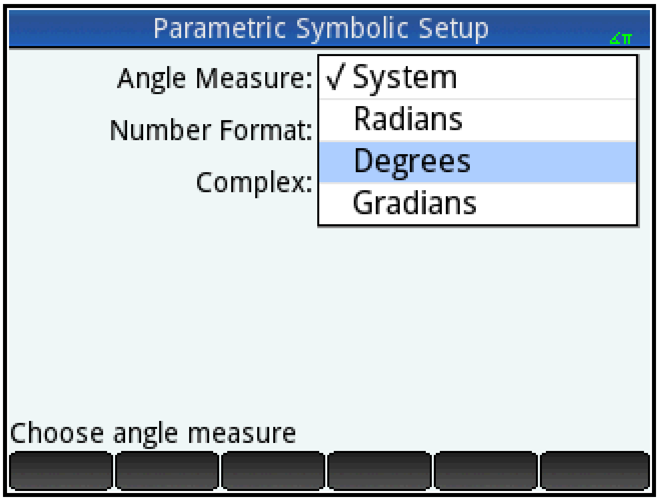
For the sake of simplicity we can ignore these operations in this example. However, they can be useful and are common Symbolic view operations.

Setting the angle measure

To set the angle measure to degrees:

1. Press Shift Symb Setup .

2. Select **Angle Measure**, and then select **Degrees**.



Parametric Symbolic Setup

Angle Measure: √ System

Radians

Degrees

Gradians

Number Format:

Complex:

Choose angle measure

You can also have set the angle measure on the **Home Settings** screen. However, Home settings are system-wide. By setting the angle measure in an app rather than Home view, you are limiting the setting just to that app.

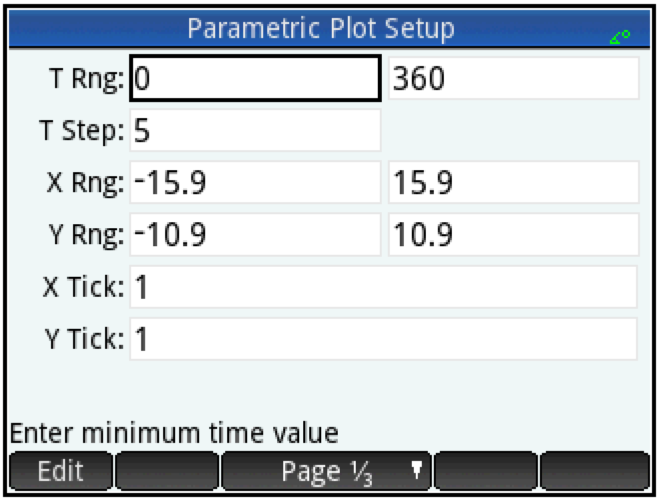
Setting up the plot

1. To open Plot Setup view, press Shift Plot Setup .

2. Set up the plot by specifying appropriate graphing options. In this example, set the **T Rng** and **T Step** fields so that T steps from 0° to 360° in 5° steps.

   Select the second **T Rng** field and enter:

   360 OK 5 OK



Parametric Plot Setup

T Rng: 0 360

T Step: 5

X Rng: -15.9 15.9

Y Rng: -10.9 10.9

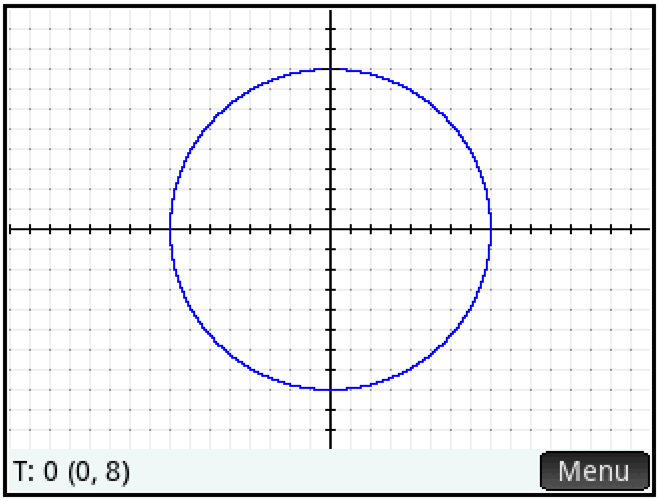
X Tick: 1

Y Tick: 1

Enter minimum time value

Plotting the functions

▲ Press Plot ↳Setup.



T: 0 (0, 8)

Exploring the graph

The menu button gives you access to the following common tools for exploring plots:

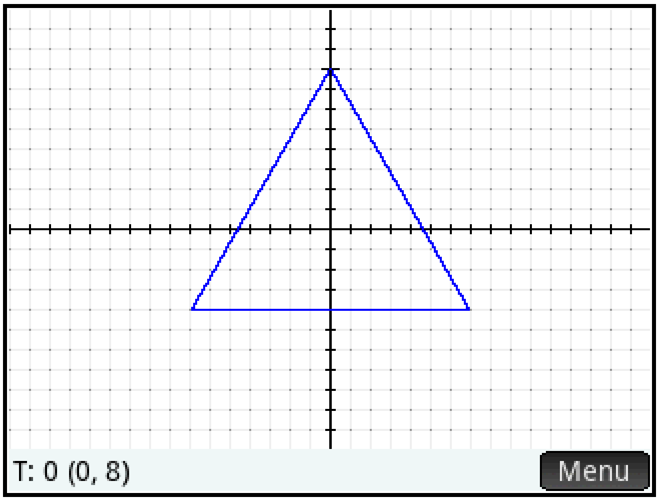
- Zoom—Displays a range of zoom options. (The + Ans and − Base keys can also be used to zoom in and out.)
- Trace•—When active, enables a tracing cursor to be moved along the contour of the plot (with the coordinates of the cursor displayed at the bottom of the screen).
- Go To—Specify a T value and the cursor moves to the corresponding x and y coordinates.
- Defn—Display the functions responsible for the plot.

These tools are common Plot view operations.

Typically, you would modify a plot by changing its definition in Symbolic view. However, you can modify some plots by changing the Plot Setup parameters. For example, you can plot a triangle instead of a circle simply by changing two plot setup parameters. The definitions in Symbolic view remain unchanged. To do this, use the following procedure.

1. Press Shift Plot ↳Setup.
2. Change **T Step** to **120**.
3. Tap Page ⅓ ▼.
4. From the **Method** menu, select **Fixed-Step Segments**.

5. Press Plot ↳Setup.

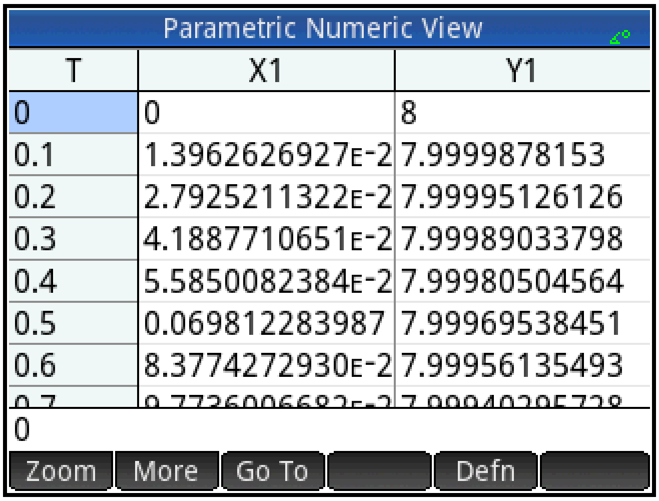


A triangle is displayed instead of a circle. This is because the new value of **T Step** makes the points being plotted 120° apart instead of the nearly continuous 5°. And by selecting **Fixed-Step Segments** the points 120° apart are connected with line segments.

Displaying the Numeric view

1. Press Num ↳Setup.

2. With the cursor in the **T** column, type a new value and tap OK.The table scrolls to the value you entered.



Parametric Numeric View

T	X1	Y1
0	0	8
0.1	1.3962626927E-2	7.9999878153
0.2	2.7925211322E-2	7.99995126126
0.3	4.1887710651E-2	7.99989033798
0.4	5.5850082384E-2	7.99980504564
0.5	0.069812283987	7.99969538451
0.6	8.3774272930E-2	7.99956135493
0.7	9.7736006682E-2	7.99940295728

0

You can also zoom in or out on the independent variable (thereby decreasing or increasing the increment between consecutive values). These are common Numeric view operations.

You can see the Plot and Numeric views side by side by combining the Plot and Numeric views.

17 Polar app

The Polar app enables you to explore polar equations. Polar equations are equations in which r—the distance a point is from the origin: (0,0)—is defined in terms of q, the angle a segment from the point to the origin makes with the polar axis. Such equations take the form r = f(θ).

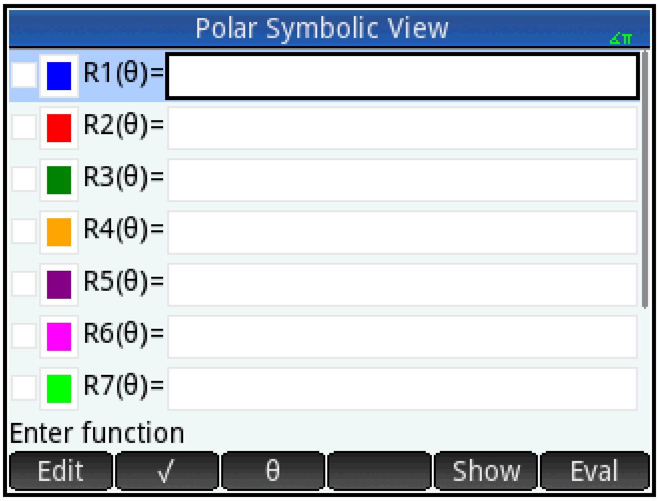
Getting started with the Polar app

The Polar app uses the six standard app views. That chapter also describes the menu buttons used in the Polar app.

Throughout this chapter, we will explore the expression $5\pi\cos(\theta/2)\cos(\theta)^2$.

Opening the Polar app

▲ Press Apps Info , and then select **Polar**.



The app opens in Symbolic view.

The graphical and numerical data you see in Plot view and Numeric view are derived from the symbolic functions defined here.

Defining the function

There are 10 fields for defining polar functions. These are labelled R1(θ) through R9(θ) and R0(θ).

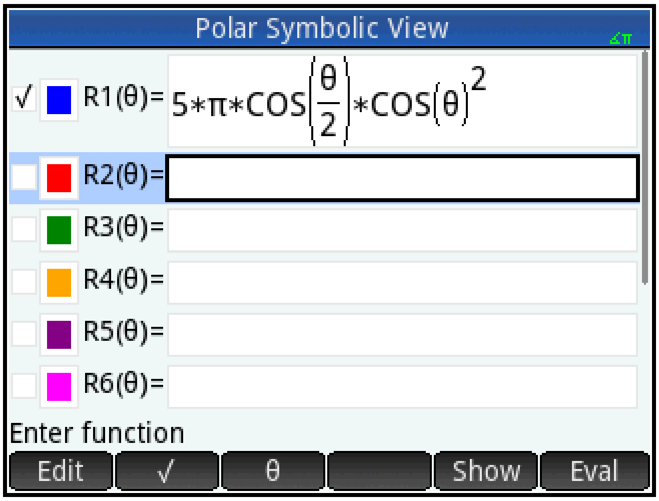
1. Highlight the field you want to use, either by tapping on it or scrolling to it. If you are entering a new function, just start typing. If you are editing an existing function, tap Edit and make your changes.

   When you have finished defining or changing the function, press Enter ≈ .

2. Define the expression 5πcos(θ/2)cos(θ)^2.

5 Shift 3 COS xtθn ÷ 2 ▶ ▶ COS xtθn ▶ x^2 Enter

The xtθn key enters whatever variable is relevant to the current app. In this app, it enters a θ.


Polar Symbolic View

☑ 🟦 R1(θ)= $5*\pi*\text{COS}\left(\frac{\theta}{2}\right)*\text{COS}(\theta)^2$

□ 🟥 R2(θ)=

□ 🟩 R3(θ)=

□ 🟧 R4(θ)=

□ 🟪 R5(θ)=

□ 🟪 R6(θ)=

Enter function

3. If you wish, choose a color for the plot other than its default. You do this by selecting the colored square to the left of the function set, tapping Choose, and selecting a color from the color-picker.

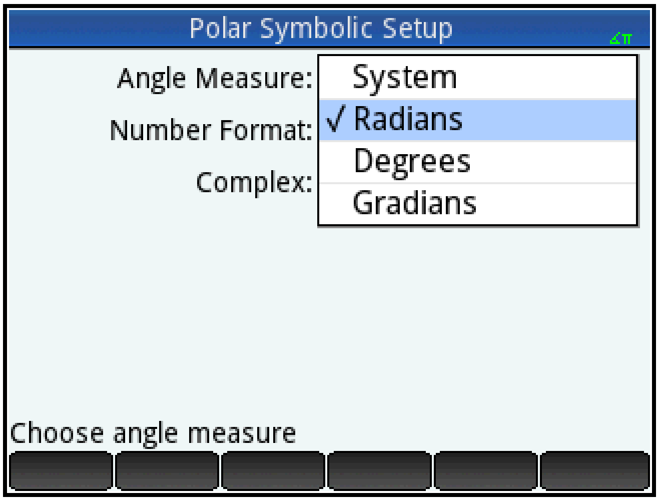
Adding definitions, modifying definitions, and evaluating dependent definitions are common operations in Symbolic view.

Setting the angle measure

To set the angle measure to radians:

1. Press Shift Symb (Setup) .

2. Select **Angle Measure**, and then select **Radians**.


Polar Symbolic Setup

Angle Measure: System / √ Radians / Degrees / Gradians

Number Format:

Complex:

Choose angle measure

These are common operations in Symbolic Setup view.

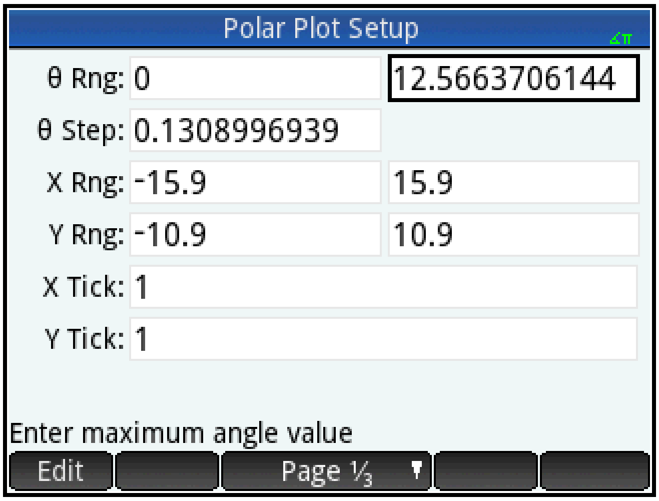
Setting up the plot

1. To open Plot Setup view, press Shift Plot ↳Setup.

2. Set up the plot by specifying appropriate graphing options. In this example, set the upper limit of the range of the independent variable to **4π**:

   Select the second **T Rng** field and enter:

   Select the second **θ Rng** field and enter 4 Shift 3 π # OK.



Polar Plot Setup

θ Rng: 0 12.5663706144

θ Step: 0.1308996939

X Rng: -15.9 15.9

Y Rng: -10.9 10.9

X Tick: 1

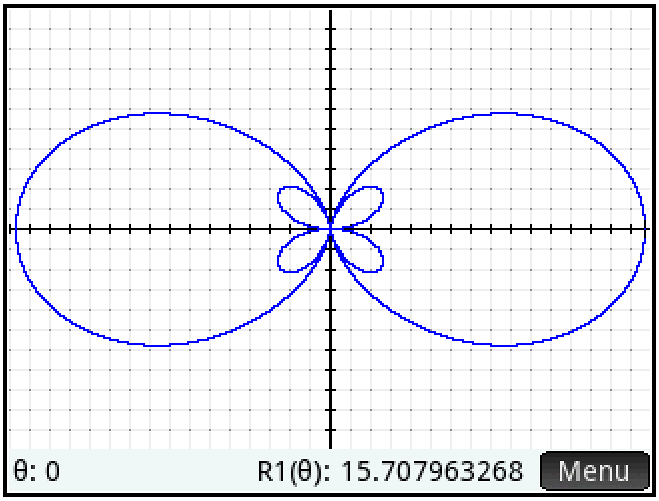
Y Tick: 1

Enter maximum angle value

There are numerous ways of configuring the appearance of Plot view, using the common operations in Plot view.

Plotting the expression

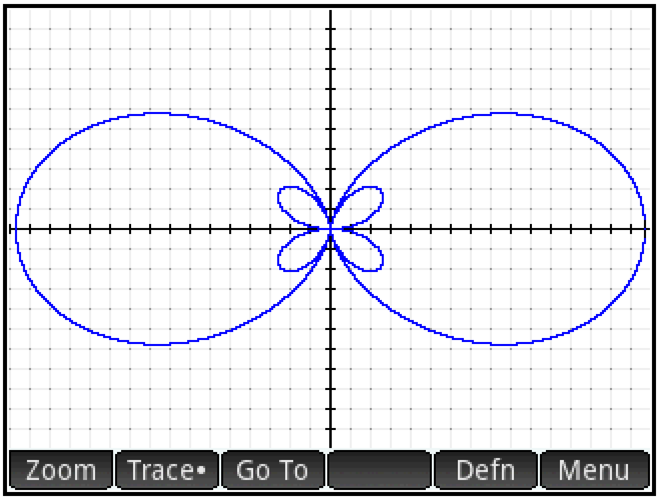
▲ Press Plot ↳Setup.



θ: 0 R1(θ): 15.707963268

Exploring the graph

- To display the Plot view menu, press Menu.



A number of options appear to help you explore the graph, such as zooming and tracing. You can also jump directly to a particular q value by entering that value. The Go To screen appears with the number you typed on the entry line. Just tap OK to accept it. (You could also tap the Go To and specify the target value.)

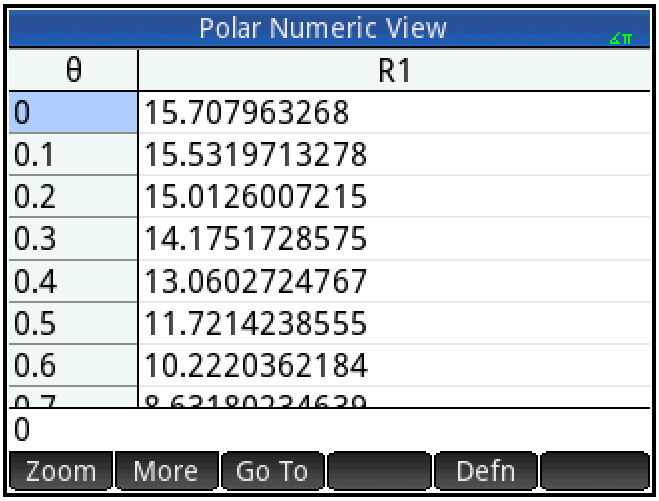
If only one polar equation is plotted, you can see the equation that generated the plot by tapping Defn. If there are several equations plotted, move the tracing cursor to the plot you are interested—by pressing ▲ or ▼—and then tap Defn.

Exploring plots is a common operation in Plot view.

Displaying the Numeric view

1. Press Num ↳Setup .

   The Numeric view displays a table of values for θ and R1. If you had specified, and selected, more than one polar function in Symbolic view, a column of evaluations would appear for each one: R2, R3, R4 and so on.



Polar Numeric View

θ	R1
0	15.707963268
0.1	15.5319713278
0.2	15.0126007215
0.3	14.1751728575
0.4	13.0602724767
0.5	11.7214238555
0.6	10.2220362184
0.7	8.63180234639

0

2. With the cursor in the **θ** column, type a new value and tap OK .The table scrolls to the value you entered.

You can also zoom in or out on the independent variable (thereby decreasing or increasing the increment between consecutive values). This and other options are common operations in Numeric view.

You can see the Plot and Numeric views side by side by combining the Plot and Numeric views.

18 Sequence app

The HP Prime Sequence app allows you to define sequences either explicitly or recursively. Recursive definitions can define U(N) in terms of only U(N − 1) or both U(N − 1) and U(N − 2). Similarly, a recursive definition can define U(N + 1) in terms of only U(N), or it can define U(N + 2) in terms of both U(N) and U(N + 1). Finally, N can start at 1 (the default value), 0, or any positive integer.

In Symbolic view, the first two boxes contain the first two numerical values in the sequence, if necessary. For an explicitly defined sequence, the values can be empty. For a recursively defined sequence, you must enter at least one value, depending on the nature of your definition.

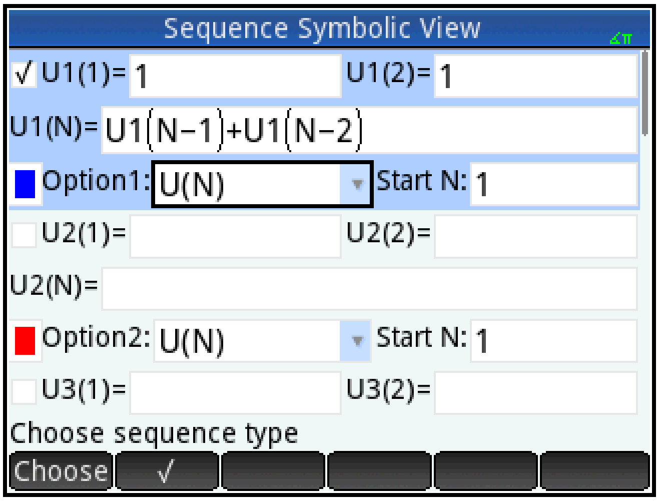
NOTE: The labels for the values change, depending on the starting value for N selected in the **Option** box.

In the third box, enter the symbolic definition.

In the Option box, select the terms for the symbolic definition. By default, U(N) is selected, which means that the symbolic definition is for U(N) in terms of N, U(N-1), both U(N-1) and U(N-2), or some combination of the previous three options. The other option is U(N+k), which means that the symbolic definition is for either U(N +1) in terms of U(N) or U(N+2) in terms of U(N+1) and U(N).

Next to the Option box, another box allows you to enter the starting value for N. This value can be 0 or any positive integer.

In the following example, the Fibonacci sequence is defined as U1(1) = 1, U1(2) = 1, and U1(N) = U1(N − 1) + U1(N − 2). The Option value is the default U(N), and the starting value of N is 1. This example is used in Getting started with the Sequence app on page 315.



Sequence Symbolic View

✓ U1(1)= 1 U1(2)= 1

U1(N)= U1(N−1)+U1(N−2)

🟦 Option1: U(N) Start N: 1

□ U2(1)= U2(2)=

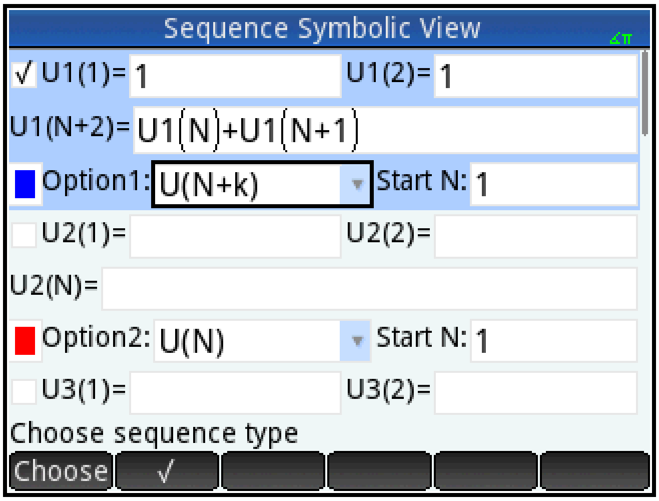
U2(N)=

🟥 Option2: U(N) Start N: 1

□ U3(1)= U3(2)=

Choose sequence type

In the following example, the Fibonacci sequence is defined as U1(1) = 1, U1(2) = 1, and U1(N +2) = U1(N) + U1(N + 1). The Option value U(N+k) is select, and the starting value of N is 1.


Sequence Symbolic View

✓ U1(1)= 1 U1(2)= 1

U1(N+2)= U1(N)+U1(N+1)

🟦 Option1: U(N+k) Start N: 1

□ U2(1)= U2(2)=

U2(N)=

🟥 Option2: U(N) Start N: 1

□ U3(1)= U3(2)=

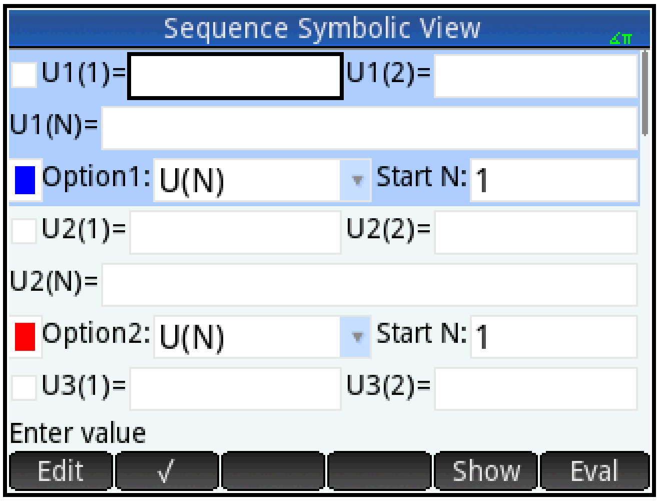
Choose sequence type

Getting started with the Sequence app

The following example explores the well-known Fibonacci sequence, where each term, from the third term on, is the sum of the preceding two terms. In this example, we specify three sequence fields: the first term, the second term and a rule for generating all subsequent terms.

Opening the Sequence app

▲ Press Apps Info , and then select **Sequence**.


Sequence Symbolic View

□ U1(1)= U1(2)=

U1(N)=

🟦 Option1: U(N) Start N: 1

□ U2(1)= U2(2)=

U2(N)=

🟥 Option2: U(N) Start N: 1

□ U3(1)= U3(2)=

Enter value

The app opens in Symbolic view.

Defining the expression

To define the following Fibonacci sequence:

$U_1 = 1, U_2 = 1, U_n = U_{n-1} + U_{n-2}$ for n > 2

1. In the **U1(1)** field, specify the first term of the sequence and the starting value of N:

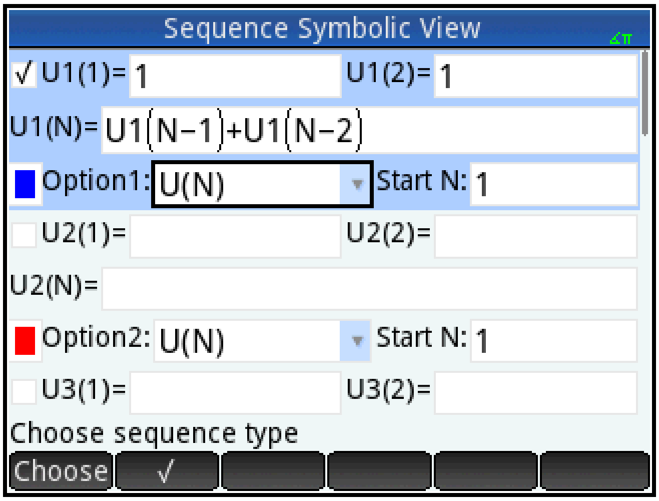
   1 Enter ≈

2. In the **U1(2)** field, specify the second term of the sequence:

   1 Enter ≈

3. In the **U1(N)** field, specify the formula for finding the nth term of the sequence from the previous two terms (using the buttons at the bottom of the screen to help with some entries):

   U1 (N−1) + Ans ; U1 (N−2) Enter ≈



Sequence Symbolic View

√ U1(1)= 1 U1(2)= 1

U1(N)= U1(N−1)+U1(N−2)

Option1: U(N) Start N: 1

U2(1)= U2(2)=

U2(N)=

Option2: U(N) Start N: 1

U3(1)= U3(2)=

Choose sequence type

4. Optionally, select a color for your graph.

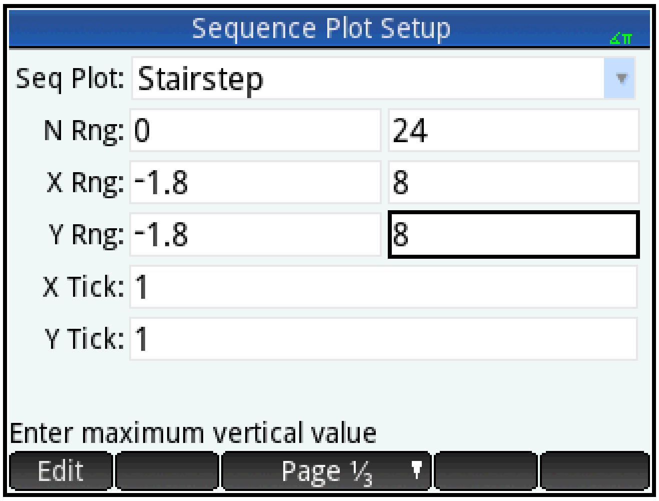
Setting up the plot

1. To open Plot Setup view, press Shift Plot Setup.

2. To reset all settings to their default values, press Shift Esc Clear .

3. Select **Stairstep** from the Seq Plot menu.

4. Set the **X Rng** maximum, and the **Y Rng** maximum, to **8** (as shown in the following figure).


Sequence Plot Setup

Seq Plot: Stairstep

N Rng: 0 24

X Rng: -1.8 8

Y Rng: -1.8 8

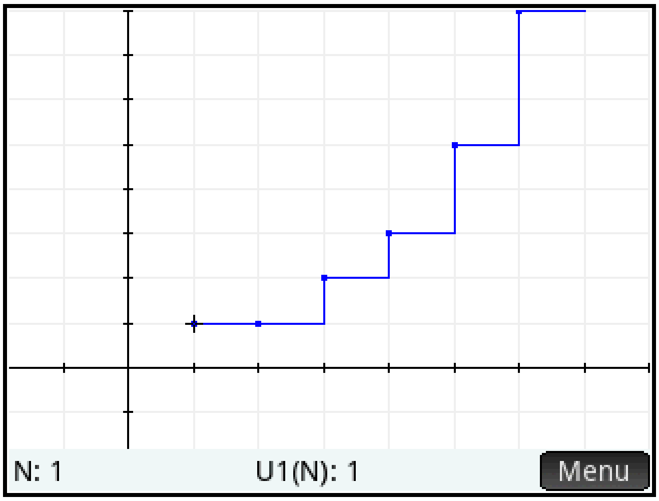
X Tick: 1

Y Tick: 1

Enter maximum vertical value

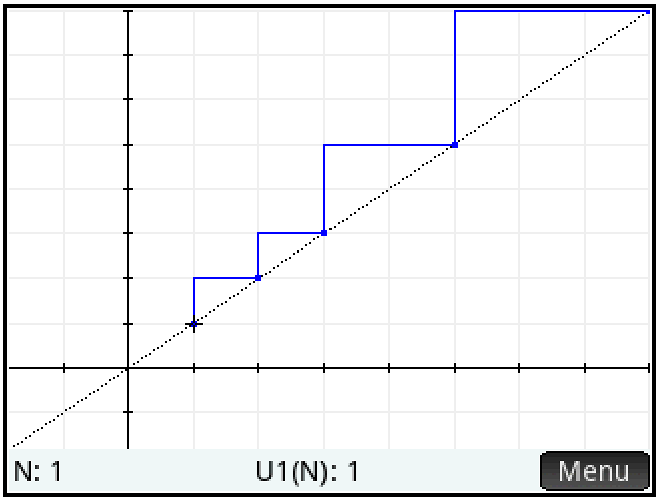
Plotting the sequence

1. Press Plot ↳Setup.


N: 1 U1(N): 1

2. To plot the sequence using the cobweb option, return to Plot Setup view (Shift Plot ↳Setup) and select **Cobweb** from the **Seq Plot** menu.

3. Press Plot ↳Setup.


N: 1 U1(N): 1

Exploring the graph

The Menu button gives you access to common plot-exploration tools, such as:

- Zoom—Zoom in or out on the plot
- Trace•—Trace along a graph
- Go To—Go to a specified n value
- Defn—Display the sequence definition

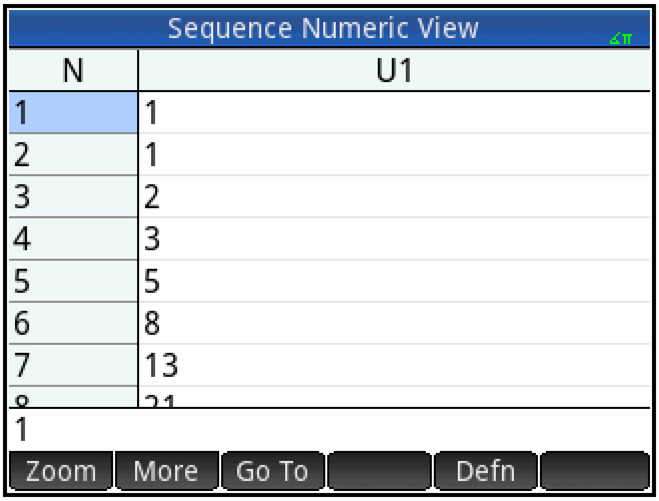
These tools are common operations in Plot view.

Split screen and autoscaling options are also available by pressing View Copy .

Displaying Numeric view

1. Display Numeric view:

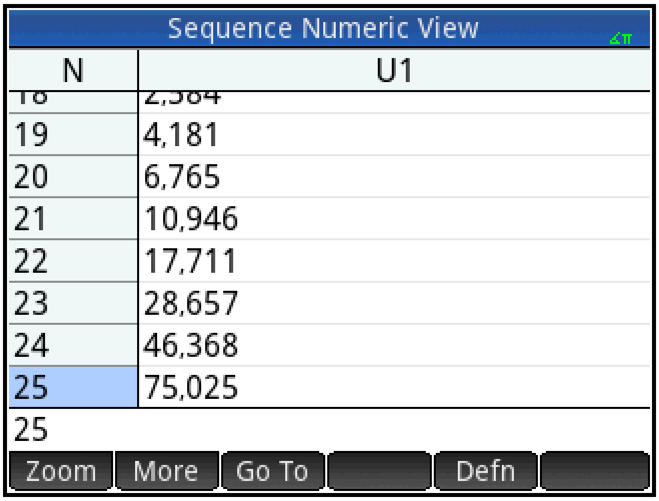
   Num ↳Setup


Sequence Numeric View

N	U1
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21

1

2. With the cursor anywhere in the **N** column, type a new value and tap OK.


Sequence Numeric View

N	U1
18	2,584
19	4,181
20	6,765
21	10,946
22	17,711
23	28,657
24	46,368
25	75,025

25

The table of values scrolls to the value you entered. You can then see the corresponding value in the sequence. The previous figure shows that the 25th value in the Fibonacci sequence is 75,025.

Exploring the table of values

The Numeric view gives you access to common table-exploration tools, such as the following:

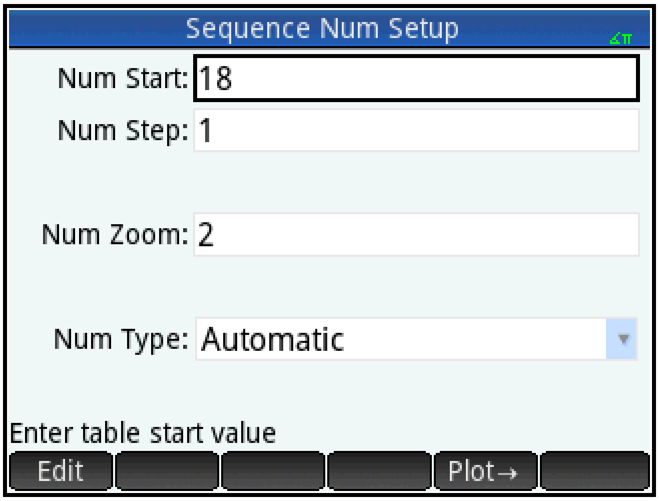
- Zoom—Change the increment between consecutive values
- Defn—Display the sequence definition
- Column—Choose the number of sequences to display

These tools are common operations in Numeric view.

Split screen and autoscaling options are also available by pressing View Copy .

Setting up the table of values

The Numeric Setup view provides options common to most of the graphing apps, although there is no zoom factor as the domain for the sequences is the set of counting numbers. These are common operations in Numeric Setup view.



Sequence Num Setup

Num Start: 18

Num Step: 1

Num Zoom: 2

Num Type: Automatic

Enter table start value

Another example: Explicitly defined sequences

In the following example, we define the nth term of a sequence simply in terms of n itself. In this case, there is no need to enter either of the first two terms numerically.

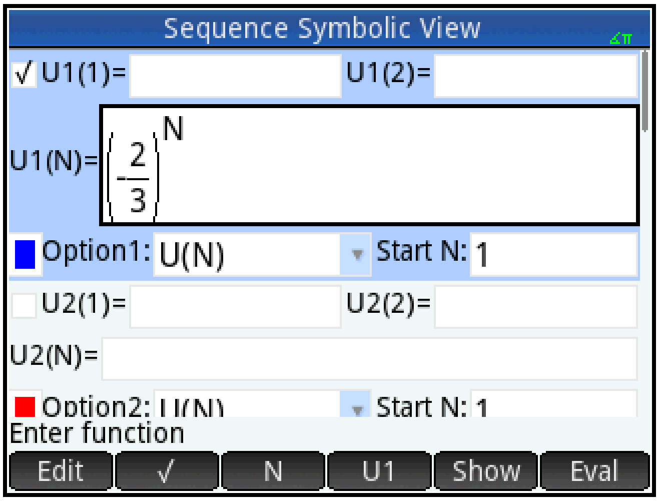
Defining the expression

▲ Define U1(N) = (–2/3)^{N}.

Select U1N:

Enter ( ) +/– □,√□,|□| , and then select $\frac{\square}{\square}$.

Enter 2 ▼ 3 ▶ ▶ x^y N Enter .



Sequence Symbolic View

√ U1(1)= U1(2)=

U1(N)= $\left(-\frac{2}{3}\right)^{N}$

Option1: U(N) Start N: 1

U2(1)= U2(2)=

U2(N)=

Option2: U(N) Start N: 1

Enter function

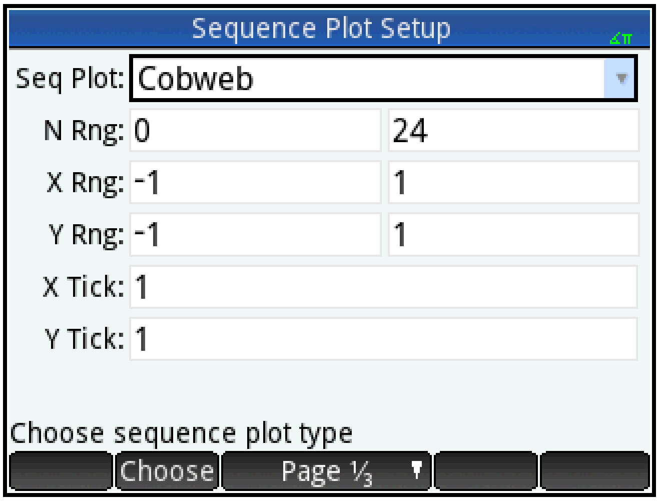
Setting up the plot

1. To open Plot Setup view, press Shift Plot ↳Setup.

2. To reset all setting to their default values, press Shift Esc Clear .

3. Tap **Seq Plot** and select **Cobweb**.

4. Set both **X Rng** and **Y Rng** to **[–1, 1]** as shown in the following figure.



Sequence Plot Setup

Seq Plot: Cobweb

N Rng: 0 24

X Rng: -1 1

Y Rng: -1 1

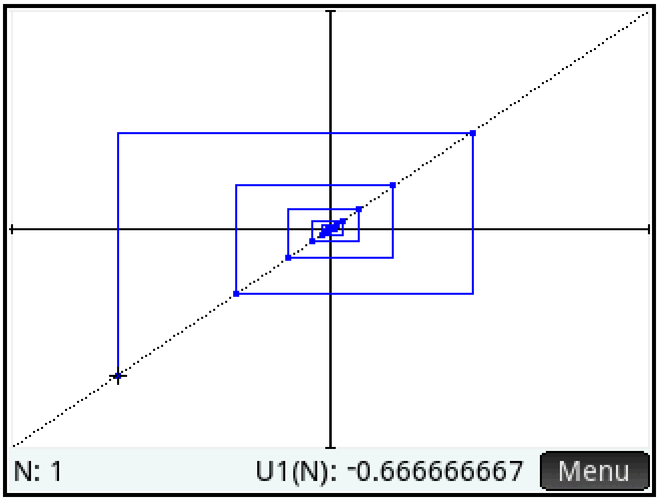
X Tick: 1

Y Tick: 1

Choose sequence plot type

Plotting the sequence

▲ Press Plot ↳Setup.



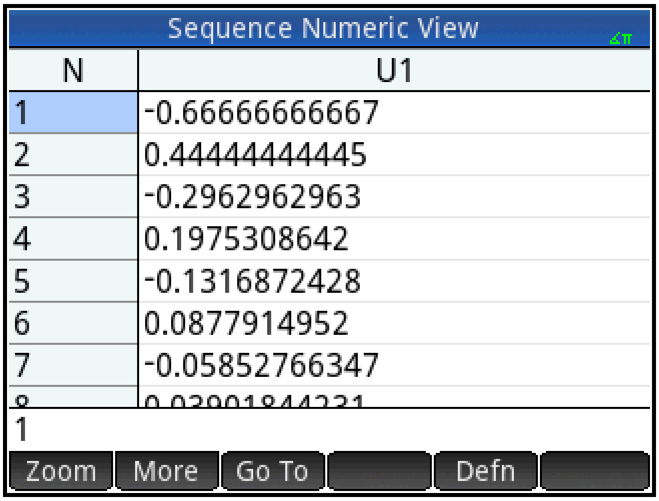
N: 1 U1(N): -0.666666667

Press Enter ≈ to see the dotted lines in the previous figure. Press it again to hide the dotted lines.

Exploring the table of values

1. Press Num Setup .

2. Tap Column and select **1** to see the sequence values.



Sequence Numeric View

N	U1
1	-0.66666666667
2	0.44444444445
3	-0.2962962963
4	0.1975308642
5	-0.1316872428
6	0.0877914952
7	-0.05852766347
8	0.03901844231

1

19 Finance app

The Finance app enables you to solve time-value-of-money (TVM) and amortization problems. You can use the app to do compound interest calculations and to create amortization tables.

Compound interest is accumulative interest, that is, interest on interest already earned. The interest earned on a given principal is added to the principal at specified compounding periods, and then the combined amount earns interest at a certain rate. Financial calculations involving compound interest include savings accounts, mortgages, pension funds, leases, and annuities.

Getting started with the Finance app

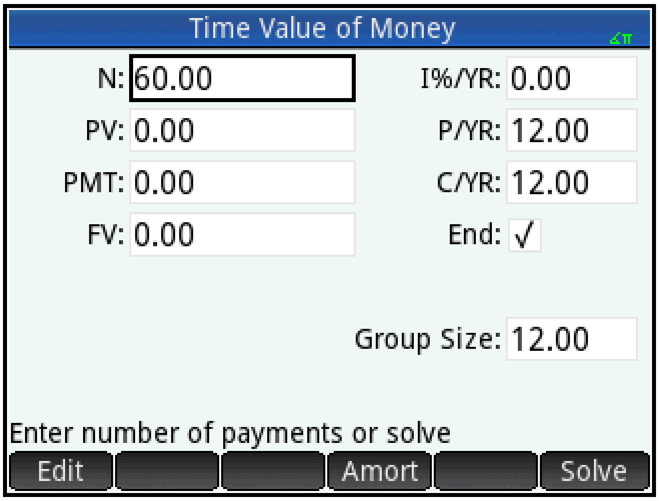
Suppose you finance the purchase of a car with a 5-year loan at 5.5% annual interest, compounded monthly. The purchase price of the car is $19,500, and the down payment is $3,000. First, what are the required monthly payments? Second, what is the largest loan you can afford if your maximum monthly payment is $300? Assume that the payments start at the end of the first period.

1. To open the Finance app, press Apps Info and then select **Finance**.

   The app opens in the Numeric view.

2. In the **N** field, enter 5 × 12 and press Enter.

   Notice that the result of the calculation (60) appears in the field. This is the number of months over a five-year period.

   Time Value of Money

   N: 60.00
   I%/YR: 0.00
   PV: 0.00
   P/YR: 12.00
   PMT: 0.00
   C/YR: 12.00
   FV: 0.00
   End: ✓

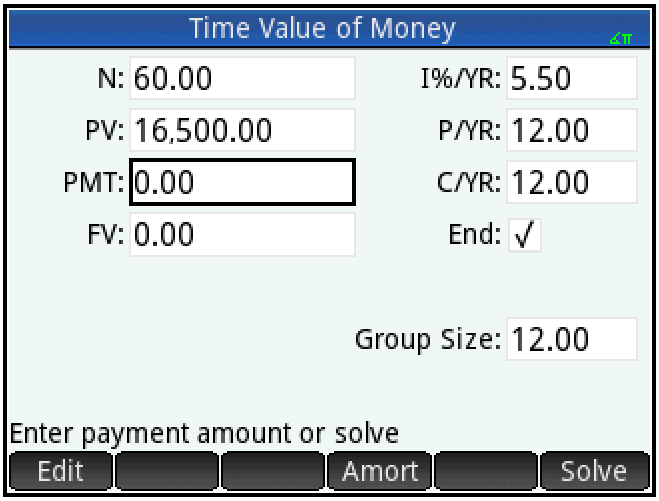
   Group Size: 12.00

   Enter number of payments or solve

3. In the **I%/YR** field, enter 5.5—the interest rate—and press Enter.

4. In **PV** field, enter 19500 − 3000 and press Enter. This is the present value of the loan, being the purchase price less the deposit.

5. Leave **P/YR** and **C/YR** both at 12 (their default values). Leave **End** as the payment option. Also, leave future value, **FV**, as **0** (as your goal is to end up with a future value of the loan of 0).


Time Value of Money

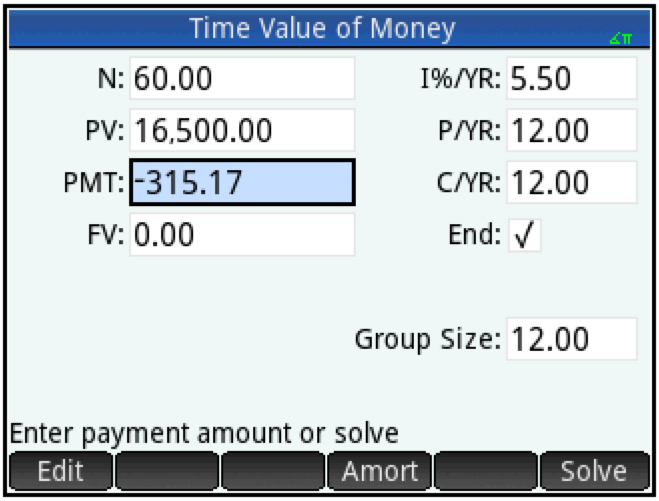
N: 60.00
I%/YR: 5.50
PV: 16,500.00
P/YR: 12.00
PMT: 0.00
C/YR: 12.00
FV: 0.00
End: ✓
Group Size: 12.00

Enter payment amount or solve

6. Move the cursor to the **PMT** field and tap Solve. The PMT value is calculated as –315.17. In other words, your monthly payment will be $315.17.

The PMT value is negative to indicate that it is money owed by you.

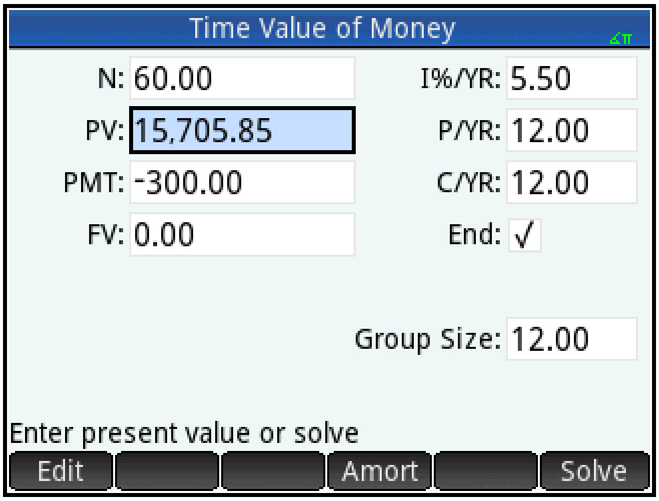
Note that the PMT value is greater than 300, that is, greater than the amount you can afford to pay each month. So you ned to re-run the calculations, this time setting the PMT value to –300 and calculating a new PV value.


Time Value of Money

N: 60.00
I%/YR: 5.50
PV: 16,500.00
P/YR: 12.00
PMT: -315.17
C/YR: 12.00
FV: 0.00
End: ✓
Group Size: 12.00

Enter payment amount or solve

7. In the PMT field, enter +/- 300, point the cursor to the **PV** field, and tap Solve.



Time Value of Money

N: 60.00

PV: 15,705.85

PMT: -300.00

FV: 0.00

I%/YR: 5.50

P/YR: 12.00

C/YR: 12.00

End: ✓

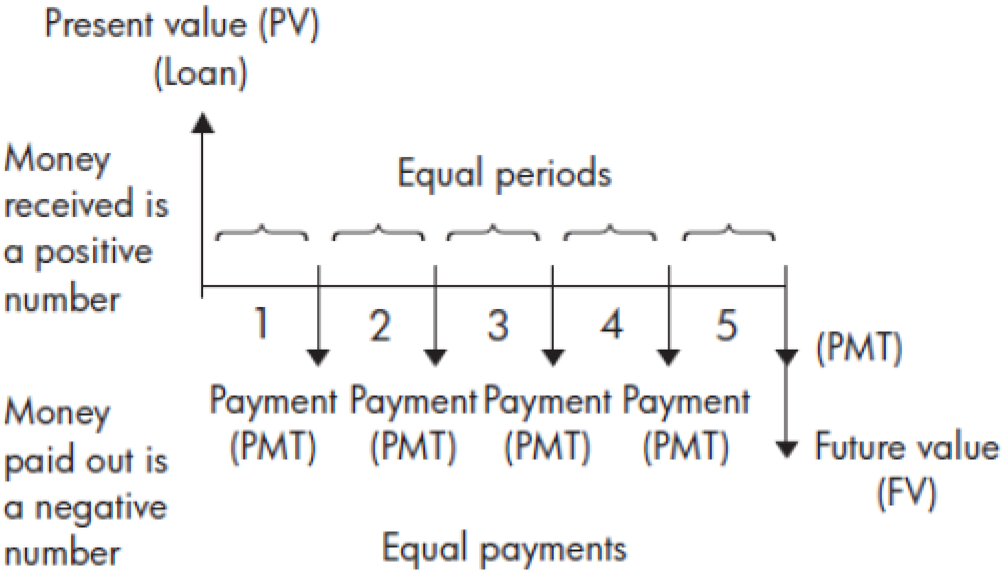
Group Size: 12.00

Enter present value or solve

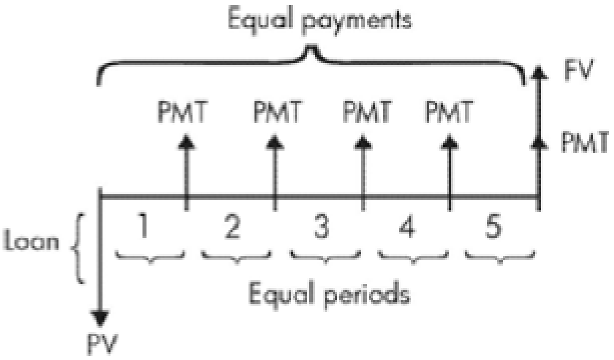
The PV value is calculated as 15,705.85, this being the maximum you can borrow. Thus, with your $3,000 deposit, you can afford a car with a price tag of up to $18,705.85.

Cash flow diagrams

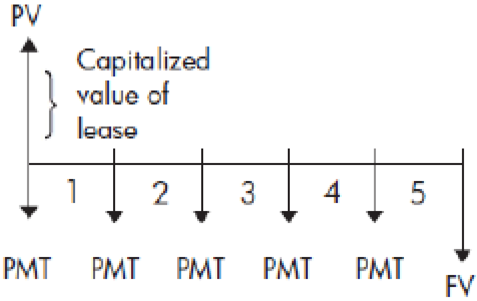
TVM transactions can be represented in cash flow diagrams. A cash flow diagram is a time line divided into equal segments representing the compounding periods. Arrows represent the cash flows. These could be positive (upward arrows) or negative (downward arrows), depending on the point of view of the lender or borrower. The following cash flow diagram shows a loan from a borrower's point of view.



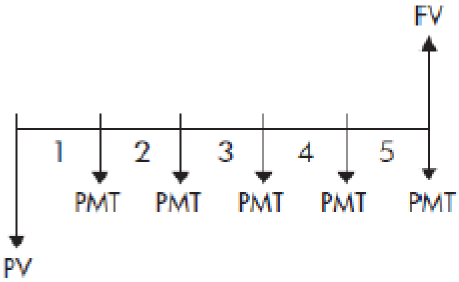
The following cash flow diagram shows a loan from the lender's point of view.



Cash flow diagrams also specify when payments occur relative to the compounding periods. The following diagram shows lease payments at the beginning of the period.



The following diagram shows deposits (PMT) into an account at the end of each period.



Time value of money (TVM)

Time-value-of-money (TVM) calculations make use of the notion that a dollar today will be worth more than a dollar sometime in the future. A dollar today can be invested at a certain interest rate and generate a return that the same dollar in the future cannot. This TVM principle underlies the notion of interest rates, compound interest, and rates of return.

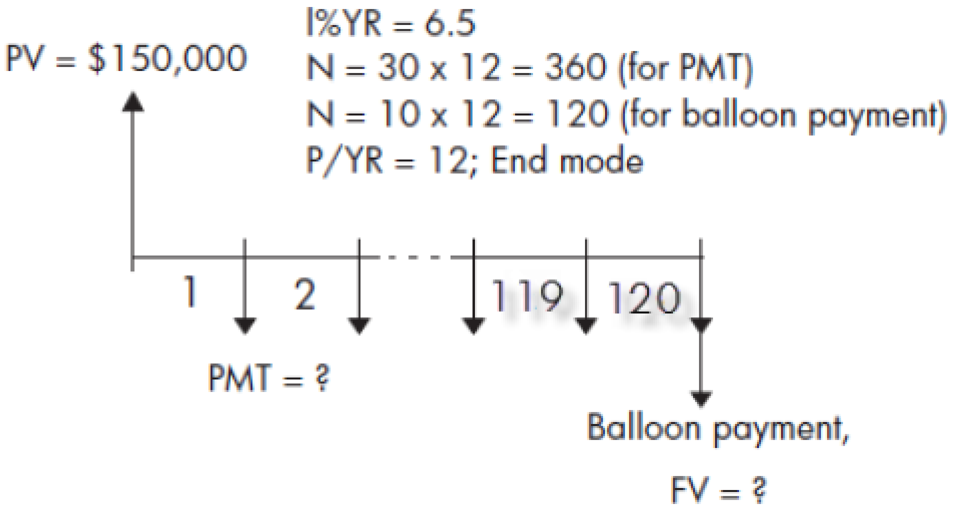
There are seven TVM variables as follows:

Variable	Description
N	The total number of compounding periods or payments.
1%/YR	The nominal annual interest rate (or investment rate). This rate is divided by the number of payments per year (P/YR) to compute the nominal interest rate per compounding period. This is the interest rate actually used in TVM calculations.
PV	The present value of the initial cash flow. To a lender or borrower, PV is the amount of the loan; to an investor, PV is the initial investment. PV always occurs at the beginning of the first period.
P/YR	The number of payments made in a year.
PMT	The periodic payment amount. The payments are the same amount each period and the TVM calculation assumes that no payments are skipped. Payments can occur at the beginning or the end of each compounding period—an option you control by selecting or clearing the **End** option.
C/YR	The number of compounding periods in a year.
FV	The future value of the transaction: the amount of the final cash flow or the compounded value of the series of previous cash flows. For a loan, this is the size of the final balloon payment (beyond any regular payment due). For an investment, this is its value at the end of the investment period.

Another example: TVM calculations

Suppose you have taken out a 30-year, $150,000 house mortgage at 6.5% annual interest. You expect to sell the house in 10 years, repaying the loan in a balloon payment. Find the size of the balloon payment—that is, the value of the mortgage after 10 years of payment.

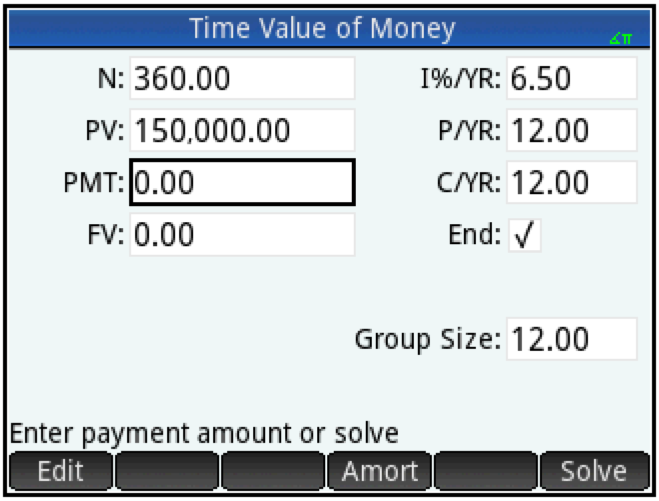
The following cash flow diagram illustrates the case of a mortgage with balloon payment.



1. To open the Finance app, press Apps Info and select **Finance**.

2. To return all fields to their default values, press Shift Esc Clear .

3. Enter the known TVM variables, as shown in the following figure.


Time Value of Money

N: 360.00
I%/YR: 6.50
PV: 150,000.00
P/YR: 12.00
PMT: 0.00
C/YR: 12.00
FV: 0.00
End: √
Group Size: 12.00

Enter payment amount or solve

4. Select **PMT** and tap Solve. The PMT field shows –984.10. In other words, the monthly payments are $948.10.

5. To determine the balloon payment or future value (FV) for the mortgage after 10 years, enter `120` for **N**, select **FV**, and tap Solve.

The FV field shows –127,164.19, indicating that the future value of the loan (that is, how much is still owing) as $127,164.19.

Amortizations

Amortization calculations determine the amounts applied towards the principal and interest in a payment, or series of payments. They also use TVM variables.

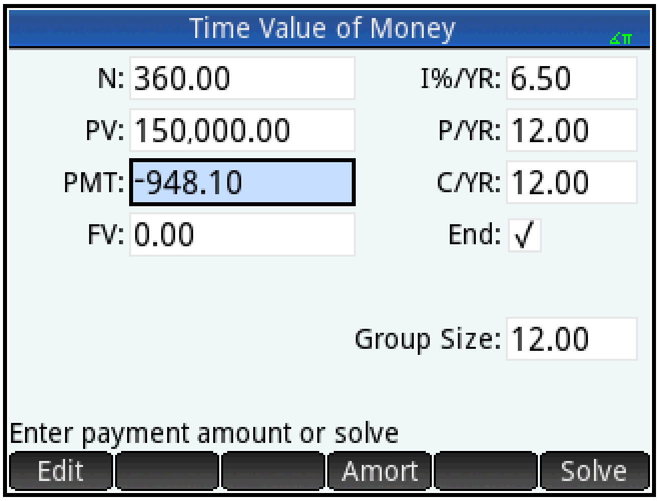
Calculating amortizations

1. To open the Finance app, press Apps Info and select **Finance**.
2. Specify the number of payments per year (**P/YR**).
3. Specify whether payments are made at the beginning or end of periods.
4. Enter values for **I%YR**, **PV**, **PMT**, and **FV**.
5. Enter the number of payments per amortization period in the **Group Size** box. By default, the group size is **12** to reflect annual amortization.
6. Tap Amort. The calculator displays an amortization table. For each amortization period, the table shows the amounts applied to interest and principal, as well as the remaining balance of the loan.

Amortization for a home mortgage example

Using the data from the previous example of a home mortgage with balloon payment (see Another example: TVM calculations on page 328), calculate how much has been applied to the principal, how much has been paid in interest, and the balance remaining after the first 10 years (that is, after 12 × 10 = 120 payments).

1. Make sure that your data matches that shown in the following figure.



Time Value of Money

N: 360.00

PV: 150,000.00

PMT: -948.10

FV: 0.00

I%/YR: 6.50

P/YR: 12.00

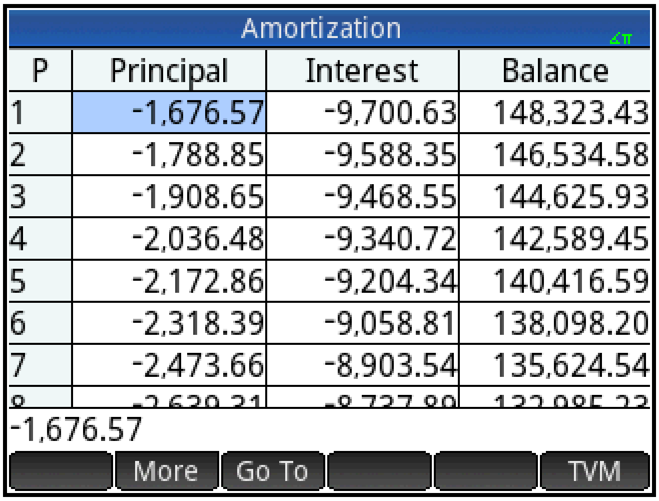
C/YR: 12.00

End: ✓

Group Size: 12.00

Enter payment amount or solve

2. Tap Amort.

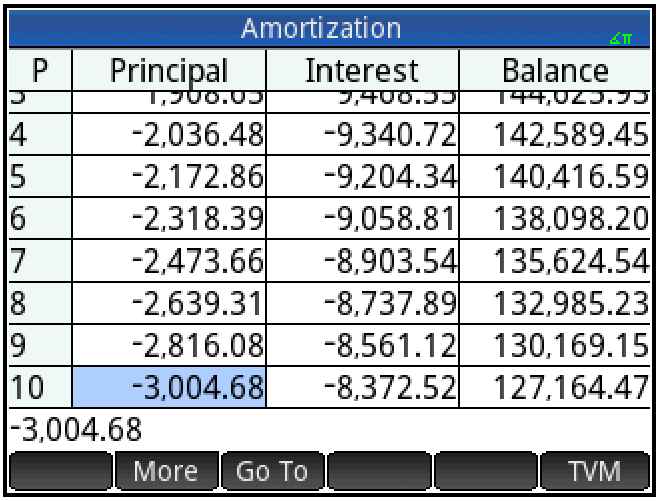


Amortization

P	Principal	Interest	Balance
1	-1,676.57	-9,700.63	148,323.43
2	-1,788.85	-9,588.35	146,534.58
3	-1,908.65	-9,468.55	144,625.93
4	-2,036.48	-9,340.72	142,589.45
5	-2,172.86	-9,204.34	140,416.59
6	-2,318.39	-9,058.81	138,098.20
7	-2,473.66	-8,903.54	135,624.54
8	-2,639.31	-8,737.89	132,985.23

-1,676.57

3. Scroll down the table to payment group 10. Note that after 10 years, $22,835.53 has been paid off the principal and $90,936.47 paid in interest, leaving a balloon payment due of $127,164.47.



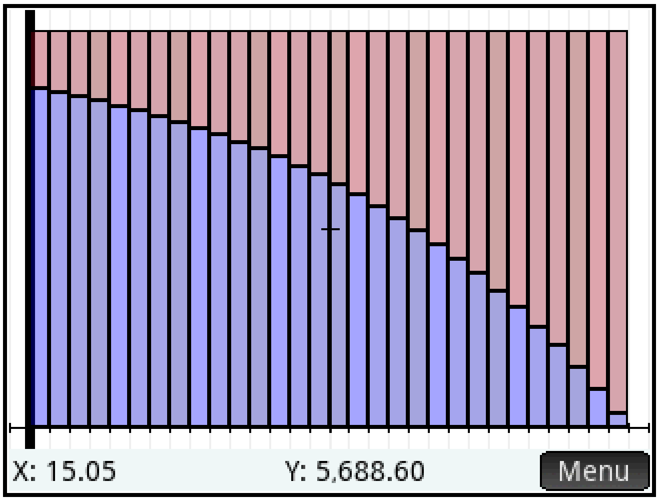
Amortization

P	Principal	Interest	Balance
3	-1,908.65	-9,468.55	144,625.93
4	-2,036.48	-9,340.72	142,589.45
5	-2,172.86	-9,204.34	140,416.59
6	-2,318.39	-9,058.81	138,098.20
7	-2,473.66	-8,903.54	135,624.54
8	-2,639.31	-8,737.89	132,985.23
9	-2,816.08	-8,561.12	130,169.15
10	-3,004.68	-8,372.52	127,164.47

-3,004.68

Amortization graph

▲ Press Plot to see the amortization schedule presented graphically.


X: 15.05 Y: 5,688.60

The balance owing at the end of each payment group is indicated by the height of a bar. The amount by which the principal has been reduced, and interest paid, during a payment group is shown at the bottom of the bottom of the screen. The previous example shows the first payment group selected. This represents the first group of 12 payments (or the state of the loan at the end of the first year). By the end of that year, the principal had been reduced by $1,676.57 and $9,700.63 had been paid in interest.

Tap ◀ or ▶ to see the amount by which the principal has been reduced, and interest paid, during other payment groups.

20 Triangle Solver app

The Triangle Solver app enables you to calculate the length of a side of a triangle, or the size of an angle in a triangle, from information you supply about the other lengths, angles, or both.

You need to specify at least three of the six possible values—the lengths of the three sides and the size of the three angles—before the app can calculate the other values. Moreover, at least one value you specify must be a length. For example, you could specify the lengths of two sides and one of the angles; or you could specify two angles and one length; or all three lengths. In each case, the app will calculate the remaining values.

The HP Prime calculator alerts you if no solution can be found, or if you have provided insufficient data.

If you are determining the lengths and angles of a right-angled triangle, a simpler input form is available by tapping ⊿.

Getting started with the Triangle Solver app

The following example calculates the unknown length of the side of a triangle whose two known sides—of lengths 4 and 6—meet at an angle of 30 degrees.

Opening the Triangle Solver app

1. Press Apps Info, and then select **Triangle Solver**.

   The app opens in Numeric view.

   Radians

2. If there is unwanted data from a previous calculation, you can clear it all by pressing Shift Esc Clear.

Setting the angle measure

Make sure that your angle measure mode is appropriate. By default, the app starts in degree mode. If the angle information you have is in radians and your current angle measure mode is degrees, change the mode to degrees before running the solver. Tap Degree or Radians depending on the mode you want. (The button is a toggle button.)

NOTE: The lengths of the sides are labeled **a**, **b**, and **c**, and the angles are labeled **A**, **B**, and, **C**. It is important that you enter the known values in the appropriate fields. In our example, the length of two sides and the angle at which those sides meet is known. Therefore, if you specify the lengths of sides **a** and **b**, you must enter the angle as **C** (since C is the angle where A and B meet). If instead you entered the lengths as **b** and **c**, you would need to specify the angle as **A**. The calculator display helps you determine where to enter the known values.

Specify the known values

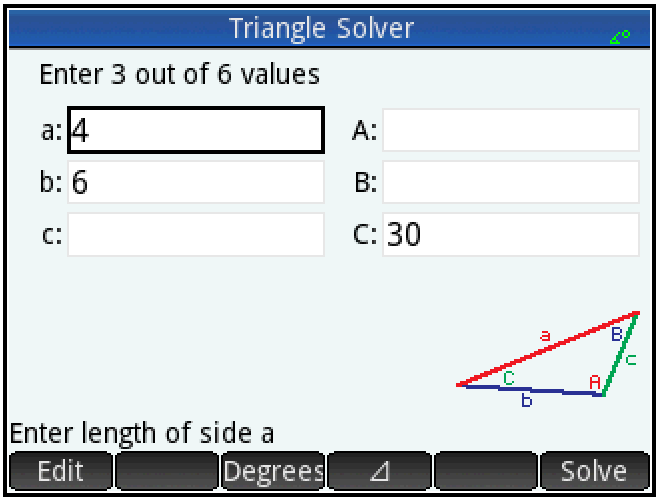
▲ Go to a field whose value you know, enter the value and either tap OK or press Enter.

Repeat for each known value.

a. In the **a** box, enter `4` and then press Enter.

b. In the **b** box, enter `6` and then press Enter.

c. In the **C** box, enter `30` and then press Enter.



Triangle Solver

Enter 3 out of 6 values

a: 4 A:

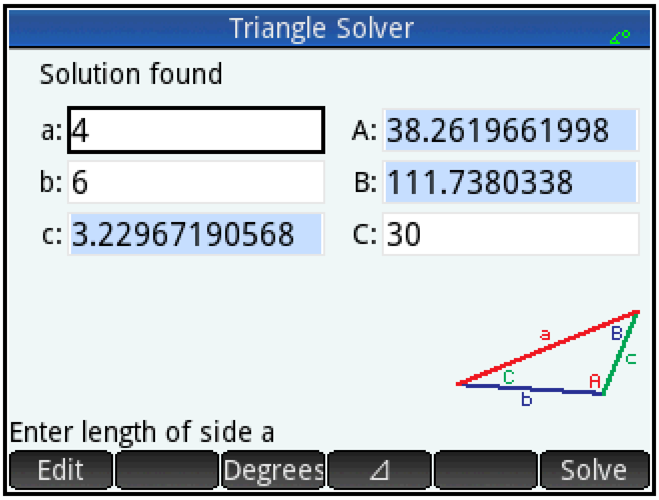
b: 6 B:

c: C: 30

Enter length of side a

Solving for the unknown values

▲ Tap Solve.


Triangle Solver

Solution found

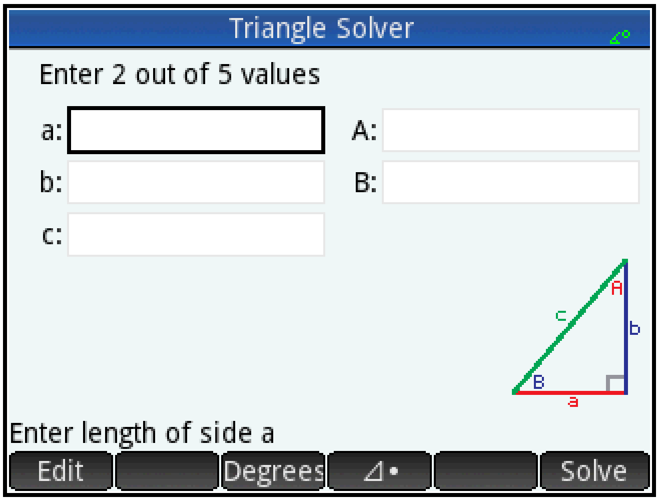
a: 4
A: 38.2619661998
b: 6
B: 111.7380338
c: 3.22967190568
C: 30

Enter length of side a

The app displays the values of the unknown variables. As the previous figure shows, the length of the unknown side in our example is 3.22967... The other two angles have also been calculated.

Choosing triangle types

The Triangle Solver app has two input forms: a general input form and a simpler, specialized form for right-angled triangles. If the general input form is displayed, and you are investigating a right-angled triangle, tap ⊿ to display the simpler input form. To return to the general input form, tap ⊿•. If the triangle you are investigating is not a right-angled triangle, or you are not sure what type it is, you should use the general input form.


Triangle Solver

Enter 2 out of 5 values

a:
A:
b:
B:
c:

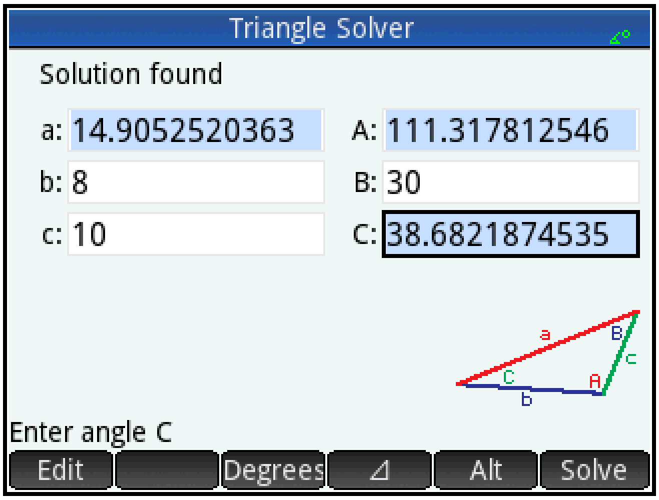
Enter length of side a

Special cases

Indeterminate case

If two sides and an adjacent acute angle are entered and there are two solutions, only one will be displayed initially.

In this case, the Alt button is displayed (as in the following figure). You can tap Alt to display the second solution and tap Alt again to return to the first solution.



Triangle Solver

Solution found

a: 14.9052520363 A: 111.317812546

b: 8 B: 30

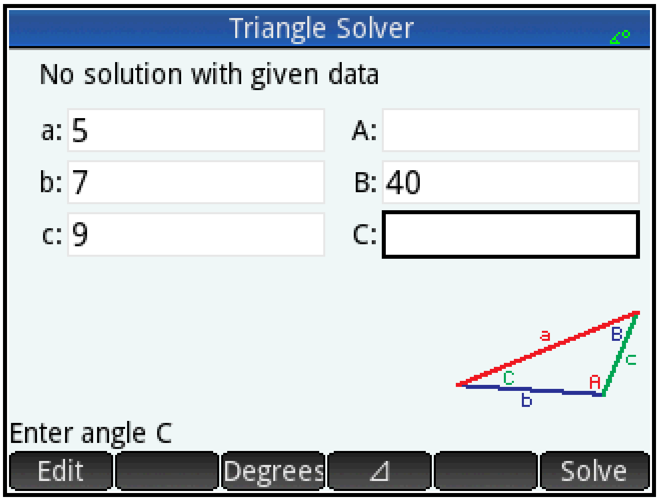
c: 10 C: 38.6821874535

Enter angle C

No solution with given data

If you are using the general input form and you enter more than 3 values, the values might not be consistent, that is, no triangle could possibly have all the values you specified. In these cases, **No sol with given data** appears on the screen.

The situation is similar if you are using the simpler input form (for a right-angled triangle) and you enter more than two values.



Triangle Solver

No solution with given data

a: 5 A:

b: 7 B: 40

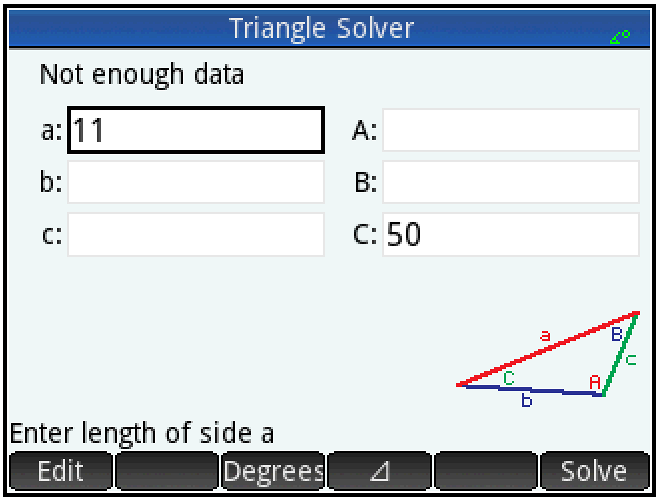
c: 9 C:

Enter angle C

Not enough data

If you are using the general input form, you need to specify at least three values for the Triangle Solver to be able to calculate the remaining attributes of the triangle. If you specify less than three, **Not enough data** appears on the screen.

If you are using the simplified input form (for a right-angled triangle), you must specify at least two values.



Triangle Solver

Not enough data

a: 11

A:

b:

B:

c:

C: 50

Enter length of side a

21 The Explorer apps

There are three explorer apps. These are designed for you to explore the relationships between the parameters of a function and the shape of the graph of that function. The explorer apps are:

- Linear Explorer

  For exploring linear functions
- Quadratic Explorer

  For exploring quadratic functions
- Trig Explorer

  For exploring sinusoidal functions

There are two modes of exploration: graph mode and equation mode. In graph mode you manipulate a graph and note the corresponding changes in its equation. In equation mode you manipulate an equation and note the corresponding changes in its graphical representation. Each explorer app has a number of equations and graphs for to explore, and app has a test mode. In test mode, you test you skills at matching equations to graphs.

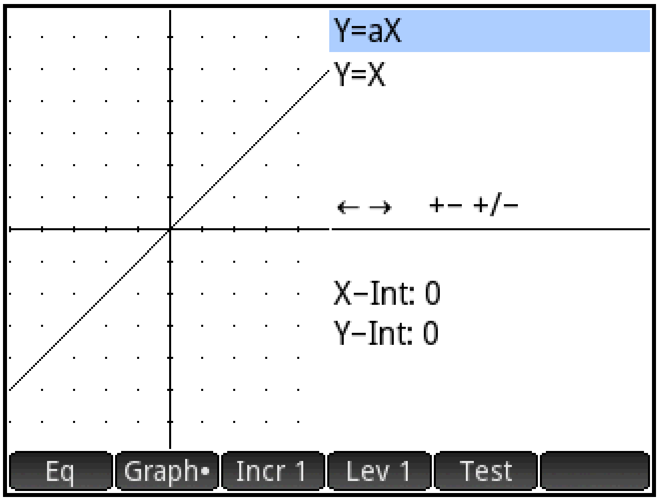
Linear Explorer app

The Linear Explorer app can be used to explore the behavior of the graphs of and as the values of a and b change.

Open the app

Press Apps Info and select **Linear Explorer**.

The left half of the display shows the graph of a linear function. The right half shows the general form of the equation being explored at the top and, below it, the current equation of that form. The keys you can use to manipulate the graph or equation appear below the equation. The x- and y-intercepts are given at the bottom.



Y=aX

Y=X

X-Int: 0

Y-Int: 0

There are two types (or levels) of linear equation available for you to explore: y = ax and y = ax + b. You choose between them by tapping Lev 1 or Lev 2.

The keys available to you to manipulate the graph or equation depend on the level you have chosen. For example, the screen for a level 1 equation shows this:

←→ +− +/−

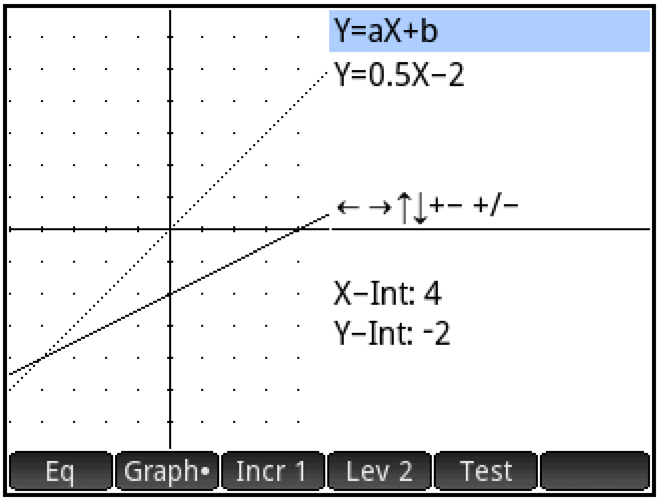
This means that you can press ◀, ▶, +, −, and +/−. If you choose a level 2 equation, the screen shows this:

←→ ↑↓+− +/−

This means that you can press ◀, ▶, ▲, ▼, +, −, and +/−.

Graph mode

The app opens in graph mode (indicated by the dot on the **Graph** button at the bottom of the screen). In graph mode, the ▲ and ▼ keys translate the graph vertically, effectively changing the y-intercept of the line. Tap to change the magnitude of the increment for vertical translations. The ◀ and ▶ keys (as well as − and +) decrease and increase the slope. Press +/− to change the sign of the slope.



Y=aX+b

Y=0.5X−2

← →↑↓+− +/−

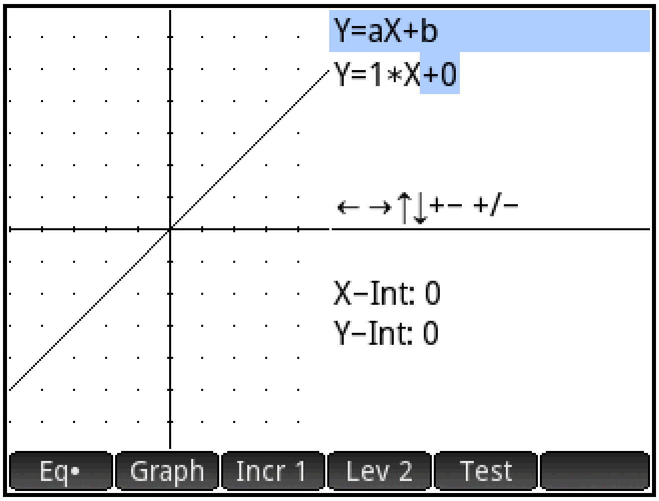
X-Int: 4

Y-Int: -2

The form of the linear function is shown at the top right of the display, with the current equation that matches the graph just below it. As you manipulate the graph, the equation updates to reflect the changes.

Equation mode

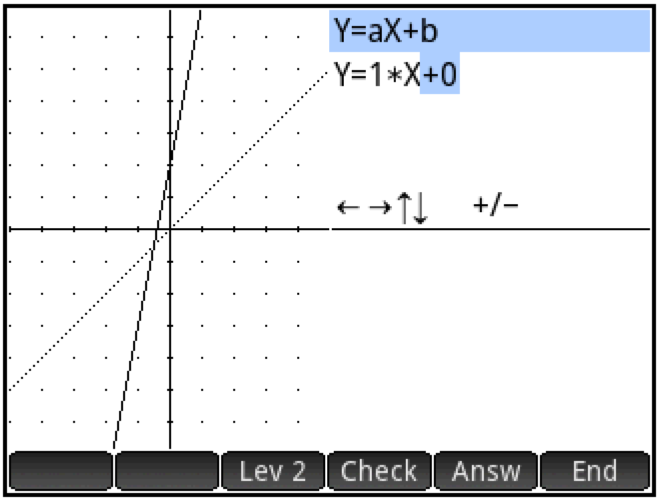
Tap Eq to enter equation mode. A dot will appear on the Eq button at the bottom of the screen.



In equation mode, you use the cursor keys to move between parameters in the equation and change their values, observing the effect on the graph displayed. Press ▼ or ▲ to decrease or increase the value of the selected parameter. Press ▶ or ◀ to select another parameter. Press +/− to change the sign of *a*.

Test mode

Tap Test to enter test mode. In Test mode you test your skill at matching an equation to the graph shown. Test mode is like equation mode in that you use the cursor keys to select and change the value of each parameter in the equation. The goal is to try to match the graph that is shown.



The app displays the graph of a randomly chosen linear function of the form dictated by your choice of level. (Tap Lev 1 or Lev 2 to change the level.) Now press the cursor keys to select a parameter and set its value. When you are ready, tap Check to see if you have correctly matched your equation to the given graph.

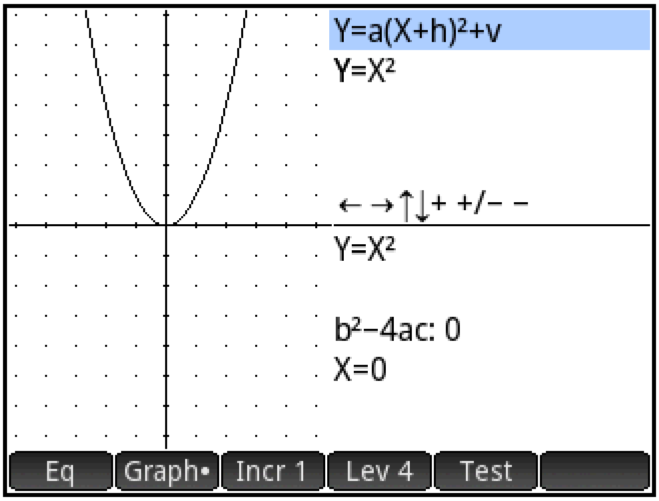
Tap Answ to see the correct answer and tap End to exit Test mode.

Quadratic Explorer app

The Quadratic Explorer app can be used to investigate the behavior of $y = a(x+h)^2 + v$ as the values of a, h and v change.

Open the app

Press Apps Info and select **Quadratic Explorer**.



Y=a(X+h)²+v
Y=X²

← →↑↓+ +/− −

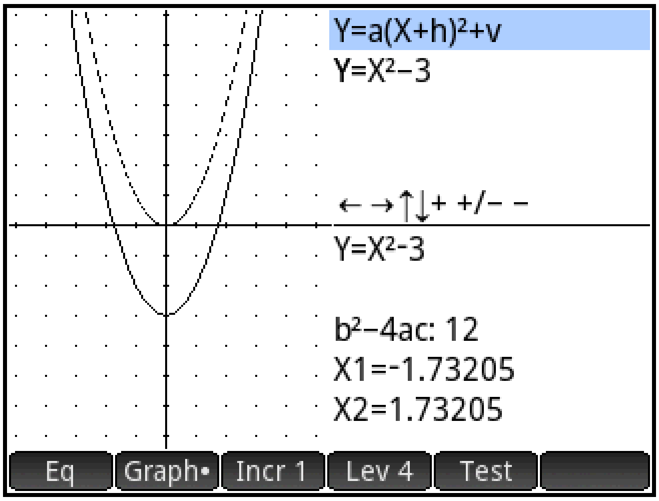
Y=X²

b²−4ac: 0
X=0

The left half of the display shows the graph of a quadratic function. The right half shows the general form of the equation being explored at the top and, below it, the current equation of that form. The keys you can use to manipulate the graph or equation appear below the equation. (These will change depending on the level of equation you choose.) Displayed beneath they keys is the equation, the discriminant (that is, $b^2–4ac$), and the roots of the quadratic.

Graph mode

The app opens in graph mode. In graph mode, you manipulate a copy of the graph using whatever keys are available. The original graph—converted to dotted lines—remains in place for you to easily see the result of your manipulations.



Y=a(X+h)²+v
Y=X²−3

← →↑↓+ +/− −

Y=X²-3

b²−4ac: 12
X1=-1.73205
X2=1.73205

Four general forms of quadratic equations are available for you to explore:

$y = ax^2$ [Level 1]

$y = (x+h)^2$ [Level 2]

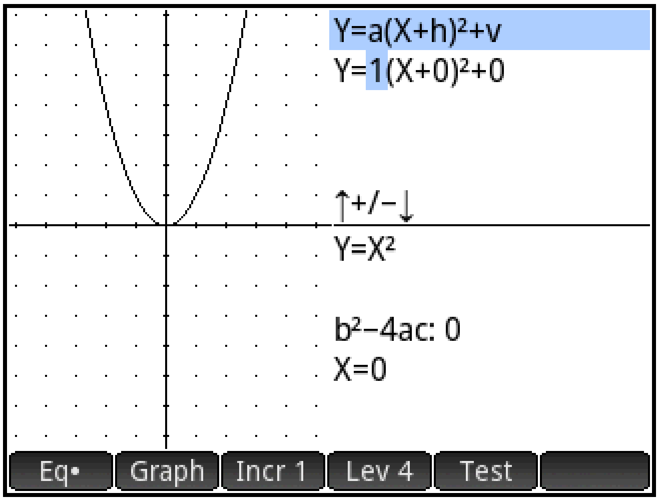
$y = x^2 + v$

$y = a(x+h)^2 + x$ [Level 4]

Choose a general form by tapping the Level button—Lev 1 Lev 2, and so on—until the form you want is displayed. The keys available to you to manipulate the graph vary from level to level.

Equation mode

Tap Eq to move to equation mode. In equation mode, you use the cursor keys to move between parameters in the equation and change their values, observing the effect on the graph displayed. Press ▼ or ▲ to decrease or increase the value of the selected parameter. Press ▶ or ◀ to select another parameter. Press +/− to change the sign. You have four forms (or levels) of graph, and the keys available for manipulating the equation depend on the level chosen.



$Y=a(X+h)^2+v$

$Y=1(X+0)^2+0$

↑+/−↓

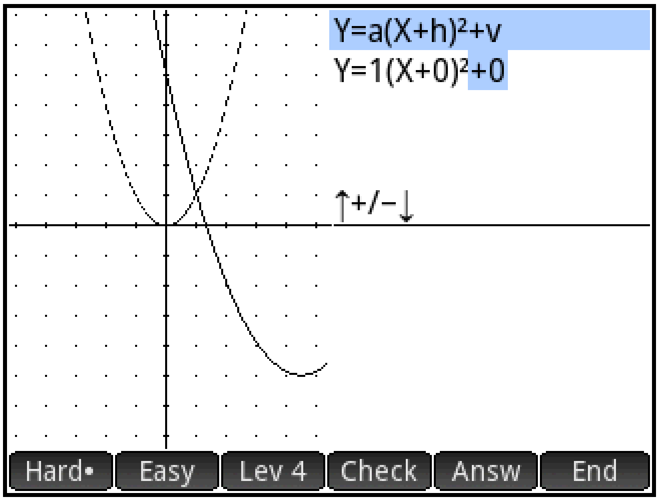
$Y=X^2$

b^2-4ac: 0

X=0

Test mode

Tap Test to enter test mode. In Test mode you test your skill at matching an equation to the graph shown. Test mode is like equation mode in that you use the cursor keys to select and change the value of each parameter in the equation. The goal is to try to match the graph that is shown.



$Y=a(X+h)^2+v$

$Y=1(X+0)^2+0$

The app displays the graph of a randomly chosen quadratic function. Tap the Level button to choose between one of four forms of quadratic equation. You can also choose graphs that are relatively easy to match or graphs that are harder match (by tapping Easy or Hard respectively).

Now press the cursor keys to select a parameter and set its value. When you are ready, tap Check to see if you have correctly matched your equation to the given graph.

Tap Answ to see the correct answer and tap End to exit Test mode.

Trig Explorer app

The Trig Explorer app can be used to investigate the behavior of the graphs $y = a \cdot sin(bx + c) + d$ and $y = a \cdot cos(bx + c) + d$ as the values of *a*, *b*, *c* and *d* change.

The menu items available in this app are:

- Eq or Graph: toggles between graph mode and equation mode.
- SIN or COS: toggles between sine and cosine graphs.
- Rad or Deg: toggles between radians and degrees as the angle measure for x.
- Orig or Extr: toggles between translating the graph (Orig), and changing its frequency or amplitude (Extr). You make these changes using the cursor keys.
- Test: enters test mode.
- π/9 or 20°: toggles the increment by which parameter values change: π/9, π/6, π/4, or 20°, 30°, 45° (depending on angle measure setting).

Open the app

Press Apps Info and select `Trig Explorer`.

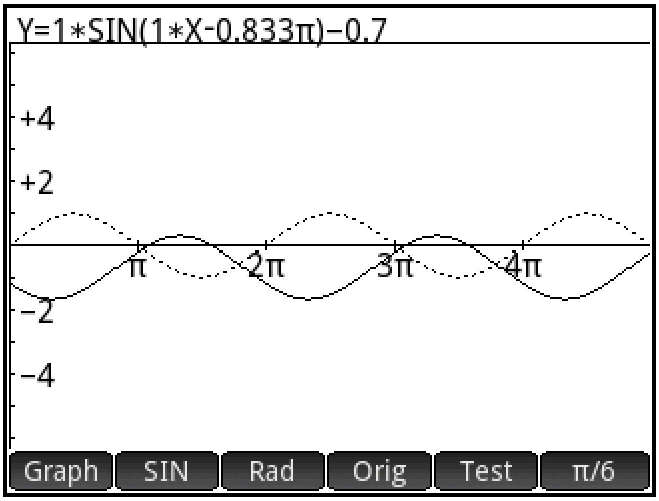
An equation is shown at the top of the display, with its graph shown below it.

COS

Choose the type of function you want to explore by tapping either SIN or COS.

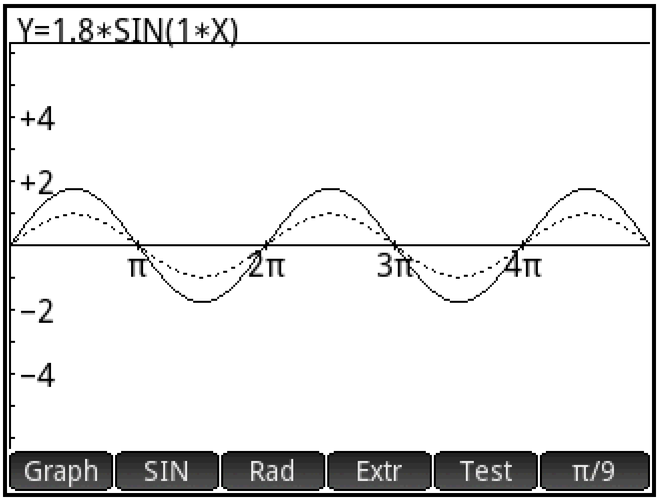
Graph mode

The app opens in graph mode. In graph mode, you manipulate a copy of the graph by pressing the cursor keys. All four keys are available. The original graph—converted to dotted lines—remains in place for you to easily see the result of your manipulations.



Y=1*SIN(1*X-0.833π)-0.7

When Orig is chosen, the cursor keys simply translate the graph horizontally and vertically. When Extr is chosen, pressing ▲ or ▼ changes the amplitude of the graph (that is, it is stretched or shrunk vertically); and pressing ◀ or ▶ changes the frequency of the graph (that is, it is stretched or shrunk horizontally).



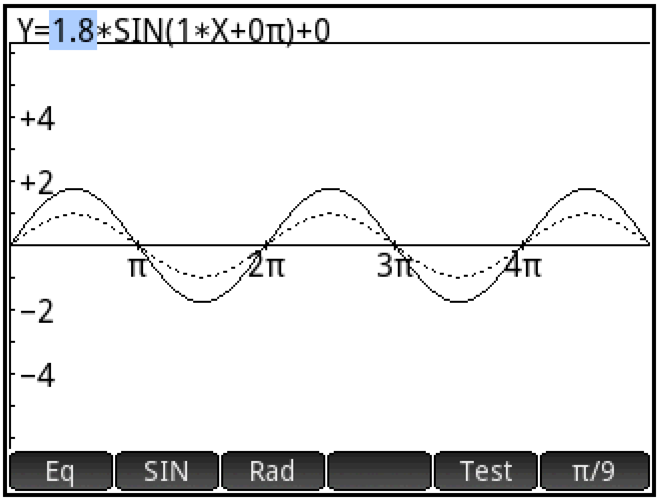
Y=1.8*SIN(1*X)

The π/9 or 20° button at the far right of the menu determines the increment by which the graph moves with each press of a cursor key. By default, the increment is set at π/9 or 20°.

Equation mode

Tap Graph to switch to equation mode. In equation mode, you use the cursor keys to move between parameters in the equation and change their values. You can then observe the effect on the graph displayed.

Press ▼ or ▲ to decrease or increase the value of the selected parameter. Press ▶ or ◀ to select another parameter.

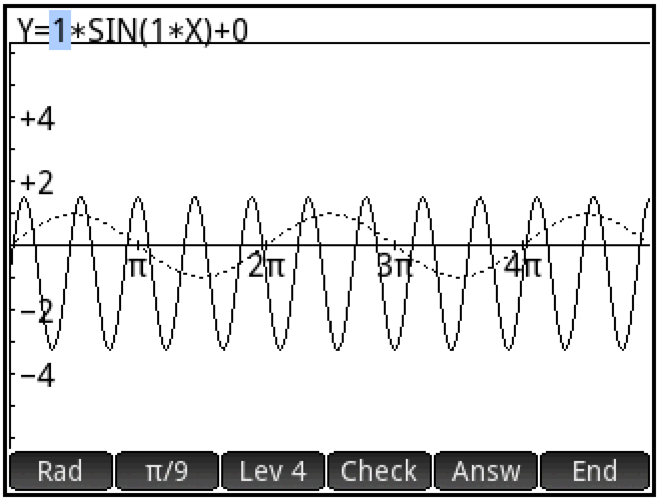


You can switch back to graph mode by tapping Eq.

Test mode

Tap Test to enter test mode. In test mode you test your skill at matching an equation to the graph shown. Test mode is like equation mode in that you use the cursor keys to select and change the value of one or more parameters in the equation. The goal is to try to match the graph that is shown.

The app displays the graph of a randomly chosen sinusoidal function. Tap a Level button—Lev 1, Lev 2 and so on—to choose between one of five types of sinusoidal equations.



Now press the cursor keys to select each parameter and set its value. When you are ready, tap Check to see if you have correctly matched your equation to the given graph.

Tap Answ to see the correct answer and tap Test to exit Test mode.

22 Functions and commands

Many mathematical functions are available from the calculator’s keyboard. These are described in “Keyboard functions” on page 101. Other functions and commands are collected together in the Toolbox menus (Mem B). There are five Toolbox menus:

Math

A collection of non-symbolic mathematical functions (see Math menu on page 352)

CAS

A collection of symbolic mathematical functions (see CAS menu on page 364)

App

A collection of app functions that can be called from elsewhere in the calculator, such as Home view, CAS view, the Spreadsheet app, and in a program (see App menu on page 385)

Note that the Geometry app functions can be called from elsewhere in the calculator, but they are designed to be used in the Geometry app. For that reason, the Geometry functions are not described in this chapter. They are described in the Geometry chapter.

User

The functions that you have created (see Creating your own functions on page 467) and the programs you have created that contain functions that have been exported.

Catlg

All the functions and commands:

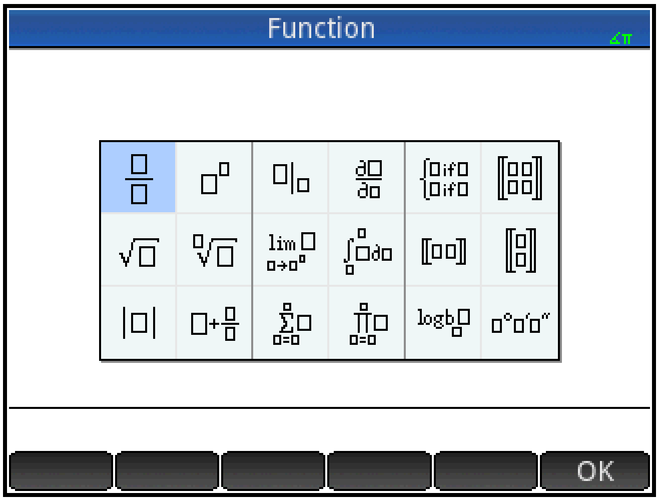
- on the **Math** menu
- on the **CAS** menu
- used in the Geometry app
- used in programming
- used in the Matrix Editor
- used in the List Editor
- and some additional functions and commands

See Ctlg menu on page 416.

Although the Catlg menu includes all the programming commands, the Commands menu (Cmds) in the Program Editor contains all the programming commands grouped by category. It also contains the Template menu (Tmplt), which contains the common programming structures.

NOTE: Some functions can be chosen from the math template (displayed by pressing Units).

You can also create your own functions. See Creating your own functions on page 467.



Setting the form of menu items

You can choose to have entries on the Math and CAS menus presented either by their descriptive name or their command name. (The entries on the Catlg menu are always presented by their command name.)

Descriptive name	Command name
Factor List	ifactors
Complex Zeros	cZeros
Groebner Basis	gbasis
Factor by Degree	factor_xn
Find Roots	proot

The default menu presentation mode is to provide the descriptive names for the Math and CAS functions. If you prefer the functions to be presented by their command name, deselect the **Menu Display** option on the second page of the Home Settings screen.

Abbreviations used in this chapter

In describing the syntax of functions and commands, the following abbreviations and conventions are used:

`Eqn:` an equation

`Expr:` a mathematical expression

`Fnc:` a function

`Frac:` a fraction

`Intgr:` an integer

`Obj:` signifies that objects of more than one type are allowable here

`Poly:` a polynomial

`RatFrac:` a rational fraction

`Val:` a real value

`Var:` a variable

Parameters that are optional are given in square brackets, as in `NORMAL_ICDF([μ,σ,]p)`.

For ease of reading, commas are used to separate parameters, but these are only necessary to separate parameters. Thus a single-parameter command needs no comma after the parameter even if, in the syntax shown below, there is a comma between it and an optional parameter. An example is the syntax `zeros(Expr,[Var])`. The comma is needed only if you are specifying the optional parameter `Var`.

Keyboard functions

The most frequently used functions are available directly from the keyboard. Many of the keyboard functions also accept complex numbers as arguments. Enter the keys and inputs shown below and press Enter to evaluate the expression.

NOTE: In the examples below, shifted functions are represented by the actual keys to be pressed, with the function name shown in parentheses. For example, Shift SIN (ASIN) means that to make an arc sine calculation (ASIN), you press Shift SIN.

The examples below show the results you would get in Home view. If you are in the CAS, the results are given in simplified symbolic format. For example:

Shift x^2 320 returns 17.88854382 in Home view, and 8*√5 in the CAS.

+ − × ÷

Add, subtract, multiply, divide. Also accepts complex numbers, lists, and matrices.

value1 + value2, etc

LN

Natural logarithm. Also accepts complex numbers.

LN(value)

Example:

`LN(1)` returns 0

Shift LN (e^x)

Natural exponential. Also accepts complex numbers.

$\mathrm{e}^{\mathrm{value}}$

Example:

`e`^{5} returns 148.413159103

LOG

Common logarithm. Also accepts complex numbers.

`LOG`*(value)*

Example:

`LOG(100)` returns 2

Shift LOG (10^x)

Common exponential (antilogarithm). Also accepts complex numbers.

`ALOG`(*value*)

Example:

`ALOG(3)` returns 1000

SIN COS TAN

The basic trigonometric functions sine, cosine, and tangent.

`SIN`(*value*)

`COS`(*value*)

`TAN`(*value*)

Example:

`TAN(45)` returns 1 (degrees mode)

Shift SIN (ASIN)

Arc sine: $\sin^{-1}x$. Output range is from –90° to 90° or –π/2 to π/2. Inputs and outputs depend on the current angle format. Also accepts complex numbers.

`ASIN`(*value*)

Example:

`ASIN(1)` returns 90 (degrees mode)

Shift COS (ACOS)

Arc cosine: $\cos^{-1}x$. Output range is from 0° to 180° or 0 to π. Inputs and outputs depend on the current angle format. Also accepts complex numbers. Output will be complex for values outside the normal cosine domain of $-1 \le x \le 1$.

`ACOS`(*value*)

Example:

`ACOS(1)` returns 0 (degrees mode)

Shift TAN (ATAN)

Arc tangent: $\tan^{-1}(x)$. Output range is from –90° to 90° or –π/2 to π/2. Inputs and outputs depend on the current angle format. Also accepts complex numbers.

`ATAN`(*value*)

Example:

`ATAN(1)` returns 45 (degrees mode)

x^2

Square. Also accepts complex numbers.

value^2

Example:

`18`^{2} returns 324

Shift x^2

Square root. Also accepts complex numbers.

`√value`

Example:

`√320` returns 17.88854382

x^y

x raised to the power of y. Also accepts complex numbers.

value^{power}

Example:

2^8 returns 256

Shift x^y

The nth root of x.

`root√value`

Example:

`3√8` returns 2

Shift ÷

Reciprocal.

value^{-1}

Example:

`3`$^{-1}$ returns .333333333333

+/-

Negation. Also accepts complex numbers.

-value

Example:

`–(1+2*i)` returns -1-2*i

Shift +/- (|x|)

Absolute value.

`|value|`

`|x+y*i|`

`|matrix|`

For a complex number, |x+y*i| returns $\sqrt{x^2+y^2}$. For a matrix, |matrix| returns the Frobenius norm of the matrix.

Example:

`|–1|` returns 1

`|(1,2)|` returns 2.2360679775

You can also use ABS() and abs() as alternative syntax forms, although they give slightly different results for some inputs. For example, abs(matix) returns the 12norm of the matrix.

a b/c

Decimal to fraction conversion. In Home view, toggles the last entry in Home view between decimal, fraction, and mixed number forms. If a result from the History is selected, then it toggles the selection through these forms. Also works with lists and matrices. In CAS view, it only toggles between decimal and fractional equivalents, and adds them as new entries to the History.

Example:

In Home view, with 2.4 as the last entry in History or selected in History, press a b/c to see 12/5; press a b/c again to see 2+2/5; press a b/c again to return to 2.4.

Shift a b/c

Decimal to hexagesimal conversion. In Home view, toggles the last entry in Home view between decimal and hexagesimal forms. If a result from the History is selected, then it toggles the selection through these forms. Also works with lists and matrices. In CAS view, it adds them as new entries to the History.

Example:

In Home view, with 2.4 as the last entry in History or selected in History, press Shift a b/c to see 2°24′0″; press Shift a b/c again to return to 2.4.

EEX

The EEX key is used to enter numbers in exponential notation.

On an HP Prime calculator, a number in exponential notation is represented by two parts separated by the E character, which corresponds to the EEX key. The first part, or mantissa, is a real number. The second part, or the exponent, is an integer. The number represented by this notation is mantissa*10^exponent.

For example, pressing 3 EEX 5 displays 3E5 on the command line. This returns the number 300,000.

Example:

`3E2` returns 300

Shift 9

To open a menu of commonly used mathematical symbols and Greek characters, press either Shift 9 or Shift Vars.

Shift 6

To open a menu of common Boolean operators, press Shift 6. These operators can also be found in the catalog.

Shift 2

The imaginary unit i.

Inserts the imaginary unit i.

Shift 3

The constant π.

Inserts the transcendental constant π.

Math menu

Press Mem B to open the Toolbox menus (one of which is the Math menu). The functions and commands available on the Math menu are listed as they are categorized on the menu.

Numbers

Ceiling

Smallest integer greater than or equal to value.

`CEILING(value)`

Examples:

`CEILING(3.2)` returns 4

`CEILING(-3.2)` returns -3

Floor

Greatest integer less than or equal to value.

`FLOOR(value)`

Examples:

`FLOOR(3.2)` returns 3

`FLOOR(-3.2)` returns -4

IP

Integer part.

`IP(value)`

Example:

`IP(23.2) returns 23`

FP

Fractional part.

`FP(value)`

Example:

`FP (23.2) returns .2`

Round

Rounds value to decimal places. Also accepts complex numbers.

`ROUND(value,places)`

`ROUND` can also round to a number of significant digits if places is a negative integer (as shown in the second example below).

Examples:

`ROUND(7.8676,2)` returns 7.87

`ROUND(0.0036757,-3)` returns 0.00368

Truncate

Truncates value to decimal places. Also accepts complex numbers.

`TRUNCATE(value,places)`

Examples:

`TRUNCATE(2.3678,2)` returns 2.36

`TRUNCATE(0.0036757,-3)` returns 0.00367

Mantissa

Mantissa—that is, the significant digits—of value, where value is a floating-point number.

`MANT(value)`

Example:

`MANT(21.2E34)` returns 2.12

Exponent

Exponent of value. That is, the integer component of the power of 10 that generates value.

`XPON(value)`

Example:

`XPON(123456)` returns 5 (since 105.0915... equals 123456)

Arithmetic

Maximum

Maximum. The greater of two values.

`MAX(value1,value2)`

Example:

`MAX(8/3,11/4)` returns 2.75

Note that in Home view a non-integer result is given as a decimal fraction. If you want to see the result as a common fraction, press a b/c ≐ . This key cycles through decimal, fraction, and mixed number representations. Or, if you prefer, press CAS Settings . This opens the computer algebra system. If you want to return to Home view to make further calculations, press ⌂ Settings .

Minimum

Minimum. Returns the least of the values given, or the least value of a list.

`MIN(value1,value2)`

Example:

`MIN(210,25)` returns 25

Modulus

Modulo. The remainder of value1/value2.

```
value1 MOD value2
```

Example:

`74 MOD 5` returns 4

Find Root

Function root-finder (like the Solve app). Finds the value for the given variable at which expression most nearly evaluates to zero. Uses guess as initial estimate.

```
FNROOT(expression,variable,guess)
```

Example:

`FNROOT((A*9.8/600)-1,A,1)` returns 61.2244897959.

Percentage

x percent of y; that is, x/100*y.

```
%(x,y)
```

Example:

`%(20,50)` returns 10

Arithmetic – Complex

Argument

Argument. Finds the angle defined by a complex number. Inputs and outputs use the current angle format set in Home modes.

```
ARG(x+y*i)
```

Example:

`ARG(3+3*i)` returns 45 (degrees mode)

Conjugate

Complex conjugate. Conjugation is the negation (sign reversal) of the imaginary part of a complex number.

```
CONJ(x+y*i)
```

Example:

```
CONJ(3+4*i) returns (3-4*i)
```

Real Part

Real part x, of a complex number, (x+y*i).

`RE(x+y*i)`

Example:

`RE(3+4*i)` returns 3

Imaginary Part

Imaginary part, y, of a complex number, (x+y*i).

`IM(x+y*i)`

Example:

`IM(3+4*i)` returns 4

Unit Vector

Sign of value. If positive, the result is 1. If negative, –1. If zero, result is zero. For a complex number, this is the unit vector in the direction of the number.

`SIGN(value)`

`SIGN((x,y))`

Examples:

`SIGN(POLYEVAL([1,2,-25,-26,2],-2))` returns –1

`SIGN((3,4))` returns (.6+.8i)

Arithmetic – Exponential

ALOG

Antilogarithm (exponential).

`ALOG(value)`

EXPM1

Exponential minus 1: e^x-1.

`EXPM1(value)`

LNP1

Natural log plus 1: ln(x+1).

`LNP1(value)`

Trigonometry

The trigonometry functions can also take complex numbers as arguments. For SIN, COS, TAN, ASIN, ACOS, and ATAN, see Keyboard functions on page 347.

CSC

Cosecant: 1/sin(x).

`CSC(value)`

ACSC

Arc cosecant: csc^{-1}(x).

```
ACSC(value)
```

SEC

Secant: 1/cos(x).

```
SEC(value)
```

ASEC

Arc secant: sec^{-1}(x).

```
ASEC(value)
```

COT

Cotangent: cos(x)/sin(x)

```
COT(value)
```

ACOT

Arc cotangent: cot^{-1}(x).

```
ACOT(value)
```

Hyperbolic

The hyperbolic trigonometry functions can also take complex numbers as arguments.

SINH

Hyperbolic sine.

```
SINH(value)
```

ASINH

Inverse hyperbolic sine: sinh^{-1}x.

```
ASINH(value)
```

COSH

Hyperbolic cosine

```
COSH(value)
```

ACOSH

Inverse hyperbolic cosine: cosh^{-1}x.

```
ACOSH(value)
```

TANH

Hyperbolic tangent.

```
TANH(value)
```

ATANH

Inverse hyperbolic tangent: $\tanh^{-1}x$.

```
ATANH(value)
```

Probability

Factorial

Factorial of a positive integer. For non-integers, x! = Γ(x + 1). This calculates the gamma function.

```
value!
```

Example:

`5!` returns 120

Combination

The number of combinations (without regard to order) of n things taken r at a time.

```
COMB(n,r)
```

Example: Suppose you want to know how many ways five things can be combined two at a time.

`COMB(5,2)` returns 10

Permutation

Number of permutations (with regard to order) of n things taken r at a time: n!/(n–r)!.

```
PERM (n,r)
```

Example: Suppose you want to know how many permutations there are for five things taken two at a time.

`PERM(5,2)` returns 20

Probability – Random

Number

Random number. With no argument, this function returns a random number between zero and one. With one argument a, it returns a random number between 0 and a. With two arguments, a, and b, returns a random number between a and b. With three arguments, n, a, and b, returns n random number between a and b.

```
RANDOM
```

```
RANDOM(a)
```

```
RANDOM(a,b)
```

```
RANDOM(n,a,b)
```

Integer

Random integer. With no argument, this function returns either 0 or 1 randomly. With one integer argument a, it returns a random integer between 0 and a. With two arguments, a, and b, returns a random integer between a and b. With three integer arguments, n, a, and b, returns n random integers between a and b.

```
RANDINT
```

```
RANDINT(a)
```

```
RANDINT(a,b)
```

```
RANDINT(n,a,b)
```

Normal

Random normal. Generates a random number from a normal distribution.

```
RANDNORM(μ,σ)
```

Example:

`RANDNORM(0,1)` returns a random number from the standard Normal distribution.

Seed

Sets the seed value on which the random functions operate. By specifying the same seed value on two or more calculators, you ensure that the same random numbers appear on each calculator when the random functions are executed.

```
RANDSEED(value)
```

Probability – Density

Normal

Normal probability density function. Computes the probability density at value x, given the mean, μ, and standard deviation, σ, of a normal distribution. If only one argument is supplied, it is taken as x, and the assumption is that μ=0 and σ=1.

```
NORMALD([μ,σ,]x)
```

Example:

`NORMALD(0.5)` and `NORMALD(0,1,0.5)` both return 0.352065326764.

T

Student's t probability density function. Computes the probability density of the Student's t-distribution at x, given n degrees of freedom.

```
STUDENT(n,x)
```

Example:

`STUDENT(3,5.2)` returns 0.00366574413491.

χ^2

χ^2 probability density function. Computes the probability density of the x^2 distribution at x, given n degrees of freedom.

```
CHISQUARE(n,x)
```

Example:

`CHISQUARE(2,3.2)` returns 0.100948258997.

F

Fisher (or Fisher–Snedecor) probability density function. Computes the probability density at the value x, given numerator n and denominator d degrees of freedom.

```
FISHER(n,d,x)
```

Example:

`FISHER(5,5,2)` returns 0.158080231095.

Binomial

Binomial probability density function. Computes the probability of k successes out of n trials, each with a probability of success of p. Returns Comb(n,k) if there is no third argument. Note that n and k are integers with k≤n.

```
BINOMIAL(n, p, k)
```

Example: Suppose you want to know the probability that just 6 heads would appear during 20 tosses of a fair coin.

`BINOMIAL(20, 0.5, 6)` returns 0.0369644165039.

Geometric

Geometric probability density function. Computes the probability density of the geometric distribution at x, given probability p.

```
GEOMETRIC(p, x)
```

Example:

`GEOMETRIC(0.3, 4)` returns 0.1029.

Poisson

Poisson probability mass function. Computes the probability of k occurrences of an event during a future interval given μ, the mean of the occurrences of that event during that interval in the past. For this function, k is a non-negative integer and μ is a real number.

```
POISSON(μ,k)
```

Example: Suppose that on average you get 20 emails a day. What is the probability that tomorrow you will get 15?

`POISSON(20,15)` returns 0.0516488535318.

Probability – Cumulative

Normal

Cumulative normal distribution function. Returns the lower-tail probability of the normal probability density function for the value x, given the mean, μ, and standard deviation, σ, of a normal distribution. If only one argument is supplied, it is taken as x, and the assumption is that μ=0 and σ=1.

```
NORMALD_CDF([μ,σ,]x)
```

Example:

`NORMALD_CDF(0,1,2) returns` 0.977249868052.

T

Cumulative Student's t distribution function. Returns the lower-tail probability of the Student's t-probability density function at x, given n degrees of freedom.

```
STUDENT_CDF(n,x)
```

Example:

`STUDENT_CDF(3,-3.2)` returns 0.0246659214814.

X^2

Cumulative X^2 distribution function. Returns the lower-tail probability of the X^2 probability density function for the value X, given n degrees of freedom.

`CHISQUARE_CDF(n,k)`

Example:

`CHISQUARE_CDF(2, 6.3)` returns 0.957147873133.

F

Cumulative Fisher distribution function. Returns the lower-tail probability of the Fisher probability density function for the value x, given numerator n and denominator d degrees of freedom.

`FISHER_CDF(n,d,x)`

Example:

`FISHER_CDF(5,5,2)` returns 0.76748868087.

Binomial

Cumulative binomial distribution function. Returns the probability of k or fewer successes out of n trials, with a probability of success, p for each trial. Note that n and k are integers with $k \leq n$.

`BINOMIAL_CDF(n,p,k)`

Example: Suppose you want to know the probability that during 20 tosses of a fair coin you will get either 0, 1, 2, 3, 4, 5, or 6 heads.

`BINOMIAL_CDF(20,0.5,6)` returns 0.05765914917.

Geometric

Cumulative geometric distribution function. With two values (p and x), returns the lower-tail probability of the geometric probability density function for the value x, given probability p. With three values (p, x_1, and x_2), returns the area under the geometric probability density function defined by the probability p, between x_1 and x_2.

`GEOMETRIC_CDF(p, x)`

GEOMETRIC_CDF(p, x_1, x_2)

Examples:

`GEOMETRIC_CDF (0.3, 4)` returns 0.7599.

`GEOMETRIC_CDF (0.5, 1, 3)` returns 0.375.

Poisson

Cumulative Poisson distribution function. Returns the probability x or fewer occurrences of an event in a given time interval, given expected occurrences.

`POISSON_CDF( ,x)`

Example:

`POISSON_CDF(4,2)` returns 0.238103305554.

Probability – Inverse

Normal

Inverse cumulative normal distribution function. Returns the cumulative normal distribution value associated with the lower-tail probability, p, given the mean, μ, and standard deviation, σ, of a normal distribution. If only one argument is supplied, it is taken as p, and the assumption is that μ=0 and σ=1.

```
NORMALD_ICDF([μ,σ,]p)
```

Example:

`NORMALD_ICDF(0,1,0.841344746069)` returns 1.

T

Inverse cumulative Student's t distribution function. Returns the value x such that the Student's-t lower-tail probability of x, with n degrees of freedom, is p.

```
STUDENT_ICDF(n,p)
```

Example:

`STUDENT_ICDF(3,0.0246659214814)` returns –3.2.

χ^2

Inverse cumulative χ^2 distribution function. Returns the value χ such that the χ^2 lower-tail probability of χ, with n degrees of freedom, is p.

```
CHISQUARE_ICDF(n,p)
```

Example:

`CHISQUARE_ICDF(2, 0.957147873133)` returns 6.3.

F

Inverse cumulative Fisher distribution function. Returns the value x such that the Fisher lower-tail probability of x, with numerator n and denominator d degrees of freedom, is p.

```
FISHER_ICDF(n,d,p)
```

Example:

`FISHER_ICDF(5,5,0.76748868087)` returns 2.

Binomial

Inverse cumulative binomial distribution function. Returns the number of successes, k, out of n trials, each with a probability of p, such that the probability of k or fewer successes is q.

```
BINOMIAL_ICDF(n,p,q)
```

Example:

`BINOMIAL_ICDF(20,0.5,0.6)` returns 11.

Geometric

Inverse cumulative geometric distribution function. Returns the x value that has the lower-tail probability value k, given the probability p.

```
GEOMETRIC_ICDF(p, k)
```

Example:

`GEOMETRIC_ICDF(0.3, 0.95)` returns 9.

Poisson

Inverse cumulative Poisson distribution function. Returns the value x such that the probability of x or fewer occurrences of an event, with μ expected (or mean) occurrences of the event in the interval, is p.

```
POISSON_ICDF( ,p)
```

Example:

`POISSON_ICDF(4,0.238103305554)` returns 3.

List

These functions work on data in a list. For more information, see the Lists chapter of the *Prime Calculator User Guide*.

Matrix

These functions work on matrix data stored in matrix variables. For more information, see the Matrices chapter of the *Prime Calculator User Guide*.

Special

Beta

Returns the value of the beta function (B) for two numbers a and b

```
Beta(a,b)
```

Gamma

Returns the value of the gamma function (Γ) for a number a.

```
Gamma(a)
```

Psi

Returns the value of the nth derivative of the digamma function at x=a, where the digamma function is the first derivative of ln(Γ(x)).

```
Psi(a,n)
```

Zeta

Returns the value of the zeta function (Z) for a real x.

```
Zeta(x)
```

erf

Returns the floating point value of the error function at x=a.

```
erf(a)
```

erfc

Returns the value of the complementary error function at x=a.

```
erfc(a)
```

Ei

Returns the exponential integral of an expression.

```
Ei(Expr)
```

Si

Returns the sine integral of an expression.

```
Si(Expr)
```

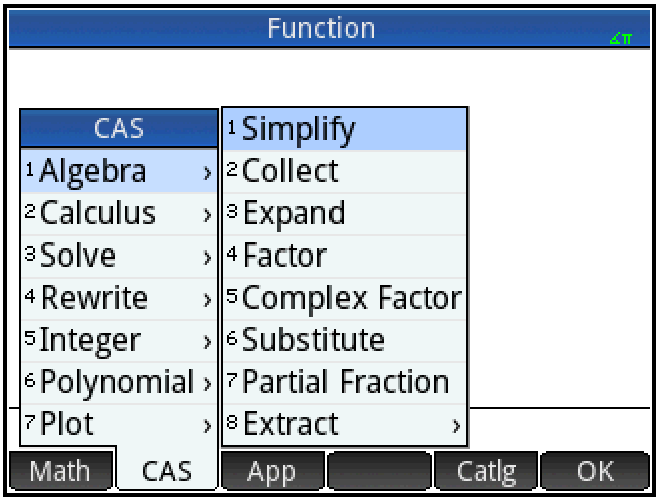
Ci

Returns the cosine integral of an expression.

```
Ci(Expr)
```

CAS menu

Press Mem B to open the Toolbox menus (one of which is the CAS menu). The functions on the CAS menu are those most commonly used. Many more functions are available. See Ctlg menu on page 416. Note that the Geometry functions appear on the App menu.



The result of a CAS command may vary depending on the CAS settings. The examples in this chapter assume the default CAS settings unless otherwise noted.

Algebra

Simplify

Returns an expression simplified.

```
simplify(Expr)
```

Example:

`simplify(4*atan(1/5)-atan(1/239))yields (1/4)*pi`

Collect

Collects like terms in a polynomial expression (or in a list of polynomial expressions). Factorizes the results, depending on the CAS settings.

`collect(Poly)` or `collect({Poly1, Poly2,..., Polyn})`

Examples:

`collect(x+2*x+1-4)` returns 3*x-3

`collect(x^2-9*x+5*x+3+1)` returns (x-2)^2

Expand

Returns an expression expanded.

```
expand(Expr)
```

Example:

`expand((x+y)*(z+1))` gives y*z+x*z+y+x

Factor

Returns a polynomial factorized.

```
factor(Poly)
```

Example:

`factor(x^4-1)` gives (x-1)*(x+1)*(x^2+1)

Substitute

Substitutes a value for a variable in an expression.

```
Syntax: subst(Expr,Var=value)
```

Example:

`subst(x/(4-x^2),x=3)` returns -3/5

Partial Fraction

Performs partial fraction decomposition on a fraction.

`partfrac(RatFrac` or `Opt)`

Example:

`partfrac(x/(4-x^2))` returns (-1/2)/(x-2)-(1/2)/((x+2)

Algebra – Extract

Numerator

Simplified Numerator. For the integers a and b, returns the numerator of the fraction a/b after simplification.

```
numer(a/b)
```

Example:

`numer(10/12)` returns 5

Denominator

Simplified Denominator. For the integers a and b, returns the denominator of the fraction a/b after simplification.

```
denom(a/b)
```

Example:

`denom(10/12)` returns 6

Left Side

Returns the left side of an equation or the left end of an interval.

`left(Expr1=Expr2)` or `left(Real1..Real2)`

Example:

`left(x^2-1=2*x+3)` returns x^2-1

Right Side

Returns the right side of an equation or the right end of an interval.

`right(Expr1=Expr2)` or `right(Real1..Real2)`

Example:

`right(x^2-1=2*x+3)` returns 2*x+3

Calculus

Differentiate

With one expression as argument, returns derivative of the expression with respect to x. With one expression and one variable as arguments, returns the derivative or partial derivative of the expression with respect to the variable. With one expression and more than one variable as arguments, returns the derivative of the expression with respect to the variables in the second argument. These arguments can be followed by $k (k is an integer) to indicate the number of times the expression should be derived with respect to the variable. For example, diff(exp(x*y),x$3,y$2,z) is the same as diff(exp(x*y),x,x,x,y,y,z).

```
diff(Expr,[var])
```

or

```
diff(Expr,var1$k1,var2$k2,...)
```

Example:

`diff(x^3-x)` gives 3*x^2-1

Integrate

Returns the integral of an expression. With one expression as argument, returns the indefinite integral with respect to x. With the optional second, third, and fourth arguments, you can specify the variable of integration and the bounds for a definite integral.

```
int(Expr,[Var(x)],[Real(a)],[Real(b)])
```

Example:

```
int(1/x) gives ln(abs(x))
```

Limit

Returns the limit of an expression when the variable approaches a limit point a or +/– infinity. With the optional fourth argument you can specify whether it is the limit from below, above or bidirectional (–1 for limit from below, +1 for limit from above, and 0 for bidirectional limit). If the fourth argument is not provided, the limit returned is bidirectional. The limit function can return ±∞, which refers to complex infinity, an infinite number in the complex plane whose argument is unknown. In the context of a limit, complex infinity is usually construed as meaning the limit is undefined.

```
limit(Expr,Var,Val,[Dir(1, 0, -1)])
```

Example:

`limit((n*tan(x)-tan(n*x))/(sin(n*x)-n*sin(x)),x,0)` returns 2

For example, lim(1/x, x, 0) returns ±∞; this is mathematically correct and in this case indicates the limit is undefined.

Series

Returns the series expansion of an expression in the vicinity of a given equality variable. With the optional third and fourth arguments you can specify the order and direction of the series expansion. If no order is specified the series returned is fifth order. If no direction is specified, the series is bidirectional.

```
series(Expr,Equal(var=limit_point),[Order],[Dir(1,0,-1)])
```

Example:

`series((x^4+x+2)/(x^2+1),x=0,5)` gives 2+x-2x^2-x^3+3x^4+x^5+x^6*order_size(x)

Summation

Returns the discrete sum of Expr with respect to the variable Var from Real1 to Real2. You can also use the summation template in the Template menu. With only the first two arguments, returns the discrete antiderivative of the expression with respect to the variable.

```
sum(Expr,Var,Real1, Real2,[Step])
```

Example:

`sum(n^2,n,1,5)` returns 55

Calculus – Differential

Curl

Returns the rotational curl of a vector field. Curl([A B C], [x y z]) is defined to be [dC/dy-dB/dz dA/dz-dC/dx dB/dx-dA/dy].

```
curl([Expr1, Expr2, …, ExprN], [Var1, Var2, …, VarN])
```

Example:

`curl([2*x*y,x*z,y*z],[x,y,z])` returns [z-x,0,z- 2*x]

Divergence

Returns the divergence of a vector field, defined by:

divergence([A,B,C],[x,y,z])=dA/dx+dB/dy+dC/dz.

```
divergence([Expr1, Expr2, …, ExprN], [Var1, Var2, …, VarN])
```

Example:

`divergence([x^2+y,x+z+y,z^3+x^2],[x,y,z])` gives 2*x+3*z^2+1

Gradient

Returns the gradient of an expression. With a list of variables as second argument, returns the vector of partial derivatives.

```
grad(Expr,LstVar)
```

Example:

`grad(2*x^2*y-x*z^3,[x,y,z])` gives [2*2*x*y-z^3,2*x^2,-x*3*z^2]

Hessian

Returns the Hessian matrix of an expression.

```
hessian(Expr,LstVar)
```

Example:

`hessian(2*x^2*y-x*z,[x,y,z])` gives [[4*y,4*x,-1],[2*2*x,0,0],[-1,0,0]]

Calculus – Integral

By Parts u

Performs integration by parts of the expression f(x)=u(x)*v'(x), with f(x) as the first argument and u(x) (or 0) as the second argument. Specifically, returns a vector whose first element is u(x)*v(x) and whose second element is v(x)*u'(x). With the optional third, fourth and fifth arguments you can specify a variable of integration and bounds of the integration. If no variable of integration is provided, it is taken as x.

```
ibpu(f(Var), u(Var), [Var], [Real1], [Real2])
```

Example:

`ibpu(x*ln(x), x)` returns [x*(x*ln(x) –x*ln(x)+x]

By Parts v

Performs integration by parts of the expression f(x)=u(x)*v'(x), with f(x) as the first argument and v(x) (or 0) as the second argument. Specifically, returns a vector whose first element is u(x)*v(x) and whose second element is v(x)*u'(x). With the optional third, fourth and fifth arguments you can specify a variable of integration and bounds of the integration. If no variable of integration is provided, it is taken as x.

```
ibpdv(f(Var), v(Var), [Var], [Real1], [Real2])
```

Example:

`ibpdv(ln(x),x)` gives x*ln(x)-x

F(b)–F(a)

Returns F(b)–F(a).

```
preval(Expr(F(var)),Real(a),Real(b),[Var])
```

Example:

`preval(x^2-2,2,3)` gives 5

Calculus – Limits

Riemann Sum

Returns an equivalent of the sum of Expr for var2 from var2=1 to var2=var1 (in the neighborhood of n=+∞) when the sum is looked at as a Riemann sum associated with a continuous function defined on [0,1].

```
sum_riemann(Expr, [Var1 Var2])
```

Example:

`sum_riemann(1/(n+k),[n,k])` gives ln(2)

Taylor

Returns the Taylor series expansion of an expression at a point or at infinity (by default, at x=0 and with relative order=5).

```
taylor(Expr,[Var=Value],[Order])
```

Example:

`taylor(sin(x)/x,x=0)` returns 1-(1/6)*x^2+(1/120)*x^4+x^6*order_size(x)

Taylor of Quotient

Returns the n-degree Taylor polynomial for the quotient of 2 polynomials.

```
divpc(Poly1,Poly2,Integer)
```

Example:

`divpc(x^4+x+2,x^2+1,5)` returns the 5th-degree polynomial x^5+3*x^4-x^3-2*x^2+x+2

Calculus – Transform

Laplace

Returns the Laplace transform of an expression.

```
laplace(Expr,[Var],[LapVar])
```

Example:

`laplace(exp(x)*sin(x))` gives 1/(x^2-2*x+2)

Inverse Laplace

Returns the inverse Laplace transform of an expression.

```
ilaplace(Expr,[Var],[IlapVar])
```

Example:

`ilaplace(1/(x^2+1)^2)` returns ((-x)*cos(x))/2+sin(x)/2

FFT

With one argument (a vector), returns the discrete Fourier transform in R.

```
fft(Vect)
```

With two additional integer arguments a and p, returns the discrete Fourier transform in the field Z/pZ, with a as primitive nth root of 1 (n=size(vector)).

```
fft((Vector, a, p)
```

Example:

`fft([1,2,3,4,0,0,0,0])` gives [10.0,-0.414213562373-7.24264068712*(i),-2.0+2.0*i, 2.41421356237-1.24264068712*i,-2.0,2.41421356237+1.24264068712*i,-2.0-2.0*i]

Inverse FFT

Returns the inverse discrete Fourier transform.

```
ifft(Vector)
```

Example:

`ifft([100.0,-52.2842712475+6*i,-8.0*i,4.28427124746-6*i, 4.0,4.28427124746+6*i,8*i,-52.2842712475-6*i])` gives [0.99999999999,3.99999999999,10.0,20.0,25.0,24.0,16.0,-6.39843733552e-12]

Solve

Solve

Returns a list of the solutions (real and complex) to a polynomial equation or a set of polynomial equations.

`solve(Eq,[Var])` or `solve({Eq1, Eq2,…}, [Var])` or `solve(Eq, Var=Guess)` or `solve(Eq, Var=Val1…Val2)`

For optimal results, if the solution is known to be an approximation, either enter a guess or define an interval for the software to search for a solution.

To enter an initial value for the solver, use the syntax `Var=Guess`.

To define the closed interval [Val1, Val2], use the syntax `Var=Val1…Val2`.

Examples:

`solve(x^2-3=1)` returns {-2,2}

`solve({x^2-3=1, x+2=0},x)` returns {-2}

`solve(x^2-(LN(x)+5)=0, x=2)` and `solve(x^2-(LN(x)+5)=0, x=2…3)` both return 2.42617293082

Zeros

With an expression as argument, returns the real zeros of the expression; that is, the solutions when the expression is set equal to zero.

With a list of expressions as argument, returns the matrix where the rows are the real solutions of the system formed by setting each expression equal to zero.

`zeros(Expr,[Var])` or `zeros({Expr1, Expr2,…},[{Var1, Var2,…}])`

Example:

`zeros(x^2-4)` returns [-2 2]

Complex Solve

Returns a list of the complex solutions to a polynomial equation or a set of polynomial equations.

`cSolve(Eq,[Var])` or `cSolve({Eq1, Eq2,…}, [Var])`

Example:

`cSolve(x^4-1=0, x)` returns {1 -1 -i i}

Complex Zeros

With an expression as argument, returns a vector containing the complex zeros of the expression; that is, the solutions when the expression is set equal to zero.

With a list of expressions as argument, returns the matrix where the rows are the complex solutions of the system formed by setting each expression equal to zero.

`cZeros(Expr,[Var]` or `cZeros({Expr1, Expr2,…},[{Var1, Var2,…}])`

Example:

`cZeros(x^4-1)` returns [1 -1 -i i]

Numerical Solve

Returns the numerical solution of an equation or a system of equations.

Optionally, you can use a third argument to specify a guess for the solution or an interval within which it is expected that the solution will occur.

Optionally, you can use a fourth argument to name the iterative algorithm to be used by the solver.

If you are solving for a single variable, your options for an iterative algorithm are bisection_solver, newton_solver, and newtonj_solver. If you are solving for two variables, your only option is newton_solver.

`fSolve(Eq,Var)` or `fSolve(Expr, Var=Guess)`

Examples:

`fSolve(cos(x)=x,x,-1..1)` returns [0.739085133215]

fSolve([x^2+y-2,x+y^2-2],[x,y],[0,0]) returns [1.,1.]

Differential Equation

Returns the solution to a differential equation.

`deSolve(Eq,[TimeVar],Var)`

Example:

`desolve(y''+y=0,y)` returns G_0*cos(x)+G_1*sin(x)

ODE Solve

Ordinary Differential Equation solver. Solves an ordinary differential equation given by Expr, with variables declared in VectrVar and initial conditions for those variables declared in VectrInit. For example, odesolve(f(t,y),[t,y],[t0,y0],t1) returns the approximate solution of y'=f(t,y) for the variables t and y with initial conditions t=t0 and y=y0.

`odesolve(Expr,VectVar,VectInitCond,FinalVal,[tstep=Val,curve])`

Example:

`odesolve(sin(t*y),[t,y],[0,1],2)` returns [1.82241255674]

Linear System

Given a vector of linear equations and a corresponding vector of variables, returns the solution to the system of linear equations.

`linsolve([LinEq1, LinEq2,…], [Var1, Var2,…])`

Example:

`linsolve([x+y+z=1,x-y=2,2*x-z=3],[x,y,z])` returns [3/2,-1/2,0]

Rewrite

lncollect

Rewrites an expression with the logarithms collected. Applies ln(a)+n*ln(b) = ln(a*b^n) for an integer n.

`lncollect(Expr)`

Example:

`lncollect(ln(x)+2*ln(y))` returns ln(x*y^2)

powexpand

Rewrites an expression containing a power that is a sum or product as a product of powers. Applies a^(b+c)=(a^b)*(a^c).

`powexpand(Expr)`

Example:

`powexpand(2^(x+y))` yields (2^x)*(2^y)

texpand

Expands a transcendental expression.

`texpand(Expr)`

Example:

`texpand(sin(2*x)+exp(x+y))` returns exp(x)*exp(y)+ 2*cos(x)*sin(x))

Rewrite – Exp & Ln

$e^{y*\ln x}$→ x^y

Returns an expression of the form $e^{n*\ln(x)}$ rewritten as a power of x. Applies $e^{n*\ln(x)}=x^n$.

`exp2pow(Expr)`

Example:

`exp2pow(exp(3*ln(x)))` gives x^3

x^y→$e^{y*\ln x}$

Returns an expression with powers rewritten as an exponential. Essentially the inverse of `exp2pow`.

`pow2exp(Expr)`

Example:

`pow2exp(a^b)` gives exp(b*ln(a))

exp2trig

Returns an expression with complex exponentials rewritten in terms of sine and cosine.

`exp2trig(Expr)`

Example:

`exp2trig(exp(i*x))` gives cos(x)+(i)*sin(x)

expexpand

Returns an expression with exponentials in expanded form.

```
expexpand(Expr)
```

Example:

`expexpand(exp(3*x))` gives exp(x)^3

Rewrite – Sine

asinx→acosx

Returns an expression with asin(x) rewritten as π/2– acos(x).

```
asin2acos(Expr)
```

Example:

`asin2acos(acos(x)+asin(x))` returns π/2

asinx→atanx

Returns an expression with asin(x) rewritten as $\operatorname{atan}\left(\frac{x}{\sqrt{1-x^2}}\right)$:

```
asin2atan(Expr)
```

Example:

`asin2atan(2*asin(x))` returns $2 \cdot \operatorname{atan}\left(\frac{x}{\sqrt{1-x^2}}\right)$

sinx→cosx*tanx

Returns an expression with sin(x) rewritten as cos(x)*tan(x).

```
sin2costan(Expr)
```

Example:

`sin2costan(sin(x))` gives tan(x)*cos(x)

Rewrite – Cosine

acosx→asinx

Returns an expression with acos(x) rewritten as π/2–asin(x).

```
acos2asin(Expr)
```

Example:

`acos2asin(acos(x)+asin(x))` returns π/2

acosx→atanx

Returns an expression with acos(x) rewritten as $\frac{\pi}{2} - \operatorname{atan}\left(\frac{x}{\sqrt{1-x^2}}\right)$:

```
cos2atan(Expr)
```

Example:

`acos2atan(2*acos(x))` gives $2 \cdot \left(\frac{\pi}{2} - \operatorname{atan}\left(\frac{x}{\sqrt{1-x^2}}\right)\right)$

cosx→sinx/tanx

Returns an expression with cos(x) rewritten as sin(x)/tan(x).

```
cos2sintan(Expr)
```

Example:

`cos2sintan(cos(x))` gives sin(x)/tan(x)

Rewrite – Tangent

atanx→asinx

Returns an expression with atan(x) rewritten as $\operatorname{asin}\left(\frac{x}{\sqrt{1-x^2}}\right)$:

```
atan2asin(Expr)
```

Example:

`atan2asin(atan(2*x))` returns $\operatorname{asin}\left(\frac{2 \cdot x}{\sqrt{1-(2 \cdot x)^2}}\right)$

atanx→acosx

Returns an expression with atan(x) rewritten as $\frac{\pi}{2} - \operatorname{acos}\left(\frac{x}{\sqrt{1+x^2}}\right)$:

```
atan2acos(Expr)
```

tanx→sinx/cosx

Returns an expression with tan(x) rewritten as sin(x)/cos(x).

```
tan2sincos(Expr)
```

Example:

`tan2sincos(tan(x))` gives sin(x)/cos(x)

halftan

Returns an expression with sin(x), cos(x) or tan(x) rewritten as tan(x/2).

```
halftan(Expr)
```

Example:

`halftan(sin(x))` returns 2*tan(1/2*x)/(tan(1/2*x)^2+1)

Rewrite – Trig

trigx→sinx

Returns an expression simplified using the formulas sin(x)^2+cos(x)^2=1 and tan(x)=sin(x)/cos(x). Sin(x) is given precedence over cos(x) and tan(x) in the result.

```
trigsin(Expr)
```

Example:

`trigsin(cos(x)^4+sin(x)^2)` returns sin(x)^4-sin(x)^2+1

trigx→cosx

Returns an expression simplified using the formulas sin(x)^2+cos(x)^2=1 and tan(x)=sin(x)/cos(x). Cos(x) is given precedence over sin(x) and tan(x) in the result.

```
trigcos(Expr)
```

Example:

`trigcos(sin(x)^4+sin(x)^2)` returns cos(x)^4-3*cos(x)^2+2

trigx→tanx

Returns an expression simplified using the formulas sin(x)^2+cos(x)^2=1 and tan(x)=sin(x)/cos(x). Tan(x) is given precedence over sin(x) and cos(x) in the result.

```
trigtan(Expr)
```

Example:

`trigtan(cos(x)^4+sin(x)^2)` returns (tan(x)^4+tan(x)^2+1)/(tan(x)^4+2*tan(x)^2+1)

atrig2ln

Returns an expression with inverse trigonometric functions rewritten using the natural logarithm function.

```
trig2ln(Expr)
```

Example:

`atrig2ln(atan(x))` returns $\frac{i}{2}\cdot\ln\frac{(i+x)}{(i-x)}$

tlin

Returns a trigonometric expression with the products and integer powers linearized.

```
tlin(ExprTrig)
```

Example:

`tlin(sin(x)^3)` gives $\frac{3}{4}\cdot\sin(x)-\frac{1}{4}\cdot\sin(3\cdot x)$

tcollect

Returns a trigonometric expression linearized and with any sine and cosine terms of the same angle collected together.

```
tcollect(Expr)
```

Example:

`tcollect(sin(x)+cos(x))` returns

$$\sqrt{2}\cdot\cos\left(x-\frac{1}{4}\cdot\pi\right)$$

trigexpand

Returns a trigonometric expression in expanded form.

`trigexpand(Expr)`

Example:

`trigexpand(sin(3*x))` gives (4*cos(x)^2- 1)*sin(x)

trig2exp

Returns an expression with trigonometric functions rewritten as complex exponentials (without linearization).

`trig2exp(Expr)`

Example:

`trig2exp(sin(x))` returns

$$\frac{-i}{2}\cdot\left(\exp(i\cdot x)-\frac{1}{\exp(i\cdot x)}\right)$$

Integer

Divisors

Returns the list of divisors of an integer or a list of integers.

`idivis(Integer)` or `idivis({Intgr1, Intgr2,…})`

Example:

`idivis(12)` returns [1, 2, 3, 4, 6, 12]

Factors

Returns the prime factor decomposition of an integer.

NOTE: In some cases, `ifactor` may fail. In these cases, it will return the product of -1 and the opposite of the input. The -1 indicates that factorization has failed.

`ifactor(Integer)`

Example:

`ifactor(150)` returns 2*3*5^2

Factor List

Returns a vector containing the prime factors of an integer or a list of integers, with each factor followed by its multiplicity.

`ifactors(Integer)`

or

```
ifactors({Intgr1, Intgr2,…})
```

Example:

`ifactors(150)` returns [2, 1, 3, 1, 5, 2]

GCD

Returns the greatest common divisor of two or more integers.

```
gcd(Intgr1, Intgr2,…)
```

Example:

`gcd(32,120,636)` returns 4

LCM

Returns the least common multiple of two or more integers.

```
lcm(Intgr1, Intgr2,…)
```

Example:

`lcm(6,4)` returns 12

Integer – Prime

Test if Prime

Tests whether or not a given integer is a prime number.

```
isPrime(Integer)
```

Example:

`isPrime(19999)` returns false

Nth Prime

Returns the nth prime number.

`ithprime(Intg(n))` where n is between 1 and 200,000

Example:

`ithprime(5)` returns 11

Next Prime

Returns the next prime or pseudo-prime after an integer.

```
nextprime(Integer)
```

Example:

`nextprime(11)` returns 13

Previous Prime

Returns the prime or pseudo-prime number closest to but smaller than an integer.

`prevprime(Integer)`

Example:

`prevprime(11)` returns 7

Euler

Compute’s Euler's totient for an integer.

`euler(Integer)`

Example:

`euler(6)` returns 2

Integer – Division

Quotient

Returns the integer quotient of the Euclidean division of two integers.

`iquo(Intgr1, Intgr2)`

Example:

`iquo(63, 23)` returns 2

Remainder

Returns the integer remainder from the Euclidean division of two integers.

`irem(Intgr1, Intgr2)`

Example:

`irem(63, 23)` returns 17

a^nMOD p

For the three integers a, n, and p, returns an modulo p in [0, p−1].

`powmod(a, n, p,[Expr],[Var])`

Example:

`powmod(5,2,13)` returns 12

Chinese Remainder

Integer Chinese Remainder Theorem for two equations. Takes two vectors of integers, [a p] and [b q], and returns a vector of two integers, [r n] such that x ≡ r mod n. In this case, x is such that x ≡ a mod p and x ≡ b mod q; also n=p*q.

`ichinrem([a,p],[b,q])`

Example:

`ichinrem([2, 7], [3, 5])` returns [23, 35]

Polynomial

Find Roots

Given a polynomial in x (or a vector containing the coefficients of a polynomial), returns a vector containing its roots.

`proot(Poly)` or `proot(Vector)`

Example:

`proot([1,0,-2])` returns [-1.41421356237,1.41421356237]

Coefficients

Given a polynomial in x, returns a vector containing the coefficients. If the polynomial is in a variable other than x, then declare the variable as the second argument. With an integer as the optional third argument, returns the coefficient of the polynomial whose degree matches the integer.

`coeff(Poly, [Var], [Integer])`

Example:

`coeff(x^2-2)` returns [1 0 -2]

`coeff(y^2-2, y, 1)` returns 0

Divisors

Given a polynomial, returns a vector containing the divisors of the polynomial.

`divis(Poly)` or `divis({Poly1, Poly2,…})`

Example:

`divis(x^2-1)` returns [1 -1+x 1+x (-1+x)*(1+x)]

Factor List

Returns a vector containing the prime factors of a polynomial or a list of polynomials, with each factor followed by its multiplicity.

`factors(Poly)` or `factors({Poly1, Poly2,…})`

Example:

`factors(x^4-1)` returns [x-1 1 x+1 1 x^2+1 1]

GCD

Returns the greatest common divisor of two or more polynomials.

`gcd(Poly1,Poly2...)`

Example:

`gcd(x^4-1, x^2-1)` returns x^2-1

LCM

Returns the least common multiple of two or more polynomials.

`lcm(Poly1, Poly2,…)`

Example:

`lcm(x^2-2*x+1,x^3-1)` gives (x-1)*(x^3-1)

Polynomial – Create

Poly to Coef

Given a polynomial, returns a vector containing the coefficients of the polynomial. With a variable as second argument, returns the coefficients of a polynomial with respect to the variable. With a list of variables as the second argument, returns the internal format of the polynomial.

`symb2poly(Expr,[Var])` or `symb2poly(Expr, {Var1, Var2,…})`

Example:

`symb2poly(x*3+2.1)` returns [3 2.1]

Coef to Poly

With one vector as argument, returns a polynomial in x with coefficients (in decreasing order) obtained from the argument vector. With a variable as second argument, returns a similar polynomial in that variable.

`poly2symb(Vector, [Var]))`

Example:

`poly2symb([1,2,3],x)` returns (x+2)*x+3

Roots to Coef

Returns a vector containing the coefficients (in decreasing order) of the univariate polynomial whose roots are specified in the argument vector.

`pcoef(List)`

Example:

`pcoeff({1,0,0,0,1})` returns [1 -2 1 0 0 0]

Roots to Poly

Takes as argument a vector. The vector contains each root or pole of a rational function. Each root or pole is followed by its order, with poles having negative order. Returns the rational function in x that has the roots and poles (with their orders) specified in the argument vector.

`fcoeff(Vector)` where `Vector` has the form `[Root1, Order1, Root2, Order2, …])`

Example:

`fcoeff([1,2,0,1,3,-1])` returns (x-1)^2*x*(x-3)^- 1

Random

Returns a vector of the coefficients of a polynomial of degree `Integer` and where the coefficients are random integers in the range –99 through 99 with uniform distribution or in an interval specified by `Interval`. Use with poly2symbol to create a random polynomial in any variable.

`randpoly(Integer, Interval, [Dist])`, where `Interval` is of the form `Real1..Real2`.

Example:

`randpoly(t, 8, -1..1)` returns a vector of 9 random integers, all of them between –1 and 1.

Minimum

With only a matrix as argument, returns the minimal polynomial in x of a matrix written as a list of its coefficients. With a matrix and a variable as arguments, returns the minimum polynomial of the matrix written in symbolic form with respect to the variable.

```
pmin(Mtrx,[Var])
```

Example:

`pmin([[1,0],[0,1]],x)` gives x-1

Polynomial – Algebra

Quotient

Returns a vector containing the coefficients of the Euclidean quotient of two polynomials. The polynomials may be written as a list of coefficients or in symbolic form.

```
quo(List1, List2, [Var])
```

or

```
quo(Poly1, Poly2, [Var])
```

Example:

`quo({1, 2, 3, 4}, {-1, 2})` returns [-1 -4 -11]

Remainder

Returns a vector containing the coefficients of the remainder of the Euclidean quotient of two polynomials. The polynomials may be written as a list of coefficients or in symbolic form.

```
rem(List1, List2, [Var])
```

or

```
rem(Poly1, Poly2, [Var])
```

Example:

`rem({1, 2, 3, 4}, {-1, 2})` returns [26]

Degree

Returns the degree of a polynomial.

```
degree(Poly)
```

Example:

`degree(x^3+x)` gives 3

Factor by Degree

For a given polynomial in x of degree n, factors out xn and returns the resulting product.

```
factor_xn(Poly)
```

Example:

`factor_xn(x^4-1)` gives x^4*(1-x^-4)

Coef. GCD

Returns the greatest common divisor (GCD) of the coefficients of a polynomial.

```
content(Poly,[Var])
```

Example:

`content(2*x^2+10*x+6)` gives 2

Zero Count

If a and b are real, this returns the number of sign changes in the specified polynomial in the interval [a,b]. If a or b are nonreal, it returns the number of complex roots in the rectangle bounded by a and b. If Var is omitted, it is assumed to be x.

```
sturmab(Poly[,Var],a,b)
```

Example:

`sturmab(x^2*(x^3+2),-2,0)` returns 1

`sturmab(n^3-1,n,-2-i,5+3i)` returns 3

Chinese Remainder

Given two matrices whose two rows each contain the coefficients of polynomials, returns the Chinese remainder of the polynomials, also written as a matrix.

```
chinrem(Matrix1,Matrix2)
```

Example:

`chinrem` $\left(\begin{bmatrix}1&2&0\\1&0&1\end{bmatrix},\begin{bmatrix}1&1&0\\1&1&1\end{bmatrix}\right)$ returns

```
[[2 2 1] [1 1 2 1 1]]
```

Polynomial – Special

Cyclotomic

Returns the list of coefficients of the cyclotomic polynomial of an integer.

```
cyclotomic(Integer)
```

Example:

`cyclotomic(20)` gives [1 0 –1 0 1 0 –1 0 1]

Groebner Basis

Given a vector of polynomials and a vector of variables, returns the Groebner basis of the ideal spanned by the set of polynomials.

```
gbasis([Poly1 Poly2…], [Var1 Var2…])
```

Example:

`gbasis([x^2-y^3,x+y^2],[x,y])` returns [y^4- y^3,x+y^2]

Groebner Remainder

Given a polynomial and both a vector of polynomials and a vector of variables, returns the remainder of the division of the polynomial by the Groebner basis of the vector of polynomials.

```
greduce(Poly1, [Poly2 Poly3 …], [Var1 Var2…])
```

Example:

`greduce(x*y-1,[x^2-y^2,2*x*y-y^2,y^3],[x,y])` returns 1/2*y^2-1

Hermite

Returns the Hermite polynomial of degree n, where n is an integer less than 1556.

```
hermite(Integer)
```

Example:

`hermite(3)` gives 8*x^3-12*x

Lagrange

Given a vector of abscissas and a vector of ordinates, returns the Lagrange polynomial for the points specified in the two vectors. This function can also take a matrix as argument, with the first row containing the abscissas and the second row containing the ordinates.

```
lagrange([X1 X2…], [Y1 Y2…]))
```

or

`lagrange`$\left(\begin{bmatrix} X1 & X2 & \dots \\ Y1 & Y2 & \dots \end{bmatrix}\right)$

Example:

`lagrange([1,3],[0,1])` gives (x-1)/2

Laguerre

Given an integer n, returns the Laguerre polynomial of degree n.

```
laguerre(Integer)
```

Example:

`laguerre(4)` returns 1/24*a^4+(-1/6)*a^3*x+5/ 12*a^3+1/4*a^2*x^2+(-3/2)*a^2*x+35/24*a^2+(-1/6)*a*x^3+7/4*a*x^2+(-13/3)*a*x+25/12*a+1/ 24*x^4+(-2/3)*x^3+3*x^2-4*x+1

Legendre

Given an integer n, returns the Legendre polynomial of degree n.

```
legendre(Integer)
```

Example:

`legendre(4)` returns $35/8 \cdot x^4 + 15/4\, x^2 + 3/8$

Chebyshev Tn

Given an integer n, returns the Tchebyshev polynomial (of the first kind) of degree n.

`tchebyshev1(Integer)`

Example:

`tchebyshev1(3)` gives 4*x^3-3*x

Chebyshev Un

Given an integer n, returns the Tchebyshev polynomial (of the second kind) of degree n.

`tchebyshev2(Integer)`

Example:

`tchebyshev2(3)` gives 8*x^3-4*x

Plot

Function

Used to define a function graph in the Symbolic view of the Geometry app. Plots the graph of an expression written in terms of the independent variable x. Note that the variable is lowercase.

`plotfunc(Expr)`

Example:

`plotfunc(3*sin(x))` draws the graph of y=3*sin(x)

Contour

Used to define a contour graph in the Symbolic view of the Geometry app. Given an expression in x and y, as well as a list of variables and a list of values, plots the contour graph of the surface z=f(x,y). Specifically, plots the contour lines z1, z2, and so on, defined by the list of values. You can also specify step-values for both x and y.

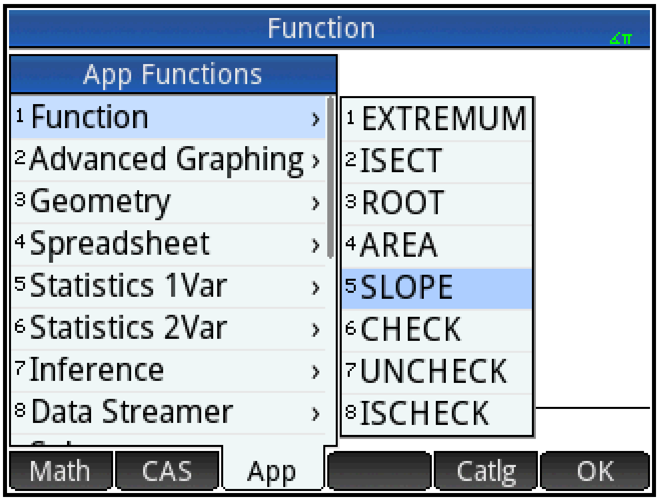
`plotcontour(Expr, [ListVars], [ListVals], [xstep=val1], [ystep=val2])`

Example:

`plotcontour(x^2+2*y^2-2, {x, y}, {2, 4, 6})` draws the three contour lines of z=x^2+2*y^2–2 for z=2, z=4, and z=6.

App menu

Press Mem B to open the Toolbox menus (one of which is the App menu). App functions are used in HP apps to perform common calculations. For example, in the Function app, the Plot view Fcn menu has a function called `SLOPE` that calculates the slope of a given function at a given point. The `SLOPE` function can also be used from the Home view or a program to give the same results. The app functions described in this section are grouped by app.



Function app functions

The Function app functions provide the same functionality found in the Function app's Plot view under the FCN menu. All these operations work on functions. The functions may be expressions in X or the names of the Function app variables F0 through F9.

AREA

Area under a curve or between curves. Finds the signed area under a function or between two functions. Finds the area under the function Fn or below Fn and above the function Fm, from lower X-value to upper X-value.

```
AREA(Fn,[Fm,]lower,upper)
```

Example:

AREA(-X,X^2-2,-2,1) returns 4.5

EXTREMUM

Extremum of a function. Finds the extremum (if one exists) of the function Fn that is closest to the X-value guess.

```
EXTREMUM(Fn, guess)
```

Example:

EXTREMUM$(X)^2$-X-2,0 returns 0.5

ISECT

Intersection of two functions. Finds the intersection (if one exists) of the two functions Fn and Fm that is closest to the Xvalue guess.

```
ISECT(Fn,Fm,guess)
```

Example:

ISECT(X,3-X,2) returns 1.5

ROOT

Root of a function. Finds the root of the function Fn (if one exists) that is closest to the X-value guess.

ROOT(Fn,guess)

Example:

ROOT(3-X^2, 2) returns 1.732...

SLOPE

Slope of a function. Returns the slope of the function Fn at the X-value (if the function's derivative exists at that value).

SLOPE(Fn,value)

Example:

SLOPE(3-X^2,2) returns -4

Solve app functions

The Solve app has a single function that solves a given equation or expression for one of its variables. En may be an equation or expression, or it may be the name of one of the Solve Symbolic variables E0–E9.

SOLVE

Solves an equation for one of its variables. Solves the equation En for the variable var, using the value of guess as the initial value for the value of the variable var. If En is an expression, then the value of the variable var that makes the expression equal to zero is returned.

SOLVE(En,var,guess)

Example:

SOLVE(X^2-X-2,X,3) returns 2

This function also returns an integer that is indicative of the type of solution found, as follows:

0—an exact solution was found

1—an approximate solution was found

2—an extremum was found that is as close to a solution as possible

3—neither a solution, an approximation, nor an extremum was found

Spreadsheet app functions

The spreadsheet app functions can be selected from the App Toolbox menu: press Mem B, tap App and select **Spreadsheet**. They can also be selected from the View menu (View Copy) when the Spreadsheet app is open.

The syntax for many, but not all, the spreadsheet functions follows this pattern:

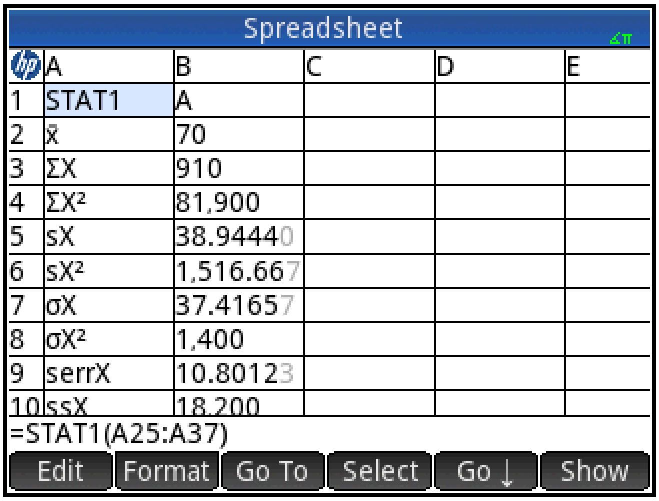
functionName(input,[optional parameters])

Input is the input list for the function. This can be a cell range reference, a simple list or anything that results in a list of values.

One useful optional parameter is `Configuration`. This is a string that controls which values are output. Leaving the parameter out produces the default output. The order of the values can also be controlled by the order that they appear in the string.

For example: `=STAT1(A25:A37)` produces the following default output, based on the numerical values in cells A25 through A37.

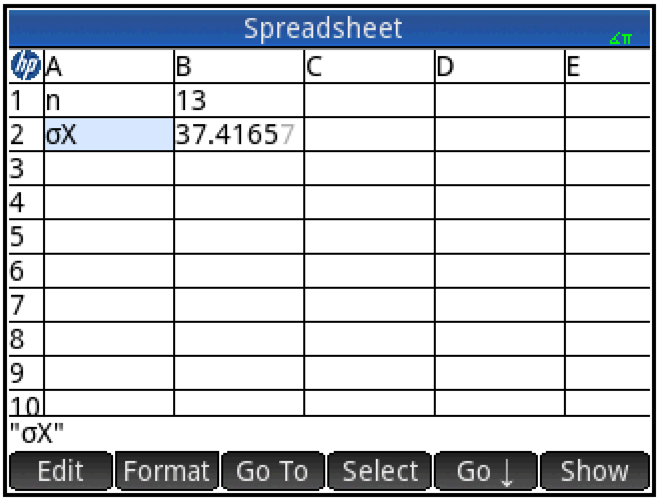
However, if you just wanted to see the number of data points and the standard deviation, you would enter `=STAT1(A25:A37,"h n σ")`. What the configuration string is indicating here is that row headings are required (h), and just the number of data-points (n) and the standard deviation (σ) will be displayed.



Spreadsheet

	A	B	C	D	E
1	STAT1	A			
2	x̄	70			
3	ΣX	910			
4	ΣX²	81,900			
5	sX	38.94440			
6	sX²	1,516.667			
7	σX	37.41657			
8	σX²	1,400			
9	serrX	10.80123			
10	ssX	18,200			

=STAT1(A25:A37)



Spreadsheet

	A	B	C	D	E
1	n	13			
2	σX	37.41657			
3					
4					
5					
6					
7					
8					
9					
10					

"σX"

SUM

Calculates the sum of a range of numbers.

```
SUM([input])
```

For example, `SUM(B7:B23)` returns the sum of the numbers in the range B7 to B23. You can also specify a block of cells, as in `SUM(B7:C23)`.

An error is returned if a cell in the specified range contains a non-numeric object.

AVERAGE

Calculates the arithmetic mean of a range of numbers.

```
AVERAGE([input])
```

For example, AVERAGE(B7:B23) returns the arithmetic mean of the numbers in the range B7 to B23. You can also specify a block of cells, as in AVERAG(B7:C23).

An error is returned if a cell in the specified range contains a non-numeric object.

AMORT

Amortization. Calculates the principal, interest, and balance of a loan over a specified period. Corresponds to pressing Amort in the Finance app.

```
AMORT(Range, NbPmt, IPYR, PV, PMTV[, PPYR=12, CPYR=PPYR, GSize=PPYR,
BEG=0, fix=current], "configuration"])
```

Range: the cell range where the results are to be placed. If only one cell is specified, then the range is automatically calculated starting from that cell.

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

h – show row headers

H – show column headers

S – show the start of the period

E – show the end of the period

P – show the principal paid this period

B – show the balance at the end of the period

I – show the interest paid this period

All the other input parameters (except fix) are Finance app Numeric view variables; see Finance app functions on page 410 for details. Note that only the first four are required. fix is the number of decimal places to be used in the displayed results.

STAT1

The STAT1 function provides a range of one-variable statistics based on lists of data.

```
STAT1(input_range, [mode], [configuration])
```

Input_range is the data source, such as A1:D8.

Mode defines how the input is treated.

The valid values for mode are as follows:

1 = Single data. Each column is treated as an independent dataset.

2 = Frequency data. Columns are used in pairs and the second column is treated as the frequency of appearance of the first column.

If more than one column is specified, they are each treated as a different input data set. If only one row is selected, it is treated as 1 data set. If two columns are selected, the mode defaults to frequency.

Configuration defines which values are in which row and whether the set has row or column headers. The values in the spreadsheet appear in the order the symbol for each value was entered.

The valid values for configuration are as follows:

- H (inserts column headers)
- h (inserts row headers)
- MeanX
- Σ
- Σ^2
- s
- s^2
- σ
- σ^2
- serr
- ss
- n
- min
- q1
- med
- q3
- max

For example, if you specify "h n σ", in the resulting spreadsheet, the first column contains row headers, the first row is the number of items in the input data, and the second row is the standard deviation of the population.

Examples:

```
STAT1(A25:A37)
```

```
STAT1(A25:A37,"h n σ")
```

STAT2

The STAT2 function provides a range of two-variable statistics.

```
STAT2(input_range, [mode], [configuration])
```

Input_range is the data source, such as A1:D8.

Mode defines how the input is treated.

The valid values for Mode are as follows:

1 = Single data. Each pair of columns is treated as an independent dataset.

2 = Frequency data. Columns are used in groups of three, and the third column is treated as the frequency of appearance of the paired columns.

If more than two column is specified, each pair is treated as a different input data set. If only one pair of columns is selected, it is treated as 1 data set. If three columns are selected, the mode defaults to frequency.

Configuration defines which values are in which row and whether the set has row or column headers. The values in the spreadsheet appear in the order the symbol for each value was entered.

The valid values for configuration are as follows:

- H (inserts column headers)
- h (inserts row headers)
- MeanX
- Σx
- Σx^2
- sx
- sx^2
- σx
- σx^2
- serrx
- ssx
- n
- $\bar{y}$
- Σy
- Σy^2
- sy
- sy^2
- σy
- σy^2
- serry
- ssy
- Σxy

For example, if you specify "h n σy", in the resulting spreadsheet, the first column contains row headers, the first row is the number of items in the input data, and the second row is the standard deviation of y.

Examples:

```
STAT2(A25:B37)
```

```
STAT2(A25:B37,"h n σy")
```

REGRS

Attempts to fit the input data to a specified function (default is linear).

- Input range: specifies the data source; for example A1:D8. It must contain an even number of columns. Each pair will be treated as a distinct set of data points.
- model: specifies the model to be used for the regression:

  1 y= sl*x+int

  2 y= sl*ln(x)+int

  3 y= int*exp(sl*x)

  4 y= int*x^sl

  5 y= int*sl^x

  6 y= sl/x+int

  7 y= L/(1 + a*exp(b*x))

  8 y= a*sin(b*x+c)+d

  9 y= cx^2+bx+a

  10 y= dx^3+cx^2+bx+a

  11 y= ex^4+dx^3+cx^2+bx+a
- Configuration: a string which indicates which values you want to place in which row and if you want row and columns headers. Place each parameter in the order that you want to see them appear in the spreadsheet. (If you do not provide a configuration string, a default one will be provided.) The valid parameters are:
  - H (Place column headers)
  - h (Place row headers)
  - sl (slope, only valid for models 1–6)
  - int (intercept, only valid for models 1–6)
  - cor (correlation, only valid for models 1–6)
  - cd (Coefficient of determination, only valid for models 1–6, 8–10)
  - sCov (Sample covariance, only valid for models 1–6)
  - pCov (Population covariance, only valid for models 1–6)
  - L (L parameter for model 7)
  - a (a parameter for models 7-–11)
  - b (b parameter for models 7-–11)
  - c (c parameter for models 8–11)
  - d (d parameter for models 8, 10–11)
  - e (e parameter for model 11)
  - py (place 2 cells, one for user input and the other to display the predicted y for the input)
  - px (place 2 cells, one for user input and the other to display the predicted x for the input)

Example: `REGRS(A25:B37,2)`

predY

Returns the predicted Y for a given x.

```
PredY(mode, x, parameters)
```

- `Mode` governs the regression model used:

  1 y= sl*x+int

  2 y= sl*ln(x)+int

  3 y= int*exp(sl*x)

  4 y= int*x^sl

  5 y= int*sl^x

  6 y= sl/x+int

  7 y= L/(1 + a*exp(b*x))

  8 y= a*sin(b*x+c)+d

  9 y= cx^2+bx+a

  10 y= dx^3+cx^2+bx+a

  11 y= ex^4+dx^3+cx^2+bx+a

- `Parameters` is either one argument (a list of the coefficients of the regression line), or the n coefficients one after another.

PredX

Returns the predicted x for a given y.

```
PredX(mode, y, parameters)
```

- `Mode` governs the regression model used:

  1 y= sl*x+int

  2 y= sl*ln(x)+int

  3 y= int*exp(sl*x)

  4 y= int*x^sl

  5 y= int*sl^x

  6 y= sl/x+int

  7 y= L/(1 + a*exp(b*x))

  8 y= a*sin(b*x+c)+d

  9 y= cx^2+bx+a

  10 y= dx^3+cx^2+bx+a

  11 y= ex^4+dx^3+cx^2+bx+a

- `Parameters` is either one argument (a list of the coefficients of the regression line), or the n coefficients one after another.

HypZ1mean

The one-sample Z-test for a mean.

HypZ1mean($\bar{x}$,n,μ_0,σ,α,mode,[”configuration”])

The input parameters can be a range reference, a list of cell references, or a simple list of values.

Mode: Specifies which alternative hypothesis to use:

- 1: $\mu < \mu_0$
- 2: $\mu > \mu_0$
- 3: $\mu \neq \mu_0$

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- acc: the test result, 0 or 1 to reject or fail to reject the null hypothesis
- tZ: the test Z-value
- tM: the input $\bar{x}$value
- prob: the lower-tail probability
- cZ: the critical Z-value associated with the input α-level
- cx1: the lower critical value of the mean associated with the critical Z-value

- cx2: the upper critical value of the mean associated with the critical Z-value
- std: the standard deviation

Example:

`HypZ1mean(0.461368, 50, 0.5, 0.2887, 0.05, 1, "")` returns two columns into the Spreadsheet app. The first column contains the headers and the second column contains the values for each of the following: Reject/Fail=1, Test Z = -0.94621, Test $\bar{x}$ = 0.461368, P= 0.172022, Critical Z= -1.64485, Critical $\bar{x}$ = 0.432843.

HYPZ2mean

The two-sample Z-test for the difference of two means.

HypZ2mean($_{1}$,$_{2}$, n$_1$,n$_2$,σ$_1$,σ$_2$,α,mode,[”configuration”])

Mode: Specifies which alternative hypothesis to use:

- 1: $\mu_1 < \mu_2$
- 2: $\mu_1 > \mu_2$
- 3: $\mu_1 \neq \mu_2$

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- acc: the test result, 0 or 1 to reject or fail to reject the null hypothesis
- tZ: the test Z-value
- tM: the input $\Delta\bar{x}$value
- prob: the lower-tail probability
- cZ: the critical Z-value associated with the input α-level
- cx1: the lower critical value of $\Delta\bar{x}$ associated with the critical Z-value
- cx2: the upper critical value of $\Delta\bar{x}$ associated with the critical Z-value
- std: the standard deviation

Example:

```
HypZ2mean(0.461368, 0.522851, 50, 50, 0.2887, 0.2887, 0.05, 1, "")
```

HypZ1prop

The one-sample Z-test for a proportion.

HypZ1prop(x,n,π$_0$,,α,mode,[”configuration”]) where x is the success count of the sample

Mode: Specifies which alternative hypothesis to use:

- 1: $\pi < \pi_0$
- 2: $\pi > \pi_0$
- 3: $\pi \neq \pi_0$

Configuration: a string that controls what results are shown and the order in which they appear. An empty `string ""` displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- acc: the test result, 0 or 1 to reject or fail to reject the null hypothesis
- tZ: the test Z-value
- tP: the test proportion of successes
- prob: the lower-tail probability
- cZ: the critical Z-value associated with the input α-level
- cp1: the lower critical proportion of successes associated with the critical Z-value
- cp2: the upper critical proportion of successes associated with the critical Z-value
- std: the standard deviation

Example:

```
HypZ1prop(21, 50, 0.5, 0.05,1, "")
```

HypZ2prop

The two-sample Z-test for comparing two proportions.

`HypZ2prop` x_1,x_2,n_1,n_2,, `α,mode,[”configuration”])` where x_1 and x_2 are the success counts of the two samples)

- 1: $\pi_1 < \pi_2$
- 2: $\pi_1 > \pi_2$
- 3: $\pi_1 \neq \pi_2$

Configuration: a string that controls what results are shown and the order in which they appear. An empty `string ""` displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- acc: the test result, 0 or 1 to reject or fail to reject the null hypothesis
- tZ: the test Z-value
- tP: the test Δπ value
- prob: the lower-tail probability
- cZ: the critical Z-value associated with the input α-level
- cp1: the lower critical value of Δπ associated with the critical Z-valueassociated with the critical Z-value
- cp2: the upper critical value of Δπ of associated with the critical Z-value

Example:

```
HypZ2prop(21, 26, 50, 50, 0.05, 1, "")
```

HypT1mean

The one-sample t-test for a mean.

```
HypT1mean(x̄,n,μ0,α,mode,[”configuration”])
```

- 1: $\mu < \mu_0$
- 2: $\mu > \mu_0$
- 3: $\mu \neq \mu_0$

Configuration: a string that controls what results are shown and the order in which they appear. An empty `string ""` displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- acc: the test result, 0 or 1 to reject or fail to reject the null hypothesis
- tT: the test T-value
- tM: the input $\bar{x}$ value
- prob: the lower-tail probability
- df: the degrees of freedom
- cT: the critical T-value associated with the input α-level
- cx1: the lower critical value of the mean associated with the critical T-value
- cx2: the upper critical value of the mean associated with the critical T-value

Example:

```
HypT1mean(0.461368, 0.2776, 50, 0.5, 0.05, 1, "")
```

HypT2mean

The two-sample T-test for the difference of two means.

```
HypT2mean (x̄1,x̄2,n1,n2,s1,s2,α,pooled,mode,[”configuration”]
```

Pooled: Specifies whether or not the samples are pooled

- 0: not pooled
- 1: pooled
- 1: $\mu_1 < \mu_2$
- 2: $\mu_1 > \mu_2$
- 3: $\mu_1 \neq \mu_2$

Configuration: a string that controls what results are shown and the order in which they appear. An empty `string ""` displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- acc: the test result, 0 or 1 to reject or fail to reject the null hypothesis
- tT: the test T-value

- tM: the input $\Delta\bar{x}$ value
- prob: the lower-tail probability
- cT: the critical T-value associated with the input α-level
- cx1: the lower critical value of $\Delta\bar{x}$ associated with the critical T-value
- cx2: the upper critical value of $\Delta\bar{x}$ associated with the critical T-value

Example:

```
HypT2mean(0.461368, 0.522851, 0.2776, 0.2943,50, 50, 0, 0.05, 1, "")
```

ConfZ1mean

The one-sample Normal confidence interval for a mean.

```
ConfZ1mean(x̄,n,s,C,[”configuration”])
```

Configuration: a string that controls what results are shown and the order in which they appear. An empty `string ""` displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- Z: the critical Z-value
- zXl: the lower bound of the confidence interval
- zXh: the upper bound of the confidence interval
- prob: the lower-tail probability
- std: the standard deviation

Example:

```
ConfZ1mean(0.461368, 50, 0.2887, 0.95, "")
```

ConfZ2mean

The two-sample Normal confidence interval for the difference of two means.

```
ConfZ2mean (x̄₁,x̄₂,n₁,n₂,s₁,s₂,C,[”configuration”]
```

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- Z: the critical Z-value
- zXl: the lower bound of the confidence interval
- zXh: the upper bound of the confidence interval
- prob: the lower-tail probability
- std: the standard deviation

Example:

```
ConfZ2mean(0.461368, 0.522851, 50, 50, 0.2887, 0.2887, 0.95, "")
```

ConfZ1prop

The one-sample Normal confidence interval for a proportion.

```
ConfZ1prop(x,n,C,[”configuration”])
```

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- Z: the critical Z-value
- zXl: the lower bound of the confidence interval
- zXh: the upper bound of the confidence interval
- zXm: the midpoint of the confidence interval
- std: the standard deviation

Example:

```
ConfZ1prop(21, 50, 0.95, "")
```

ConfZ2prop

The two-sample Normal confidence interval for the difference of two proportions.

ConfZ2prop(x_1,x_2,n_1,n_2,C,[”configuration”])

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- Z: the critical Z-value
- zXl: the lower bound of the confidence interval
- zXh: the upper bound of the confidence interval
- zXm: the midpoint of the confidence interval
- std: the standard deviation

Example:

```
ConfZ2prop(21, 26, 50, 50, 0.95, "")
```

ConfT1mean

The one-sample Student’s T confidence interval for a mean.

ConfT1mean($\bar{x}$,s,n,C,[”configuration”])

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- DF: the degrees of freedom

- T: the critical T-value
- tXl: the lower bound of the confidence interval
- tXh: the upper bound of the confidence interval
- std: the standard deviation

Example:

```
ConfT1mean(0.461368, 0.2776, 50, 0.95, "")
```

ConfT2mean

The two-sample Student’s T confidence interval for the difference of two means.

ConfT2mean ($\bar{x}_1$,$\bar{x}_2$,n_1,n_2,s_1,s_2,C,pooled,[”configuration”]

Configuration: a string that controls what results are shown and the order in which they appear. An empty string "" displays the default: all results, including headers. The options in the configuration string are separated by spaces.

- h: header cells will be created
- DF: the degrees of freedom
- T: the critical T-value
- tXl: the lower bound of the confidence interval
- tXh: the upper bound of the confidence interval
- tXm: the midpoint of the confidence interval
- std: the standard deviation

Example:

```
ConfT2mean(0.461368, 0.522851, 0.2776, 0.2943, 50, 50, 0, 0.95, "")
```

Statistics 1Var app functions

The Statistics 1Var app has three functions designed to work together to calculate summary statistics based on one of the statistical analyses (H1–H5) defined in the Symbolic view of the Statistics 1Var app.

Do1VStats

Do1-variable statistics. Performs the same calculations as tapping Stats in the Numeric view of the Statistics 1Var app and stores the results in the appropriate Statistics 1Var app results variables. Hn must be one of the Statistics 1Var app Symbolic view variables H1–H5.

```
Do1VStats(Hn)
```

Example:

Do1VStats(H1) executes summary statistics for the currently defined H1 analysis.

SetFreq

Set frequency. Sets the frequency for one of the statistical analyses (H1–H5) defined in the Symbolic view of the Statistics 1Var app. The frequency can be either one of the columns D0–D9, or any positive integer. Hn must be one of the Statistics 1Var app Symbolic view variables `H1–H5`. If used, Dn must be one of the column variables `D0–D9`; otherwise, value must be a positive integer.

`SetFreq(Hn,Dn)` or `SetFreq(Hn,value)`

Example:

`SetFreq(H2,D3)` sets the **Frequency** field for the H2 analysis to use the list D3.

SetSample

Set sample data. Sets the sample data for one of the statistical analyses (`H1–H5`) defined in the Symbolic view of the Statistics 1Var app. Sets the data column to one of the column variables `D0–D9` for one of the statistical analyses `H1–H5`.

`SetSample(Hn,Dn)`

Example:

`SetSample(H2,D2)` sets the Independent **Column** field for the H2 analysis to use the data in the list D2.

Statistics 2Var app functions

The Statistics 2Var app has a number of functions. Some are designed to calculate summary statistics based on one of the statistical analyses (`S1–S5`) defined in the Symbolic view of the Statistics 2Var app. Others predict X- and Y-values based on the fit specified in one of the analyses.

PredX

Predict X. Uses the fit from the first active analysis (`S1–S5`) found to predict an x-value given the y-value.

`PredX(value)`

PredY

Predict Y. Uses the fit from the first active analysis (`S1–S5`) found to predict a y-value given the x-value.

`PredY(value)`

Resid

Residuals. Returns the list of residuals for the given analysis (`S1–S5`), based on the data and a fit defined in the Symbolic view for that analysis.

`Resid(Sn)` or `Resid()`

Resid() looks for the first defined analysis in the Symbolic view (S1-S5).

Do2VStats

Do 2-variable statistics. Performs the same calculations as tapping Stats in the Numeric view of the Statistics 2Var app and stores the results in the appropriate Statistics 2Var app results variables. `Sn` must be one of the Statistics 2Var app Symbolic view variables `S1–S5`.

`Do2VStats(Sn)`

Example:

`Do1VStats(S1)` executes summary statistics for the currently defined S1 analysis.

SetDepend

Set dependent column. Sets the dependent column for one of the statistical analyses `S1–S5` to one of the column variables `C0–C9`.

```
SetDepend(Sn,Cn)
```

Example:

`SetDepend(S1,C3)` sets the Dependent Column field for the `S1` analysis to use the data in list `C3`.

SetIndep

Set independent column. Sets the independent column for one of the statistical analyses `S1–S5` to one of the column variables `C0–C9`.

```
SetIndep(Sn,Cn)
```

Example:

`SetIndep(S1, C2)` sets the **Independent Column** field for the `S1` analysis to use the data in list `C2`.

Inference app functions

The Inference app has a single function that returns the same results as tapping Calc in the Numeric view of the Inference app. The results depend on the contents of the Inference app variables `Method`, `Type`, and `AltHyp`.

DoInference

Calculate confidence interval or test hypothesis. Uses the current settings in the Symbolic and Numeric views to calculate a confidence interval or test an hypothesis. Performs the same calculations as tapping Calc in the Numeric view of the Inference app and stores the results in the appropriate Inference app results variables.

```
DoInference()
```

HypZ1mean

The one-sample Z-test for a mean. Returns a list containing (in order):

- 0 or 1 to reject or fail to reject the null hypothesis
- The test Z-value
- The input $\bar{x}$ value
- The upper-tail probability
- The upper critical Z-value associated with the input α-level
- The critical value of the statistic associated with the critical Z-value

$\mathtt{HypZ1mean}(\bar{x},n,\mu_0,\sigma,\alpha,\mathtt{mode})$

Mode: Specifies which alternative hypothesis to use:

- 1: $\mu < \mu_0$
- 2: $\mu > \mu_0$
- 3: $\mu \neq \mu_0$

Example:

`HypZ1mean(0.461368, 50, 0.5, 0.2887, 0.05, 1)` returns {1, -.9462…, 0.4614, 0.8277…, 1.6448…, 0.5671…}

HypZ2mean

The two-sample Z-test for means. Returns a list containing (in order):

- 0 or 1 to reject or fail to reject the null hypothesis
- The test Z-value
- tZ: the test Z-value
- The test $\Delta\bar{x}$ value
- The upper-tail probability
- The upper critical Z-value associated with the input α-level
- The critical value of $\Delta\bar{x}$ associated with the critical Z-value

`HypZ2mean(`$\bar{x}_1\bar{x}_2,n_1,n_2,\sigma_1,\sigma_2,\alpha$`,mode)`

Mode: Specifies which alternative hypothesis to use:

- 1: $\mu_1 < \mu_2$
- 2: $\mu_1 > \mu_2$
- 3: $\mu_1 \neq \mu_2$

Example:

`HypZ2mean(0.461368, 0.522851, 50, 50, 0.2887, 0.2887, 0.05, 1)` returns {1, -1.0648…, -0.0614…, 0.8565…, 1.6448…, 0.0334…}.

HypZ1prop

The one-proportion Z-test. Returns a list containing (in order):

- 0 or 1 to reject or fail to reject the null hypothesis
- The test Z-value
- The test π value
- The upper-tail probability
- The upper critical Z-value associated with the input α-level
- The critical value of π associated with the critical Z-value

`HypZ1mean(0.461368, 50, 0.5, 0.2887, 0.05, 1)HypZ1prop(x,n,`π_0,α`,mode)`

Mode: Specifies which alternative hypothesis to use:

- 1: $\pi < \pi_0$
- 2: $\pi > \pi_0$
- 3: $\pi \neq \pi_0$

Example:

`HypZ1prop(21, 50, 0.5, 0.05,1)` returns {1, -1.1313..., 0.42, 0.8710..., 1.6448..., 0.6148...}

HypZ2prop

The two-sample Z-test for proportions. Returns a list containing (in order):

- 0 or 1 to reject or fail to reject the null hypothesis
- The test Z-value
- The test Z-value
- The test Δπ value
- The upper-tail probability
- The upper critical Z-value associated with the input α-level
- The critical value of Δπ associated with the critical Z-value

`HypZ2prop(`$\bar{x}_1$,$\bar{x}_2$,n_1,n_2,α`, mode)`

Mode: Specifies which alternative hypothesis to use:

- 1: $\pi_1 < \pi_2$
- 2: $\pi_1 > \pi_2$
- 3: $\pi_1 \neq \pi_2$

Example:

`HypZ2prop(21, 26, 50, 50, 0.05, 1)` returns {1, -1.0018..., -0.1, 0.8417..., 1.6448..., 0.0633...}

HypT1mean

The one-sample t-test for a mean. Returns a list containing (in order):

- 0 or 1 to reject or fail to reject the null hypothesis
- The test T-value
- The input $\bar{x}$ value
- The upper-tail probability
- The degrees of freedom
- The upper critical T-value associated with the input α-level
- The critical value of the statistic associated with the critical t-value

`HypT1mean(`$\bar{x}$,s,n,μ_0,α`, mode)`

Mode: Specifies which alternative hypothesis to use:

- 1: $\mu < \mu_0$
- 2: $\mu > \mu_0$
- 3: $\mu \neq \mu_0$

Example:

`HypT1mean(0.461368, 0.2776, 50, 0.5, 0.05, 1)` returns {1, -.9462..., 0.4614, 0.8277..., 1.6448..., 0.5671...}

HypT2mean

The two-sample T-test for means. Returns a list containing (in order):

- 0 or 1 to reject or fail to reject the null hypothesis
- The test T-value
- The test $\Delta\bar{x}$ value
- The upper-tail probability
- The degrees of freedom
- The upper critical T-value associated with the input α-level
- The critical value of $\Delta\bar{x}$ associated with the critical t-value

`HypT2mean((x̄₁,x̄₂,s₁,s₂,n₁,n₂,α,pooled,mode)`

Pooled: Specifies whether or not the samples are pooled

- 0: not pooled
- 1: pooled

Mode: Specifies which alternative hypothesis to use:

- 1: $\mu_1 < \mu_2$
- 2: $\mu_1 > \mu_2$
- 3: $\mu_1 \neq \mu_2$

Example:

`HypT2mean(0.461368, 0.522851, 0.2776, 0.2943,50, 50, 0.05, 0, 1)` returns {1, -1.0746..., -0.0614..., 0.8574..., 97.6674..., 1.6606..., 0.0335...}

ConfZ1mean

The one-sample Normal confidence interval for a mean. Returns a list containing (in order):

- The lower critical Z-value
- The lower bound of the confidence interval
- The upper bound of the confidence interval

`ConfZ1mean(x̄,n,σ,C)`

Example:

`ConfZ1mean(0.461368, 50, 0.2887, 0.95)` returns {- 1.9599..., 0.3813..., 0.5413...}

ConfZ2mean

The two-sample Normal confidence interval for the difference of two means. Returns a list containing (in order):

- The lower critical Z-value
- The lower bound of the confidence interval
- The upper bound of the confidence interval

ConfZ2mean($\bar{x}_1,\bar{x}_2,n_1,n_2,\sigma_1,\sigma_2$,C)

Example:

`ConfZ2mean(0.461368, 0.522851, 50, 50, 0.2887, 0.2887, 0.95)` returns {-1.9599…, -0.1746…, 0.0516…)}

ConfZ1prop

The one-sample Normal confidence interval for a proportion. Returns a list containing (in order):

- The lower critical Z-value
- The lower bound of the confidence interval
- The upper bound of the confidence interval

```
ConfZ1prop(x,n,C)
```

Example:

`ConfZ1prop(21, 50, 0.95)` returns {-1.9599…, 0.2831…, 0.5568…}

ConfZ2prop

The two-sample Normal confidence interval for the difference of two proportions. Returns a list containing (in order):

- The lower critical Z-value
- The lower bound of the confidence interval
- The upper bound of the confidence interval

ConfZ2prop($\bar{x}_1,\bar{x}_2,n_1,n_2$,C)

Example:

`ConfZ2prop(21, 26, 50, 50, 0.95)` returns {-1.9599…, -0.2946…, 0.0946…)}

ConfT1mean

The one-sample Student’s T confidence interval for a mean. Returns a list containing (in order):

- The degrees of freedom
- The lower bound of the confidence interval
- The upper bound of the confidence interval

ConfT1mean($\bar{x}$,s,n,C)

Example:

`ConfT1mean(0.461368, 0.2776, 50, 0.95)` returns {49, -.2009..., 0.5402...}

ConfT2mean

The two-sample Student's T confidence interval for the difference of two means. Returns a list containing (in order):

- The degrees of freedom
- The lower bound of the confidence interval
- The upper bound of the confidence interval

`ConfT2mean(`$\bar{x}_1,\bar{x}_2,s_1,s_2,n_1,n_2$`,pooled,C)`

Example:

`ConfT2mean(0.461368, 0.522851, 0.2887, 0.2887, 50, 50, 0.95,0)` returns {98.0000..., -1.9844, - 0.1760..., 0.0531...)}

Chi2GOF

Chi-square goodness of fit test. Takes as arguments a list of observed count data, a second list, and a value of 0 or 1. If value=0, the second list is taken as a list of expected probabilities. If value=1, then the second list is taken as a list of expected counts. Returns a list containing the chi-square statistic value, the probability, and the degrees of freedom.

`Chi2GOF(List1, List2, Value)`

Example:

`Chi2GOF({10,10,12,15,10,6},{.24,.2,.16,.14,.1 3,.13},0)` returns {10.1799..., 0.07029..., 5}

Chi2TwoWay

Chi-square two-way test. Given a matrix of count data, returns a list containing the chi-square statistic value, the probability, and the degrees of freedom.

`Chi2TwoWay(Matrix)`

Example:

`Chi2TwoWay([[30,35,30],[11,2,19],[43,35,35]])` returns {14.4302..., 0.0060..., 4}

LinRegrTConf- Slope

The linear regression confidence interval for the slope. Given a list of explanatory variable data (X), a list of response variable data (Y), and a confidence level, returns a list containing the following values in the order shown:

- C: the given confidence level
- Critical T: the value of t associated with the given confidence level
- DF: the degrees of freedom
- β_1: the slope of the linear regression equation
- serrSlope: the standard error of the slope

- Lower: the lower bound of the confidence interval for the slope
- Upper: the upper bound of the confidence interval for the slope

```
LinRegrTConfSlope(List1, List2, C-value)
```

Example:

`LinRegrTConfSlope({1,2,3,4},{3,2,0,-2},0.95)` returns {0.95, 4.302…, 2, -1.7, 0.1732…, -2.445…, -0.954…}

LinRegrTConfInt

The linear regression confidence interval for the intercept. Given a list of explanatory variable data (X), a list of response variable data (Y), and a confidence level, returns a list containing the following values in the order shown:

- C: the given confidence level
- Critical T: the value of t associated with the given confidence level
- DF: the degrees of freedom
- β_0: the intercept of the linear regression equation
- serrInter: the standard error of the intercept
- Lower: the lower bound of the confidence interval for the intercept
- Upper: the upper bound of the confidence interval for the intercept

```
LinRegrTConfInt(List1, List2, C-value)
```

Example:

`LinRegrTConfInt({1, 2, 3, 4}, {3, 2, 0, - 2},0.95)` returns {0.95, 4.302…, 2, 5, 0.474…, 2.959…, 7.040…}

LinRegrTMean-Resp

The linear regression confidence interval for a mean response. Given a list of explanatory variable data (X), a list of response variable data (Y), an X-value, and a confidence level, returns a list containing the following values in the order shown:

- X: the given X-value
- C: the given confidence level
- DF: the degrees of freedom
- $\hat{Y}$: the mean response for the given X-value
- serr $\hat{Y}$: the standard error of the mean response
- serrInter: the standard error of the intercept
- Lower: the lower bound of the confidence interval for the mean response
- Upper: the upper bound of the confidence interval for the mean response

```
LinRegrTMeanResp(List1, List2, X-value, Cvalue)
```

Example:

`LinRegrTMeanResp({1, 2, 3, 4}, {3, 2, 0, -2}, 2.5, 0.95)` returns {2.5, 0.95, 4.302…, 2, 0.75, 0.193…, −0.083, 1.583…}

LinRegrTPredInt

The linear regression prediction interval for a future response. Given a list of explanatory variable data (X), a list of response variable data (Y), a future X-value, and a confidence level, returns a list containing the following values in the order shown:

- X: the given future X-value
- C: the given confidence level
- DF: the degrees of freedom
- Ŷ: the mean response for the given future X-value
- serr Ŷ: the standard error of the mean response
- serrInter: the standard error of the intercept
- Lower: the lower bound of the prediction interval for the mean response
- Upper: the upper bound of the prediction interval for the mean response

```
LinRegrTPredInt(List1, List2, X-value, Cvalue)
```

Example:

`LinRegrTPredInt({1, 2, 3, 4}, {3, 2, 0, -2}, 2.5, 0.95)` returns {2.5, 0.95, 4.302…, 2, 0.75, 0.433…, −1.113…, 2.613…}

LinRegrTTest

The linear regression t-test. Given a list of explanatory variable data (X), a list of response variable data (Y), and a value for AltHyp, returns a list containing the following values in the order shown:

- T: the t-value
- P: the probability associated with the t-value
- DF: the degrees of freedom
- β_0: the y-intercept of the regression line
- β_1: the slope of the regression line
- serrLine: the standard error of the regression line
- serr Ŷ: the standard error of the mean response
- serrSlope: the standard error of the slope
- serrInter: the standard error of the y-intercept
- r: the correlation coefficient
- R^2: the coefficient of determination

The values for AltHyp are as follows:

- AltHyp=0 for $\mu<\mu_0$
- AltHyp=1 for $\mu>\mu_0$
- AltHyp=2 for $\mu\neq\mu_0$

Example:

`LinRegrTTest({1,2,3,4}, {3,2,0,-2}, 0)` returns {−9.814…, 2, 5, −1.7, 0.387…, 0.173…, 0.474…, −0.989…, 0.979…}

Finance app functions

The Finance app uses a set of functions that all reference the same set of Finance app variables. These correspond to the fields in the Finance app Numeric view. There are 5 main TVM variables, 4 of which are mandatory for each of these functions, as they each solve for and return the value of the fifth variable to two decimal places. `DoFinance` is the sole exception to this syntax rule. Note that money paid to you is entered as a positive number and money you pay to others as part of a cash flow is entered as a negative number. There are 3 other variables that are optional and have default values. These variables occur as arguments to the Finance app functions in the following set order:

- `NbPmt`—the number of payments
- `IPYR`—the annual interest rate
- `PV`—the present value of the investment or loan
- `PMTV`—the payment value
- `FV`—the future value of the investment or loan
- `PPYR`—the number of payments per year (12 by default)
- `CPYR`—the number of compounding periods per year (12 by default)
- `BEG`—payments made at the beginning or end of the period; the default is BEG=0, meaning that payments are made at the end of each period

The arguments `PPYR`, `CPYR`, and `BEG` are optional; if not supplied, `PPYR`=12, `CPYR`=`PPYR`, and `BEG`=0.

CalcFV

Solves for the future value of an investment or loan.

```
CalcFV(NbPmt,IPYR,PV,PMTV[,PPYR,CPYR,BEG]
```

Example:

`CalcFV(360, 6.5, 150000, -948.10)` returns -2.25

CalcIPYR

Solves for the interest rate per year of an investment or loan.

```
CalcIPYR(NbPmt,PV,PMTV,FV[,PPYR,CPYR, BEG])
```

Example:

`CalcIPYR(360, 150000, -948.10, -2.25)` returns 6.50

CalcNbPmt

Solves for the number of payments in an investment or loan.

```
CalcNbPmt(IPYR,PV,PMTV,FV[,PPYR,CPYR,BEG])
```

Example:

`CalcNbPmt(6.5, 150000, -948.10, -2.25)` returns 360.00

CalcPMT

Solves for the value of a payment for an investment or loan.

```
CalcPMT(NbPmt,IPYR,PV,FV[,PPYR,CPYR,BEG])
```

Example:

`CalcPMT(360, 6.5, 150000, -2.25)` returns -948.10

CalcPV

Solves for the present value of an investment or loan.

```
CalcPV(NbPmt,IPYR,PMTV,FV[,PPYR,CPYR,BEG])
```

Example:

`CalcPV(360, 6.5, -948.10, -2.25)` returns 150000.00

DoFinance

Calculate TVM results. Solves a TVM problem for the variable TVMVar. The variable must be one of the Finance app's Numeric view variables. Performs the same calculation as tapping Solve in the Numeric view of the Finance app with TVMVar highlighted.

```
DoFinance(TVMVar)
```

Example:

`DoFinance(FV)` returns the future value of an investment in the same way as tapping Solve in the Numeric view of the Finance app with `FV` highlighted.

Linear Solver app functions

The Linear Solver app has 3 functions that offer the user flexibility in solving 2x2 or 3x3 linear systems of equations.

Solve2x2

Solves a 2x2 linear system of equations.

```
Solve2x2(a, b, c, d, e, f)
```

Solves the linear system represented by:

$ax+by=c$

$dx+ey=f$

Solve3x3

Solves a 3x3 linear system of equations.

```
Solve3x3(a, b, c, d, e, f, g, h, i, j, k, l)
```

Solves the linear system represented by:

ax+by+cz=d

ex+fy+gz=h

ix+jy+kz=l

LinSolve

Solve linear system. Solves the 2x2 or 3x3 linear system represented by matrix.

```
LinSolve(matrix)
```

Example:

`LinSolve([[A, B, C], [D, E,F]])` solves the linear system:

ax+by=c

dx+ey=f

Triangle Solver app functions

The Triangle Solver app has a group of functions which allow you to solve a complete triangle from the input of three consecutive parts of the triangle (one of which must be a side length). The names of these commands use A to signify an angle and S to signify a side length. To use these commands, enter three inputs in the specified order given by the command name. These commands all return a list of the three unknown values (lengths of sides and/or measures of angles).

AAS

Angle-Angle-Side. Takes as arguments the measures of two angles and the length of the side opposite the first angle and returns a list containing the length of the side opposite the second angle, the length of the third side, and the measure of the third angle (in that order).

```
AAS(angle,angle,side)
```

Example:

`AAS(30, 60, 1)` in degree mode returns {1.732..., 2, 90}

ASA

Angle-Side-Angle. Takes as arguments the measure of two angles and the length of the included side and returns a list containing the length of the side opposite the first angle, the length of the side opposite the second angle, and the measure of the third angle (in that order).

```
ASA(angle,side,angle)
```

Example:

`ASA(30, 2, 60)` in degree mode returns {1, 1.732..., 90}

SAS

Side-Angle-Side. Takes as arguments the length of two sides and the measure of the included angle and returns a list containing the length of the third side, the measure of the angle opposite the third side and the measure of the angle opposite the second side.

```
SAS(side,angle,side)
```

Example:

`SAS(2, 60, 1)` in degree mode returns {1.732..., 30, 90}

SSA

Side-Side-Angle. Takes as arguments the lengths of two sides and the measure of a non-included angle and returns a list containing the length of the third side, the measure of the angle opposite the second side, and the measure of the angle opposite the third side. Note: In an ambiguous case, this command will only give you one of the two possible solutions.

```
SSA(side,side,angle)
```

Example:

`SSA(1, 2, 30)` returns {1.732..., 90, 60}

SSS

Side-Side-Side Takes as arguments the lengths of the three sides of a triangle and returns the measures of the angles opposite them, in order.

```
SSS(side,side,side)
```

Example:

`SSS(3, 4, 5)` in degree mode returns {36.8..., 53.1..., 90}

DoSolve

Solves the current problem in the Triangle Solver app. The Triangle Solver app must have enough data entered to ensure a successful solution; that is, there must be at least three values entered, one of which must be a side length. Returns a list containing the unknown values in the Numeric view, in their order of appearance in that view (left to right and top to bottom).

```
DoSolve()
```

Linear Explorer functions

SolveForSlope

Solve for slope. Takes as input the coordinates of two points (x_1, y_1) and (x_2, y_2) and returns the slope of the line containing those two points.

`SolveForSlope(`x_1,y_1,x_2,y_2`)`

Example:

`SolveForSlope(3,4,2,2)` returns 2

SolveForYIntercept

Solve for y-intercept. Takes as input the coordinates of a point (x, y), and a slope m, and returns the y-intercept of the line with the given slope that contains the given point.

```
SolveForYIntercept(x, y, m)
```

Example:

`SolveForYIntercept(2,3,-1)` returns 5

Quadratic Explorer functions

SOLVE

Solve quadratic. Given the coefficients of a quadratic equation $ax^2+bx+c=0$, returns the real solutions.

```
SOLVE(a, b, c)
```

Example:

`SOLVE(1,0,-4)` returns {-2, 2}

DELTA

Discriminant. Given the coefficients of a quadratic equation $ax^2+bx+c=0$, returns the value of the discriminant in the Quadratic Formula.

```
DELTA(a, b, c)
```

Example:

`DELTA(1,0,-4)` returns 16

Common app functions

In addition to the app functions specific to each app, there are three functions common to the following apps. These use as an argument an integer from 0 to 9, which corresponds to one of the Symbolic view variables for that app.

- Function (F0–F9)
- Solve (E0–E9)
- Statistics 1Var (H1–H5)
- Statistics 2Var (S1–S5)
- Parametric (X0/Y0–X9/Y9)
- Polar (R0–R9)
- Sequence (U0–U9)
- Advanced Graphing (V0–V9)

CHECK

Check. Checks—that is, selects—the Symbolic view variable corresponding to `Digit`. Used primarily in programming to activate Symbolic view definitions in apps.

```
CHECK(Digit)
```

Example:

With the Function app as the current app, CHECK(1) checks the Function app Symbolic view variable F1. The result is that `F1(X)` is drawn in the Plot view and has a column of function values in the Numeric view of the Function app. With another app as the current app, you would have to enter `Function.CHECK(1)`.

UNCHECK

Un-Check. Un-checks—that is, deselects—the Symbolic view variable corresponding to `Digit`. Used primarily in programming to de-activate symbolic view definitions in apps.

```
UNCHECK(Digit)
```

Example:

With the Sequence app as the current app, `UNCHECK(2)` unchecks the Sequence app Symbolic view variable `U2`. The result is that `U2(N)` is no longer drawn in Plot view and has no column of values in the Numeric view of the Sequence app. With another app as the current app, you would have to enter `Sequence.UNCHECK(2)`.

ISCHECK

Test for check. Tests whether a Symbolic view variable is checked. Returns 1 if the variable is checked and 0 if it is not checked.

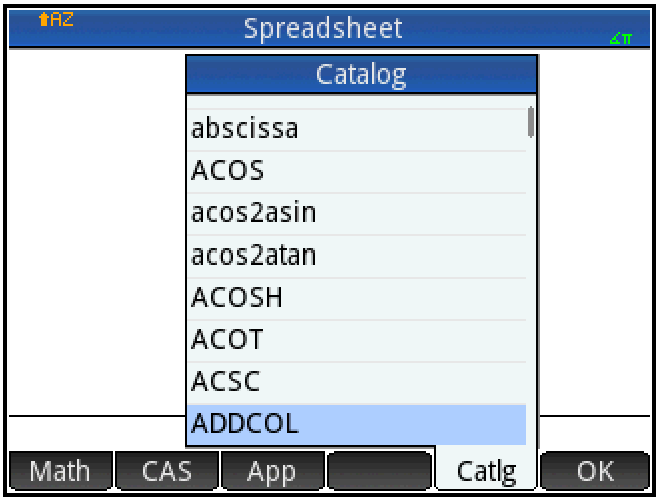
```
ISCHECK(Digit)
```

Example:

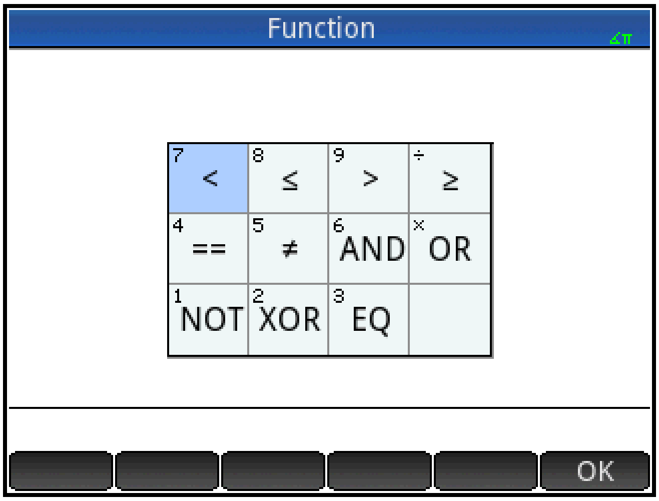
With the Function app as the current app, `ISCHECK(3)` checks to see if `F3 (X)` is checked in the Symbolic view of the Function app.

Ctlg menu

The Catlg menu brings together all the functions and commands available on the HP Prime. However, this section describes the functions and commands that can only be found on the Catlg menu. The functions and commands that are also on the Math menu are described in Keyboard functions on page 347. Those that are also on the CAS menu are described in CAS menu on page 364.



Some of the options on the Catlg menu can also be chosen from the relations palette (Shift 9)



!

Factorial. Returns the factorial of a positive integer. For nonintegers, ! = Γ(x + 1). This calculates the Gamma function.

```
value!
```

Example:

`6!` returns 720

%

x percent of y. Returns (x/100)*y.

```
%(x, y)
```

Example:

`%(20,50)` returns 10

%TOTAL

Percent total; the percentage of x that is y. Returns 100*y/x.

```
%TOTAL(x, y)
```

Example:

`%TOTAL(20,50)` returns 250

(

Inserts opening parenthesis.

*

Multiplication symbol. Returns the product of two numbers or the scalar product of two vectors.

+

Addition symbol. Returns the sum of two numbers, the term-by-term sum of two lists or two matrices, or adds two strings together.

–

Subtraction symbol. Returns the difference of two numbers, or the term-by-term subtraction of two lists or two matrices.

.*

Term by term multiplication for matrices. Returns the term-by-term multiplication of two matrices.

```
Matrix1.*Matrix2
```

Example:

`[[1,2],[3,4]].*[[3,4],[5,6]]` gives [[3,8],[15,24]]

./

Term by term division for matrices. Returns the term-by-term division of two matrices.

```
Matrix1 ./ Matrix2
```

.^

Term by term exponentiation for matrices. Returns the term by term exponentiation for a matrix.

```
Matrix .^ Integer
```

/

Division symbol. Returns the quotient of two numbers, or the term by term quotient of two lists. For division of a matrix by a square matrix, returns the left-multiplication by the inverse of the square matrix.

:=

Stores the evaluated expression in the variable. Note that:= cannot be used with the graphics variables G0–G9. See the command `BLIT`.

`var:=expression`

Example:

`A:=3` stores the value 3 in the variable A

<

Strict less-than-inequality test. Returns 1 if the left side of the inequality is less than the right side, and 0 otherwise. Note that more than two objects can be compared. Thus 6 < 8 < 11 returns 1 (because it is true) whereas 6 < 8 < 3 returns 0 (as it is false).

<=

Less than or equal inequality test. Returns 1 if the left side of the inequality is less than the right side or if the two sides are equal, and 0 otherwise. Note that more than two objects can be compared. See comment above regarding <.

<>

Inequality test. Returns 1 if the inequality is true, and 0 if the inequality is false.

=

Equality symbol. Connects two members of an equation.

==

Equality test. Returns 1 if the left side and right side are equal, and 0 otherwise.

EQ

Tests for the equality of two lists.

Example:

`EQ({1,2,3},{1,2,3})` returns 1

>

Strict greater than inequality test. Returns 1 if the left side of the inequality is greater than the right side, and 0 otherwise. Note that more than two objects can be compared. See comment above regarding <.

>=

Greater than or equal inequality test. Returns 1 if the left side of the inequality is greater than the right side or if the two sides are equal, and 0 otherwise. Note that more than two objects can be compared. See comment above regarding <.

^

Power symbol. Raises a number to a power or a matrix to an integer power.

a2q

Given a symmetric matrix and a vector of variables, returns the quadratic form of the matrix using the variables in the vector.

`a2q(Matrix, [Var1, Var2….])`

Example:

`a2q([[1,2],[4,4]],[x,y])` returns x^2+6*x*y+4*y^2

abcuv

Given three polynomials A, B, and C, returns U and V such that A*U+B*V=C. With a variable as the final argument, U and V are expressed in terms of that variable (if needed); otherwise, x is used.

`abcuv(PolyA, PolyB, PolyC, [Var])`

Example:

`abcuv(x^2+2*x+1,x^2-1,x+1)` returns [1/2-1/2]

additionally

Used in programming with assume to state an additional assumption about a variable.

Example:

```
assume(n,integer);
```

```
additionally(n>5);
```

Airy Ai

Returns the Ai value of the Airy function solution of w''-xw=0.

Airy Bi

Returns the Bi value of the Airy function solution of w''-xw=0.

algvar

Returns the matrix of the symbolic variable names used in an expression. The list is ordered by the algebraic extensions required to build the original expression.

```
algvar(Expr)
```

Example:

`algvar(sqrt(x)+y)` gives $\begin{bmatrix} y \\ x \end{bmatrix}$

AND

Logical And. Returns 1 if the left and right sides both evaluate to true and returns 0 otherwise.

`Expr1` AND `Expr2`

Example:

`3 +1==4 AND 4 < 5` returns 1

append

Appends an element to a list or vector.

```
append((List, Element)
```

or

```
append(Vector, Element)
```

Example:

`append([1,2,3],4)` gives [1,2,3,4]

apply

Returns a vector or matrix containing the results of applying a function to the elements in the vector or matrix.

`apply(Var→f(Var), Vector)` or `apply(Var→f(Var), Matrix)`

Example:

`apply(x→x^3,[1 2 3])` gives [1 8 27]

assume

Used in programming to state an assumption about a variable.

```
assume(Var,Expr)
```

Example:

```
assume(n, integer)
```

basis

Given a matrix, returns the basis of the linear subspace defined by the set of vectors in the matrix.

```
basis(Matrix))
```

Example:

`basis([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])` gives [[-3,0,3],[0,-3,-6]]

betad

The Beta probability density function. Computes the probability density of the beta distribution at x, given parameters α and β.

```
betad(α, β, x)
```

Example:

`betad(2.2, 1.5, 8)` returns 1.46143068876

betad_cdf

The Beta cumulative probability density function. Returns the lower-tail probability of the beta probability density function for the value x, given parameters α and β. With the optional parameter x_2, returns the area under the Beta probability density function between x and x_2.

```
betad_cdf(α, β, x, [x₂])
```

Examples:

`betad_cdf(2, 1, 0.2)` returns 0.04

`betad_cdf(2, 1, 0.2, 0.5)` returns 0.21

betad_icdf

Inverse cumulative beta probability density function. Returns the value x such that the beta lower-tail probability of x, given parameters α and β, is p.

```
betad_icdf(α, β, p)
```

Example:

`betad_icdf(2,1,0.95)` returns 0.974679434481

bounded_function

Argument returned by the limit command, indicating that the function is bounded.

breakpoint

Used in programming to insert an intentional stopping or pausing point.

canonical_form

Returns a second degree trinomial in canonical form.

```
canonical_form(Trinomial,[Var])
```

Example:

`canonical_form(2*x^2-12*x+1)` gives 2*(x-3)^2- 17

cat

Evaluates the objects in a sequence, then returns them concatenated as a string.

```
cat(Object1, Object2,…)
```

Example:

`cat("aaa",c,12*3)` gives "aaac36"

Cauchy

The Cauchy probability density function. Computes the probability density of the Cauchy distribution at x, given parameters x_0 and a. By default, $x_0 = 0$ and a = 1.

cauchy([x_0], [a], x)

Example:

`cauchy(0,1,1)` returns 0.159154943092, as does `cauchy(1)`

Cauchy_cdf

Cumulative Cauchy probability density function. Returns the lower-tail probability of the Cauchy probability density function for the value x, given parameters x_0 and a. With the optional parameter x_2, returns the area under the Cauchy probability density function between x and x_2.

cauchy_cdf(x_0, a, x, [x_2])

Examples:

`cauchy_cdf(0,2,2.1)` returns 0.757762116818

`cauchy_cdf(0,2,2.1,3.1)` returns 0.0598570954516

Cauchy_icdf

Inverse cumulative Cauchy probability density function. Returns the value x such that the Cauchy lower-tail probability of x, given parameters x_0 and a, is p.

cauchy_icdf(x_0, a, p)

Example:

`cauchy_icdf(0, 2, 0.95)` returns 12.6275030293

cFactor

Returns an expression factorized over the complex field (on Gaussian integers if there are more than two).

```
cfactor(Expr)
```

Example:

`cFactor(x^2*y+y)` gives (x+i)*(x-i)*y

charpoly

Returns the coefficients of the characteristic polynomial of a matrix. With only one argument, the variable used in the polynomial is x. With a variable as second argument, the polynomial returned is in terms of that variable.

```
charpoly(Matrix,[Var])
```

Example:

`charpoly([[1,2],[3,4]], z)` returns z^2-5*z- 2

chrem

Returns a vector containing the Chinese remainders for two sets of integers, contained in either two vectors or two lists.

`chrem(List1, List2)` or `chrem(Vector1, Vector2)`

Example:

`chrem([2,3],[7,5])` returns [-12,35]

col

Given a matrix and an integer n, returns the nth column of the matrix as a vector.

```
col(Matrix, Integer)
```

Example:

$\texttt{col}\left(\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}, 2\right)$ returns [2,5,8]

colDim

Returns the number of columns of a matrix.

```
colDim(Matrix)
```

Example:

`colDim` returns 3

comDenom

Rewrites a sum of rational fractions as a one rational fraction. The denominator of the one rational fraction is the common denominator of the rational fractions in the original expression. With a variable as second argument, the numerator and denominator are developed according to it.

```
comDenom(Expr,[Var])
```

Example:

`comDenom(1/x+1/y^2+1)` gives (x*y^2+x+y^2)/ (x*y^2)

companion

Returns the companion matrix of a polynomial.

```
companion(Poly,Var)
```

Example:

`companion(x^2+5x-7,x)` returns $\left(\begin{bmatrix} 0 & 7 \\ 1 & -5 \end{bmatrix}\right)$

compare

Compares two objects and returns 1 if type(Obj1)<type(Obj2) or if type(Obj1)=type(Obj2) and Obj1<Obj2; otherwise, it returns 0.

```
compare(Obj1, Obj2)
```

Example:

`compare(1,2)` gives 1

complexroot

With a polynomial and a real as its two arguments, returns a matrix. Each row of the matrix contains either a complex root of the polynomial with its multiplicity or an interval containing such a root and its multiplicity. The interval defines a (possibly) rectangular region in the complex plane where a complex root lies.

With two additional complex numbers as third and fourth arguments, returns a matrix as described for two arguments, but only for those roots lying in the rectangular region defined by the diagonal created by the two complex numbers.

```
complexroot(Poly, Real, [Complex1], [Complex2])
```

Example:

`complexroot(x^3+8, 0.01)` returns $\begin{bmatrix} -2 & 1 \\ \begin{bmatrix} \frac{1017-1782\cdot i}{1024} & \frac{1026-1773\cdot i}{1024} \end{bmatrix} & 1 \\ \begin{bmatrix} \frac{1395+378\cdot i}{512-512\cdot i} & \frac{-189+702\cdot i}{256+256\cdot i} \end{bmatrix} & 1 \end{bmatrix}$

This matrix indicates there is 1 complex root at x=–2, with another root between the two values in the second row vector and a third root between the two values in the third row vector.

contains

Given a list or vector and an element, returns the index of the first occurrence of the element in the list or vector; if the element does not appear in the list or vector, returns 0.

`contains((List, Element)` or `contains(Vector, Element)`

Example:

`contains({0,1,2,3},2)` returns 3

CopyVar

Copies the first variable into the second variable without evaluation.

`CopyVar(Var1,Var2)`

correlation

Returns the correlation of the elements of a list or matrix.

`correlation(List)` or `correlation(Matrix)`

Example:

correlation$\begin{bmatrix}1 & 2\\ 1 & 1\\ 4 & 7\end{bmatrix}$ returns $\frac{33}{6\cdot\sqrt{31}}$

count

There are two uses for this function, in which first argument is always a mapping of a variable onto an expression. If the expression is a function of the variable, the function is applied to each element in the vector or matrix (the second argument) and the sum of the results is returned; if the expression is a Boolean test, each element in the vector or matrix is tested and the number of elements that pass the test is returned.

`count(Var → Function, Matrix)` or `count(Var → Test, Matrix)`

Example:

count(x→x^2,[1 2 3]) returns 14

`count(x→ x>1, [1 2 3])` returns 2

covariance

Returns the covariance of the elements in a list or matrix.

`covariance(List)` or `covariance(Matrix)`

Example:

covariance$\left(\begin{bmatrix}1 & 2\\ 1 & 1\\ 4 & 7\end{bmatrix}\right)$ returns $\frac{11}{3}$

covariance_correlation

Returns a vector containing both the covariance and the correlation of the elements of a list or matrix.

`covariance_correlation(List)` or

`covariance_correlation(Matrix)`

Example:

`covariance_correlation` $\left(\begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 4 & 7 \end{bmatrix}\right)$ returns $\begin{bmatrix} \frac{11}{3} & \frac{33}{6 \cdot \sqrt{31}} \end{bmatrix}$

cpartfrac

Returns the result of partial fraction decomposition of a rational fraction in the complex field.

`cpartfrac(RatFrac)`

Example:

`cpartfrac` $\left(\frac{x}{4-x^2}\right)$ returns $-\frac{\frac{1}{2}}{x-2} - \frac{\frac{1}{2}}{x+2}$

crationalroot

Returns the list of complex rational roots of a polynomial without indicating the multiplicity.

`crationalroot(Poly)`

Example:

`crationalroot(2*x^3+(-5-7*i)*x^2+ (-4+14*i)*x+8-4*i)` returns $\begin{bmatrix} \frac{3+i}{2} & 2 \cdot i & 1+i \end{bmatrix}$

cumSum

Accepts as argument either a list or a vector and returns a list or vector whose elements are the cumulative sum of the original argument.

`cumSum(List)` or `cumSum(Vector)`

Example:

`cumSum([0,1,2,3,4])` returns [0,1,3,6,10]

DateAdd

Adds NbDays to Date, returning the resulting date in YYYY.MMDD format.

`DATEADD(Date, NbDays)`

Example:

`DATEADD(20081228, 559)` returns 2010.0710

Day of the week

Given a date in YYYY.MMDD format, returns a number between 1 (Monday) and 7 (Sunday) which represents the day of the week associated with the date.

`DAYOFWEEK(Date)`

Example:

`DAYOFWEEK(2006.1228)` returns 4 (for Thursday)

DeltaDays

Calculates the numbers of days between 2 dates, expressed in YYYY.MMDD format.

`DELTADAYS(Date1, Date2)`

Example:

`DELTADAYS(2008.1228,2010.0710)` returns 559

delcols

Given a matrix and an integer n, deletes the nth column from the matrix and returns the result. If an interval of two integers is used instead of a single integer, deletes all columns in the interval and returns the result.

`delcols(Matrix, Integer)` or `delcols(Matrix, Intg1..Intg2)`

Example:

`delcols` $\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2\right)$ returns $\begin{bmatrix}1&3\\4&6\\7&9\end{bmatrix}$

delrows

Given a matrix and an integer n, deletes the nth row from the matrix and returns the result. If an interval of two integers is used instead of a single integer, deletes all rows in the interval and returns the result.

`delrows(Matrix, Integer)` or `delrows(Matrix, Intg1..Intg2)`

Example:

`delrows` $\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2..3\right)$ returns $\begin{bmatrix}1&2&3\end{bmatrix}$

deltalist

Returns the list of the differences between consecutive terms in the original list.

`deltalist(Lst)`

Example:

`deltalist([1,4,8,9])` gives [3,4,1]

deltalist

Returns the list of the differences between consecutive terms in the original list.

`deltalist(Lst)`

Example:

`deltalist([1,4,8,9])` gives [3,4,1]

Dirac

Returns the value of the Dirac delta function for a real number.

`Dirac(Real)`

Example:

`Dirac(1)` gives 0

e

Enters the mathematical constant e (Euler's number).

egcd

Given two polynomials, A and B, returns three polynomials U, V and D such that:

`U(x)*A(x)+V(x)*B(x)=D(x),`

where `D(x)=GCD(A(x),B(x))`, the greatest common divisor of polynomials A and B.

The polynomials can be provided in symbolic form or as lists of coefficients in descending order.

Without a third argument, it is assumed that the polynomials are expressions of x. With a variable as third argument, the polynomials are expressions of it.

`egcd((PolyA, PolyB, [Var])` or `egcd(ListA, ListB, [Var])`

Example:

`egcd((x-1)^2,x^3-1)` gives [-x-2,1,3*x-3]

eigenvals

Returns the sequence of eigenvalues of a matrix.

`eigenvals(Matrix)`

Example:

`eigenvals`$\left(\begin{bmatrix}-2 & -2 & 1\\ -2 & 1 & -2\\ 1 & -2 & -2\end{bmatrix}\right)$ returns [3 -3 3]

eigenvects

Returns the eigenvectors of a diagonalizable matrix.

`eigenvects(Matrix)`

Example:

`eigenvects`$\left(\begin{bmatrix}-2 & -2 & 1\\ -2 & 1 & -2\\ 1 & -2 & -2\end{bmatrix}\right)$ returns $\begin{bmatrix}1 & -3 & -3\\ -2 & 0 & -3\\ 1 & 3 & -3\end{bmatrix}$

eigVl

Returns the Jordan matrix associated with a matrix when the eigenvalues are calculable.

EVAL

Evaluates an expression.

`eval(Expr)`

Example:

`eval(2+3)` returns 5

evalc

Returns a complex expression written in the form `real+i*imag`.

```
evalc(Expr)
```

Example:

`evalc` $\left(\frac{1}{x+y\cdot i}\right)$ returns $\frac{x}{x^2+y^2}-\frac{i\cdot y}{x^2+y^2}$

evalf

Given an expression and a number of significant digits, returns the numerical evaluation of the expression to the given number of significant digits. With just an expression, returns the numerical evaluation based on the CAS settings.

```
evalf(Expr,[Integer])
```

Example:

`evalf(2/3)` gives 0.666666666667

even

Tests whether or not an integer is even. Returns 1 if it is and 0 if it is not.

Example:

`even(1251)` returns 0

exact

Converts a decimal expression to a rational or real expression.

```
exact(Expr)
```

Example:

`exact(1.4141)` gives 14141/10000

EXP

Returns the solution to the mathematical constant e to the power of an expression

```
exp(Expr)
```

Example:

`exp(0)` gives 1

exponential

The discrete exponential probability density function. Computes the probability density of the exponential distribution at x, given parameter k.

```
exponential(x, k)
```

Example:

`exponential(2.1,0.5)` returns 0.734869273133

exponential_cdf

The exponential cumulative probability density function. Returns the lower-tail probability of the exponential probability density function for the value x, given parameter k. With the optional parameter x_2, returns the area under the exponential probability density function between x and x_2.

`exponential_cdf(k, x, [x`$_2$`])`

Examples:

`exponential_cdf(4.2, 0.5)` returns 0.877543571747

`exponential_cdf(4.2, 0.5, 3)` returns 0.122453056238

exponential_icdf

The inverse exponential cumulative probability density function. Returns the value x such that the exponential lower-tail probability of x, given parameter k, is p.

`exponential_icdf(k, p)`

Example:

`exponential_icdf(4.2,0.95)` returns 0.713269588941

exponential_regression

Given a set of points, returns a vector containing the coefficients a and b of y=b*a^x, the exponential which best fits the set of points. The points may be the elements in two lists or the rows of a matrix.

`exponential_regression(Matrix)` or `exponential_regression(List1, List2)`

Example:

`exponential_regression`$\left(\begin{bmatrix} 1.0 & 2.0 \\ 0.0 & 1.0 \\ 4.0 & 7.0 \end{bmatrix}\right)$ returns 1.60092225473,1.10008339351

EXPR

Parses a string into a number or expression and returns the result evaluated.

`EXPR(String)`

Examples:

`expr("2+3")` returns 5

`expr("X+10")` returns 100, if the variable X has the value 90

ezgcd

Uses the EZ GCD algorithm to return the greatest common divisor of two polynomials with at least two variables.

`ezgcd(Poly1,Poly2)`

Example:

`ezgcd(x^2-2*x-x*y+2*y,x^2-y^2)` returns x-y

f2nd

Returns a vector consisting of the numerator and denominator of an irreducible form of a rational fraction.

```
f2nd(RatFrac)
```

Example:

`f2nd`$\left(\frac{x}{x\cdot\sqrt{x}}\right)$ returns $\left[1\ \ \sqrt{x}\right]$

factorial

Returns the factorial of an integer or the solution to the gamma function for a non-integer. For an integer n, factorial(n)=n!. For a non-integer real number a, factorial(a)=a! = Gamma(a + 1).

`factorial(Integer)` or `factorial(Real)`

Examples:

`factorial(4)` returns 24

`factorial(1.2)` returns 1.10180249088

float

`FLOAT_DOM` or `float` is an option of the `assume` command; it is also a name returned by the `type` command.

fMax

Given an expression in x, returns the value of x for which the expression has its maximum value. Given an expression and a variable, returns the value of that variable for which the expression has its maximum value.

```
fMax(Expr,[Var])
```

Example:

`fMax(-x^2+2*x+1,x)` gives 1

fMin

Given an expression in x, returns the value of x for which the expression has its minimum value. Given an expression and a variable, returns the value of that variable for which the expression has its minimum value.

```
fMin(Expr,[Var])
```

Example:

`fMin(x^2-2*x+1,x)` gives 1

format

Returns a real number as a string with the indicated format (f=float, s=scientific, e=engineering).

```
format(Real, String)
```

Example:

`format(9.3456,"s3")` returns 9.35

Fourier a_n

Returns the nth Fourier coefficient a_n=2/T*∫(f(x)*cos(2*pi*n*x/T),a,a+T).

Fourier b_n

Returns the nth Fourier coefficient b_n=2/T*∫(f(x)*sin(2*pi*n*x/T),a,a+T).

Fourier c_n

Returns the nth Fourier coefficient c_n=1/T*∫(f(x)*exp(-2*i*pi*n*x/T),a,a+T).

fracmod

For a given integer n (representing a fraction) and an integer p (the modulus), returns the fraction a/b such that n=a/b(mod p).

```
fracmod(Integern, Integerp)
```

Example:

`fracmod(41,121)` gives 2/3

froot

Returns a vector containing the roots and poles of a rational polynomial. Each root or pole is followed by its multiplicity.

```
froot(RatPoly)
```

Example:

`froot`$\left(\frac{x^5-2\cdot x^4+x^3}{x-3}\right)$ returns [0 3 1 2 3 -1]

fsolve

Returns the numerical solution of an equation or a system of equations. With the optional third argument, you can specify a guess for the solution or an interval within which it is expected that the solution will occur. With the optional fourth argument you can name the iterative algorithm to be used by the solver by specifying bisection_solver, newton_solver, or newtonj_solver.

```
fsolve(Expr,Var,[Guess or Interval],[Method])
```

Example:

`fsolve(cos(x)=x,x,-1..1,bisection_solver)` gives [0.739085133215]

function_diff

Returns the derivative function of a function (as a mapping).

```
function_diff(Fnc)
```

Example:

`function_diff(sin)` gives (_x)→cos(_x)

gammad

Gamma probability density function. Computes the probability density of the gamma distribution at x, given parameters a and t.

```
gammad(a, t, x)
```

Example:

`gammad(2.2,1.5,0.8)` returns 0.510330619114

gammad_cdf

Cumulative gamma distribution function. Returns the lower-tail probability of the gamma probability density function for the value x, given parameters a and t. With the optional fourth argument x_2, returns the area between the two x-values.

`gammad_cdf(a,t,x,[`x_2`])`

Examples:

`gammad_cdf(2,1,2.96)` returns 0.794797087996

`gammad_cdf(2,1,2.96,4)` returns 0.11362471756

gamma_icdf

Inverse cumulative gamma distribution function. Returns the value x such that the gamma lower-tail probability of x, given parameters a and t, is p.

```
gammad_icdf(a,t,p)
```

Example:

`gammad_icdf(2,1,0.95)` returns 4.74386451839

gauss

Given an expression followed by a vector of variables, uses the Gauss algorithm to return the quadratic form of the expression written as a sum or difference of squares of the variables given in the vector.

```
gauss(Expr,VectVar)
```

Example:

`gauss(x^2+2*a*x*y,[x,y])` gives (a*y+x)^2+(- y^2)*a^2

GF

Creates a Galois Field of characteristic p with p^n elements.

```
GF(Integerp, Integern)
```

Example:

`GF(5,9)` gives GF(5,k^9-k^8+2*k^7+2*k^5-k^2+2*k- 2,[k,K,g],undef)

gramschmidt

Given a basis of a vector subspace, and a function that defines a scalar product on this vector subspace, returns an orthonormal basis for that function.

```
gramschmidt(Vector, Function)
```

Example:

`gramschmidt` $\left(\begin{bmatrix}1 & 1+x\end{bmatrix}, (p,q) \rightarrow \int_{-1}^{1} p \cdot q\, dx\right)$ returns $\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1+x-1}{\frac{\sqrt{6}}{3}}\end{bmatrix}$

hadamard

Hadamard bound of a matrix or element by element multiplication of 2 matrices.

```
hadamard(Matrix,[Matrix])
```

Examples:

`hadamard([[1,2],[3,4]])` returns 5√5

`hadamard([[1,2],[3,4]],[[3,4],[5,6]])` returns [[3,8],[15,24]]

halftan2hypexp

Returns an expression with sine, cosine, and tangent rewritten in terms of half-tangent, and sinh, cosh, and tanh rewritten in terms of the natural exponential.

```
halftan_hyp2exp(ExprTrig)
```

Example:

`halftan_hyp2exp(sin(x)+sinh(x))` returns $\frac{2 \cdot \tan\left(\frac{x}{2}\right)}{\tan\left(\frac{x}{2}\right)^2+1}+\frac{\exp(x)-\frac{1}{\exp(x)}}{2}$

halt

Used in programming to go into step-by-step debugging mode.

hamdist

Returns the Hamming distance between two integers.

```
hamdist(Integer1, Integer2)
```

Example:

`hamdist(0x12,0x38)` gives 3

has

Returns 1 if a variable is in an expression, and returns 0 otherwise.

```
has(Expr,Var)
```

Example:

`has(x+y,x)` gives 1

head

Returns the first element of a given vector, sequence or string.

`head(Vector)` or `head(String)` or `head(Obj1, Obj2,…)`

Example:

`head(1,2,3)` gives 1

Heaviside

Returns the value of the Heaviside function for a given real number (i.e. 1 if x>=0, and 0 if x<0).

`Heaviside(Real)`

Example:

`Heaviside(1)` gives 1

horner

Returns the value of a polynomial P(a) calculated with Horner's method. The polynomial may be given as a symbolic expression or as a vector of coefficients.

`horner(Polynomial,Real)`

Examples:

`horner(x^2+1,2)` returns 5

`horner([1,0,1],2)` returns 5

hyp2exp

Returns an expression with hyperbolic terms rewritten as exponentials.

`hyp2exp(Expr)`

Example:

`hyp2exp(cosh(x))` returns $\frac{\exp(x)+\frac{1}{\exp(x)}}{2}$

iabcuv

Returns [u,v] such that au+bv=c for three integers a, b, and c. Note that c must be a multiple of the greatest common divisor of a and b for there to be a solution.

`iabcuv(Intgra, Intgrb, Intgrc)`

Example:

`iabcuv(21,28,7)` gives [-1,1]

ibasis

Given two matrices, interprets them as two vector spaces and returns the vector basis of their intersection.

`ibasis(Matrix1, Matrix2)`

Example:

`ibasis` $\left(\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix},\begin{bmatrix}1&1&1\\0&0&1\end{bmatrix}\right)$ returns [-1, -1, 0]

icontent

Returns the greatest common divisor of the integer coefficients of a polynomial.

```
icontent(Poly,[Var])
```

Example:

`icontent(24x^3+6x^2-12x+18)` gives 6

id

Returns a vector containing the solution to the identity function for the argument(s).

```
id(Object1, [Object2,…])
```

Example:

`id([1 2], 3, 4)` returns [[1 2] 3 4]

identity

Given an integer n, returns the identity matrix of dimension n.

```
identity(Integer)
```

Example:

`identity(3)` returns $\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$

iegcd

Returns the extended greatest common divisor of two integers.

```
iegcd(Integer1, Integer2)
```

Example:

`iegcd(14, 21)` returns [-1, 1, 7]

igcd

Returns the greatest common divisor of two integers or two rational numbers or two polynomials of several variables.

`igcd((Integer1, Integer2)` or `igcd(Ratnl1, Ratnl2)` or `igcd(Poly1, Poly2)`

Example:

`igcd(24, 36)` returns 12

`igcd(2/3,3/4)` returns 1/12

image

Image of a linear application of a matrix.

`image(Matrix)`

Example:

`image([[1,2],[3,6]])` returns [1,3]

interval2center

Returns the center of an interval.

`interval2center(Interval)`

Example:

`interval2center(2..5)` returns 7/2

inv

Returns the inverse of an expression or matrix.

`inv(Expr)` or `inv(Matrix)`

Example:

`inv(9/5)` returns 5/9

iPart

Returns a real number without its fractional part or a list of real numbers each without its fractional part.

`iPart(Real)` or `iPart(List)`

Example:

`iPart(4.3)` returns 4

iquorem

Returns the Euclidean quotient and remainder of two integers.

`iquorem(Integer1, Integer2)`

Example:

`iquorem(63, 23)` returns [2, 17]

jacobi_symbol

Returns the kernel of a linear application of a matrix.

`jacobi_symbol(Integer1, Integer2)`

Example:

`jacobi_symbol(132,5)` gives -1

ker

Returns the Jacobi symbol of the given integers.

`ker(Matrix)`

Example:

`ker([[1 2], [3 6]]` returns [2 1]

laplacian

Returns the Laplacian of an expression with respect to a vector of variables.

```
laplacian(Expr, Vector)
```

Example:

`laplacian(exp(z)*cos(x*y),[x,y,z])` returns –x^2*cos(x*y)*exp(z)- y^2*cos(x*y)*exp(z) +cos(x*y)*exp(z)

latex

Returns the evaluated CAS expression written in Latex format.

```
latex(Expr)
```

Examples:

`latex(1/2)` returns "\frac{1}{2}"

`latex((x^4-1)/(x^2+3)` returns "\frac{(x^{4}-1)}{(x^{2}+3)}"

lcoeff

Returns the coefficient of the term of highest degree of a polynomial. The polynomial can be expressed in symbolic form or as a list.

`lcoeff(Poly)` or `lcoeff(List)` or `lcoeff(Vector)`

Example:

`lcoeff(-2*x^3+x^2+7*x)` returns -2

legendre_symbol

With a single integer n, returns the Legendre polynomial of degree n. With two integers, returns the Legendre symbol of the second integer, using the Legendre polynomial whose degree is the first integer.

```
legendre_symbol(Integer1, [Integer2])
```

Example:

`legendre(4)` gives 35*x^4/8+-15*x^2/4+3/8 while legendre(4,2) returns 443/8 after simplification

length

Returns the length of a list, string or set of objects.

`length(List)` or `length(String)` or `length(Object1, Object2,…)`

Example:

`length([1,2,3])` gives 3

lgcd

Returns the greatest common divisor of a set of integers or polynomials, contained in a list, a vector, or just entered directly as arguments.

`lgcd(List)` or `lgcd(Vector)` or `lgcd(Integer1, Integer2, …)` or `lgcd(Poly1, Poly2, …)`

Example:

`lgcd([45,75,20,15])` gives 5

lin

Returns an expression with the exponentials linearized.

`lin(Expr)`

Example:

`lin((exp(x)^3+exp(x))^2)` gives exp(6*x)+2*exp(4*x)+exp(2*x)

linear_interpolate

Takes a regular sample from a polygonal line defined by a matrix of two rows.

`linear_interpolate(Matrix,Xmin,Xmax,Xstep)`

Example:

`linear_interpolate([[1,2,6,9],[3,4,6,7]],1,9, 1)` returns [[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0], [3.0,4.0,4.5,5.0,5.5,6.0,6.33333333333,6.6666 6666667,7.0]]

linear_regression

Given a set of points, returns a vector containing the coefficients a and b of y=a*x+b, the linear which best fits the set of points. The points may be the elements in two lists or the rows of a matrix.

`linear_regression(Matrix)` or `linear_regression(List1, List2)`

Example:

`linear_regression`$\left(\begin{bmatrix}1.0 & 2.0\\ 0.0 & 1.0\\ 4.0 & 7.0\end{bmatrix}\right)$ returns [1.53…, 0.769…]

LineHorz

Used in the Symbolic view of the Geometry app. Given a real number a or an expression that evaluates to a real number a, draws the horizontal line y=a.

`LineHorz(Exp)` or `LineHorz(Real)`

Example:

`LineHorz(-1)` draws the line which has the equation y = -1

LineTan

Draws the line tangent to f(Var) at Var=Value.

```
LineTan(f(Var), [Var], Value)
```

Example:

LineTan(x^2 − x, 1) draws the line y=x-1; that is, the line tangent to y= x^2 – x at x=1

LineVert

Used in the Symbolic view of the Geometry app. Given a real number a or an expression that evaluates to a real number a, draws the vertical line x=a.

`LineVert(Expr)` or `LineVert(Real)`

Example:

`LineVert(2)` draws the line which has the equation x=2

list2mat

Returns a matrix of n columns made by splitting a list into rows, each containing n terms. If the number of elements in the list is not divisible by n, then the matrix is completed with zeros.

```
list2mat(List, Integer)
```

Example:

`list2mat({1,8,4,9},1)` returns $\begin{bmatrix}1\\8\\4\\9\end{bmatrix}$

lname

Returns a list of the variables in an expression.

```
lname(Expr)
```

Example:

`lname(exp(x)*2*sin(y))` gives [x,y]

lnexpand

Returns the expanded form of a logarithmic expression.

```
lnexpand(Expr)
```

Example:

`lnexpand(ln(3*x))` gives ln(3)+ln(x)

logarithmic_regression

Given a set of points, returns a vector containing the coefficients a and b of y=a*ln(x)+b, the natural logarithmic function which best fits the set of points. The points may be the elements in two lists or the rows of a matrix.

`logarithmic_regression(Matrix)` or `logarithmic_regression(List1, List2)`

Example:

`logarithmic_regression`$\begin{bmatrix} 1.0 & 1.0 \\ 2.0 & 4.0 \\ 3.0 & 9.0 \\ 4.0 & 9.0 \end{bmatrix}$ returns [6.3299..., 0.7207...]

logb

Returns the logarithm of base b of a.

`logb(a,b)`

Example:

`logb(5,2)` gives ln(5)/ln(2) which is approximately 2.32192809489

logistic_regression

Returns y, y', C, y'max, xmax, and R, where y is a logistic function (the solution of y'/y=a*y+b), such that y(x0)=y0 and where [y'(x0),y'(x0+1)...] is the best approximation of the line formed by the elements in the list L.

`logistic_regression(Lst(L),Real(x0),Real(y0))`

Example:

`logistic_regression([0.0,1.0,2.0,3.0,4.0],0.0 ,1.0)` gives [-17.77/(1+exp(-0.496893925384*x+2.82232341488+3.14159265359* i)),-2.48542227469/(1+cosh(- 0.496893925384*x+2.82232341488+3.14159265359* i))]

lu

For a numerical matrix A, returns permutation P, L and U such that PA=LU.

`lu(Matrix)`

Example:

`lu([1 2],[3 4])` returns [[1 2] [[1 0],[3 1]] [[1 2], [0 -2]]]

lvar

Given an expression, returns a list of the functions of the expression which utilize variables, including occurrences of the variables themselves.

`lvar(Expr)`

Example:

`lvar(e^(x)*2*sin(y) + ln(x))` returns [e^(x) sin(y) ln(x)]

map

There are two uses for this function, in which the second argument is always a mapping of a variable onto an expression. If the expression is a function of the variable, the function is applied to each element in the vector or matrix (the first argument) and the resulting vector or matrix is returned; if the expression is a Boolean test, each element in the vector or matrix is tested and the results are returned as a vector or matrix. Each test returns either 0 (fail) or 1 (pass).

map(Matrix, Var → Function) or map(Matrix, Var → Test)

Example:

map([1 2 3], x→x^3) returns [1 8 27]

map([1 2 3], x→ x>1) returns [0 1 1]

mat2list

Returns a vector containing the elements of a matrix.

mat2list(Matrix)

Example:

mat2list([[1 8],[4 9]]) gives [1 8 4 9]

matpow

Given a matrix and an integer n, returns the nth power of the matrix by jordanization.

matpow(Matrix, Integer)

Example:

matpow([[1,2],[3,4]],n) gives [[(sqrt(33)- 3)*((sqrt(33)+5)/2)^n*-6/(-12*sqrt(33))+(-(sqrt(33))-3)*((-(sqrt(33))+5)/2)^n*6/(- 12*sqrt(33)),(sqrt(33)-3)*((sqrt(33)+5)/ 2)^n*(-(sqrt(33))-3)/(-12*sqrt(33))+(- (sqrt(33))-3)*((-(sqrt(33))+5)/2)^n*(- (sqrt(33))+3)/(- 12*sqrt(33))],[6*((sqrt(33)+5)/2)^n*-6/(- 12*sqrt(33))+6*((-(sqrt(33))+5)/2)^n*6/(- 12*sqrt(33)),6*((sqrt(33)+5)/2)^n*(- (sqrt(33))-3)/(-12*sqrt(33))+6*((- (sqrt(33))+5)/2)^n*(-(sqrt(33))+3)/(- 12*sqrt(33))]]

matrix

Given two integers p and q, makes a matrix with p rows and q columns, filled with zeroes. Given a value as a third argument, returns a matrix filled with that value. Given a mapping using j and k, uses the mapping to fill the matrix (j is the current row and k the current column). This function can be used with the apply command as well.

matrix(p, q, [Value or Mapping(j,k)])

Example:

matrix(1,3,5) returns [5 5 5]

MAXREAL

Returns the maximum real number that the HP Prime calculator is capable of representing in Home and CAS views: In the CAS, MAXREAL=1.79769313486*10^{308} In Home view, MAXREAL=9.99999999999E499

mean

Returns the arithmetic mean of a list (with an optional list as a list of weights). With a matrix as argument, returns the mean of the columns.

`mean(List1, [List2])` or `mean(Matrix)`

Example:

`mean([1,2,3],[1,2,3])` gives 7/3

median

Returns the median of a list (with an optional list as a list of weights). With a matrix as argument, returns the median of the columns.

`median(List1, [List2])` or `median(Matrix)`

Example:

`median([1,2,3,5,10,4])` gives 3.5

member

Given a list or vector and an element, returns the index of the first occurrence of the element in the list or vector; if the element does not appear in the list or vector, returns 0. Similar to contains, except that the element comes first in the argument order.

`member(( Element, List)` or `contains(Element, Vector)`

Example:

`member(2, {0,1,2,3})` returns 3

MEMORY

Returns a list containing either integers that represent the memory and storage space or an individual integer for either memory (n=1) or storage space (n=2).

MEMORY()

MEMORY(n)

MINREAL

Returns the minimum real number (closest to zero) that the HP Prime calculator is capable of representing in Home and CAS views:

In the CAS, MINREAL=2.22507385851*10^{-308}

In Home view, MINREAL=1 E-499

modgcd

Uses the modular algorithm to return the greatest common divisor of two polynomials.

```
modgcd(Poly1,Poly2)
```

Example:

`modgcd(x^4-1,(x-1)^2)` gives x-1

mRow

Given an expression, a matrix, and an integer n, multiplies row n of the matrix by the expression.

```
mRow(Expr, Matrix, Integer)
```

Example:

$\texttt{mRow}\left(12, \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, 1\right)$ returns $\begin{bmatrix} 12 & 24 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

mult_c_conjugate

If the given complex expression has a complex denominator, returns the expression after both the numerator and the denominator have been multiplied by the complex conjugate of the denominator. If the given complex expression does not have a complex denominator, returns the expression after both the numerator and the denominator have been multiplied by the complex conjugate of the numerator.

```
mult_c_conjugate(Expr)
```

Example:

$\texttt{mult\_c\_conjugate}\left(\frac{1}{3+2\cdot i}\right)$ returns $\frac{1\cdot(3+2\cdot -i)}{(3+2\cdot i)\cdot(3+2\cdot -i)}$

mult_conjugate

Takes an expression in which the numerator or the denominator contains a square root. If the denominator contains a square root, returns the expression after both the numerator and the denominator have been multiplied by the conjugate of the denominator. If the denominator does not contain a square root, returns the expression after both the numerator and the denominator have been multiplied by the conjugate of the numerator.

```
mult_conjugate(Expr)
```

Example:

$\texttt{mult\_conjugate}(\sqrt{3}-\sqrt{2})$ returns $\frac{(\sqrt{3}-\sqrt{2})\cdot(\sqrt{3}+\sqrt{2})}{\sqrt{3}+\sqrt{2}}$

nDeriv

Given an expression, a variable of differentiation, and a real number h, returns an approximate value of the derivative of the expression, using `f’(x)=(f(x+h)-f(x+h))/(2*h)`.

Without a third argument, the value of h is set to 0.001; with a real as third argument, it is the value of h. With a variable as the third argument, returns the expression above with that variable in place of h.

`nDeriv(Expr,Var, Real)` or `nDeriv(Expr, Var1, Var2)`

Example:

`nDeriv(f(x),x,h)` returns (f(x+h)-(f(x-h)))*0.5/h

NEG

Unary minus. Enters the negative sign.

negbinomial

The negative binomial probability density function. Computes the probability density of the negative binomial distribution at x, given parameters n and k.

```
negbinomial(n, k, x)
```

Example:

`negbinomial(4, 2, 0.6)` returns 0.20736

negbinomial_cdf

The cumulative probability density function for the negative binomial distribution. Returns the lower-tail probability of the negative binomial probability density function for the value x, given parameters n and k. With the optional parameter x_2, returns the area under the negative binomial probability density function between x and x_2.

negbinomial_cdf(n, k, x, [x_2])

Examples:

`negbinomial_cdf(4, 0.5, 2)` returns 0.34375

`negbinomial_cdf(4, 0.5, 2, 3)` returns 0.15625

negbinomial_icdf

The inverse cumulative probability density function for the negative binomial distribution. Returns the value x such that the negative binomial lower-tail probability of x, given parameters n and k, is p.

```
negbinomial_icdf(n, k, p)
```

Example:

`negbinomial_icdf(4, 0.5, 0.7)` returns 5

newton

Uses Newton method to estimate the root of a function, beginning with Guess and calculating Integer iterations. By default, Integer is 20.

```
newton(Expr,Var, [Guess],[Integer])
```

Example:

`newton(3-x^2,x,2)` returns 1.73205080757

normal

Returns the expanded irreducible form of an expression.

`normal(Expr)`

Example:

`normal(2*x*2)` gives 4*x

normalize

Given a vector, returns it divided by its l_2 norm (where the l2 norm is the square root of the sum of the squares of the vector's coordinates).

Given a complex number, returns it divided by its modulus.

`normalize(Vector)` or `normalize(Complex)`

Example:

`normalize(3+4*i)` gives (3+4*i)/5

NOT

Returns the logical inverse of a Boolean expression.

`not(Expr)`

odd

Returns 1 if a given integer is odd, and returns 0 otherwise.

`odd(Integer)`

Example:

`odd(6)` gives 0

OR

Logical Or. Returns 1 if either or both sides evaluates to true and 0 otherwise.

`Expr1` or `Expr2`

Example:

`3 +1==4 OR 8 < 5` returns 1

order_size

Returns the remainder (O term) of a series expansion: limit(x^a*order_size(x),x=0)=0 if a>0.

`order_size(Expr)`

pa2b2

Takes a prime integer n congruent to 1 modulo 4 and returns [a,b] such that a^2+b^2=n.

`pa2b2(Integer)`

Example:

`pa2b2(17)` gives [4 1]

pade

Returns the Pade approximation of an expression, i.e. a rational fraction P/Q such that P/Q=Expr mod x^(n+1) or mod N with degree(P)<p.

```
pade(Expr, Var, Integern, Integerp)
```

Example:

`pade(exp(x), x, 5, 3)` returns $\frac{-3 \cdot x^2 - 24 \cdot x - 60}{x^3 - 9 \cdot x^2 + 36 \cdot x - 60}$

part

Returns the nth subexpression of an expression.

```
part(Expr, Integer)
```

Examples:

`part(sin(x)+cos(x),1)` returns sin(x)

`part(sin(x)+cos(x),2)` returns cos(x)

peval

Given a polynomial defined by a vector of coefficients, and a real value n, evaluates the polynomial at that value.

```
peval(Vector, Value)
```

Example:

`peval([1,0,-2],1)` returns -1

PI

Inserts π.

PIECEWISE

Used to define a piecewise-defined function. Takes as arguments pairs consisting of a condition and an expression. Each of these pairs defines a sub-function of the piecewise function and the domain over which it is active.

$$\mathtt{PIECEWISE}\begin{cases} \text{Case1} & \textit{if } \text{Test1} \\ \text{Case2} & \textit{if } \text{Test2} \\ \quad\dots & \end{cases}$$

Example:

$$\mathtt{PIECEWISE}\begin{cases} -x & \textit{if } x < 0 \\ x^2 & \textit{if } x \geq 0 \end{cases}$$

Note that the syntax varies if the Entry setting is not set to Textbook:

```
PIECEWISE(Case1, Test1, ...[ Casen, Testn])
```

plotinequation

Shows the graph of the solution of inequations with 2 variables.

```
plotinequation(Expr,[x=xrange,y=yrange],[xstep],[ystep])
```

polar_point

Given the radius and angle of a point in polar form, returns the point with rectangular coordinates in complex form.

```
polar_point(Radius, Angle)
```

Example:

`polar_point(2, π/3)` returns point $\left(2\cdot\left(\frac{1}{2}+\frac{i\cdot\sqrt{3}}{2}\right)\right)$

pole

Given a circle and a line, returns the point for which the line is polar with respect to the circle.

```
pole(Crcle,Line)
```

Example:

`pole(circle(0, 1), line(1+i, 2))` returns point(1/2,1/2)

POLYCOEF

Returns the coefficients of a polynomial with roots given in the vector or list argument.

`POLYCOEF(Vector)` or `POLYCOEF(List)`

Example:

`POLYCOEF({-1, 1})` returns {1, 0, -1}

POLYEVAL

Given a vector or list of coefficients and a value, evaluates the polynomial given by those coefficients at the given value.

`POLYEVAL(Vector, Value)` or `POLYEVAL(List, Value)`

Example:

`POLYEVAL({1,0,-1},3)` returns 8

polygon

Draws the polygon whose vertices are elements in a list.

```
polygon(Point1, Point2, …, Pointn)
```

Example:

`polygon(GA,GB,GD)` draws ΔABD

polygonplot

Used in the Geometry app Symbolic view. Given an n × m matrix, draws and connects the points (xk, yk), where xk is the element in row k and column 1, and yk is the element in row k and column j (with j fixed for k=1 to n rows). Thus, each column pairing generates its own figure, resulting in m–1 figures.

```
polygonplot(Matrix)
```

Example:

`polygonplot`$\left(\begin{bmatrix}1 & 2 & 3\\ 2 & 0 & 1\\ -1 & 2 & 3\end{bmatrix}\right)$ draws two figures, each with three points connected by segments.

polygonscatterplot

Used in the Geometry app Symbolic view. Given an n × m matrix, draws and connects the points (xk, yk), where xk is the element in row k and column 1, and yk is the element in row k and column j (with j fixed for k=1 to n rows). Thus, each column pairing generates its own figure, resulting in m— figures.

```
polygonscatterplot(Matrix)
```

Example:

`polygonscatterplot`$\left(\begin{bmatrix}1 & 2 & 3\\ 2 & 0 & 1\\ -1 & 2 & 3\end{bmatrix}\right)$ draws two figures, each with three points connected by segments.

polynomial_regression

Given a set of points defined by two lists, and a positive integer n, returns a vector containing the coefficients (a_n, a_{n-1} … a_0) of y = a_n*x^n + $a_{n-1}x^{n-1}$+ … a_1*x + a_0), the nth order polynomial which best approximates the given points.

```
polynomial_regression(List1, List2, Integer)
```

Example:

`polynomial_regression({1, 2, 3, 4}, {1, 4, 9, 16},3)` returns [0 1 0 0]

POLYROOT

Returns the zeros of the polynomial given as a vector of coefficients.

```
POLYROOT(Vector)
```

Example:

`POLYROOT([1 0 -1])` returns {-1, 1}

potential

Returns a function whose gradient is the vector field defined by a vector and a vector of variables.

```
potential(Vector1, Vector2)
```

Example:

`potential([2*x*y+3,x^2-4*z,-4*y],[x,y,z])` returns x2*y+3*x-4*y*z

power_regression

Given a set of points defined by two lists, returns a vector containing the coefficients m and b of y=b*x^m, the monomial which best approximates the given points.

```
power_regression(List1, List2)
```

Example:

`power_regression({1, 2, 3, 4}, {1, 4, 9, 16})` returns [2 1]

powerpc

Given a circle and a point, returns the real number d2–r2, where d is the distance between the point and the center of the circle, and r is the radius of the circle.

```
powerpc(Circle, Point)
```

Example:

`powerpc(circle(0,1+i),3+i)` gives 8

prepend

Adds an element to the beginning of a list or vector.

`prepend(List, Element)` or `prepend(Vector, Element)`

Example:

`prepend([1,2],3)` gives [3,1,2]

primpart

Returns a polynomial divided by the greatest common divisor of its coefficients.

```
primpart(Poly,[Var])
```

Example:

`primpart(2x^2+10x+6)` gives x^2+5*x+3

product

With an expression as the first argument, returns the product of solutions when the variable in the expression goes from a minimum value to a maximum value by a given step. If no step is provided, it is taken as 1.

With a list as the first argument, returns the product of the values in the list.

With a matrix as the first argument, returns the element-byelement product of the matrix.

`product(Expr, Var, Min, Max, Step)` or `product(List)` or `product(Matrix)`

Example:

`product(n,n,1,10,2)` gives 945

propfrac

Returns a fraction or rational fraction A/B simplified to Q+r/ B, where R<B or the degree of R is less than the degree of B.

`propfrac(Fraction)` or `propfrac(RatFrac)`

Example:

`propfrac(28/12)` gives 2+1/3

ptayl

Given a polynomial P and a value a, returns the Taylor polynomial Q such that P(x)=Q(x – a).

`ptayl(Poly, Value, [Var])`

Example:

`ptayl(x^2+2*x+1,1)` gives x^2+4*x+4

purge

Unassigns a variable name in CAS view.

For example, if f is defined, then purge(f) deletes that definition and returns f to a symbolic state.

`purge(Var)`

Q2a

Given a quadratic form and a vector of variables, returns the matrix of the quadratic form with respect to the given variables.

`q2a(Expr, Vector)`

Example:

`q2a(x^2+2*x*y+2*y^2,[x,y])` returns $\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}$

quantile

Given a list or vector, and a quantile value between 0 and 1, returns the corresponding quantile of the elements of the list or vector.

`quantile(List, Value)` or `quantile(Vector, Value)`

Example:

`quantile([0,1,3,4,2,5,6],0.25)` returns 1

quartile1

Given a list or vector, returns the first quartile of the elements of the list or vector. Given a matrix, returns the first quartile of the columns of the matrix.

`quartile1(List)` or `quartile1(Vector)` or `quartile1(Matrix)`

Example:

`quartile1([1,2,3,5,10,4])` gives 2

quartile3

Given a list or vector, returns the third quartile of the elements of the list or vector. Given a matrix, returns the third quartile of the columns of the matrix.

`quartile3(List)` or `quartile3(Vector)` or `quartile3(Matrix)`

Example:

`quartile3([1,2,3,5,10,4])` returns 5

quartiles

Returns a matrix containing the minimum, first quartile, median, third quartile, and maximum of the elements of a list or vector. With a matrix as argument, returns the 5-number summary of the columns of the matrix.

`quartiles(List)` or `quartiles(Vector)` or `quartiles(Matrix)`

Example:

`quartiles([1,2,3,5,10,4])` returns $\begin{bmatrix}1\\2\\3\\5\\10\end{bmatrix}$

quorem

Returns the Euclidean quotient and remainder of the quotient of two polynomials, each expressed either in symbolic form directly or as a vector of coefficients. If the polynomials are expressed as vectors of their coefficients, this command returns a similar vector of the quotient and a vector of the remainder.

`quorem(Poly1, Poly2)` or `quorem(Vector1, Vector2)`

Example:

`quorem(x^3+2*x^2+3*x+4,-x+2)` returns [-x^2-4*x- 11, 26]

`quorem([1,2,3,4],[-1,2])` returns [[-1, -4, -11] [26]]

QUOTE

Returns an expression unevaluated.

`quote(Expr)`

randbinomial

Returns a random number for the binomial distribution of n trials, each with probability of success p.

`randbinomial(n, p)`

Example:

`randbinomial(10, 0.4)` returns an integer between 0 and 10

randchisquare

Returns a random number from the Chi-square distribution with n degrees of freedom.

`randchisquare(n)`

Example:

`randchisquare(5)` returns a positive real number from the Chi-Square distribution with 5 degrees of freedom

randexp

Given a positive real number, returns a random real number according to the exponential distribution with real a>0.

```
randexp(Real)
```

randfisher

Returns a random number from the F-distribution with numerator n and denominator d degrees of freedom.

```
randfisher(n, d)
```

Example:

`randfisher(5, 2)` returns a real number from the F-distribution with a numerator 5 degrees of freedom and denominator 2 degrees of freedom

randgeometric

Returns a random number from the geometric distribution with probability of success p.

```
randgeometric(p)
```

Example:

`randgeometric(0.4)` returns a positive integer from the geometric distribution with probability of success 0.4

randperm

Given a positive integer, returns a random permutation of [0,1,2,…,n–1].

```
randperm(Intg(n))
```

Example:

`randperm(4)` returns a random permutation of the elements of the vector [0 1 2 3]

randpoisson

Returns a random number from the Poisson distribution, given parameter k.

```
randpoisson(k)
```

Example:

```
randpoisson(5.4)
```

randstudent

Returns a random number from the Student's t-distribution with n degrees of freedom.

```
randstudent(n)
```

Example:

```
randstudent(5)
```

randvector

Given an integer n, returns a vector of size n that contains random integers in the range -99 through 99 with uniform distribution. With an optional second integer m, returns a vector filled with integers in the range (0, m]. With an optional interval as second argument, fills the vector with real numbers in that interval.

```
randvector(n, [m or p..q])
```

ranm

Given one integer n, returns a vector of size n containing random integers in the range [-99, 99], with uniform distribution. Given two integers n and m, returns an nxm matrix. With an interval as the final argument, returns a vector or matrix whose elements are random real numbers confined to that interval.

ratnormal

Rewrites an expression as an irreducible rational fraction.

```
ratnormal(Expr)
```

Example:

`ratnormal`$\left(\frac{x^2-1}{x^3-1}\right)$ returns $\frac{x+1}{x^2+x+1}$

rectangular_coordinates

Given a vector containing the polar coordinates of a point, returns a vector containing the rectangular coordinates of the point.

```
rectangular_coordinates(Vector)
```

Example:

`rectangular_coordinates([1, π/4])` returns $\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$

reduced_conic

Takes a conic expression and returns a vector with the following items:

- The origin of the conic
- The matrix of a basis in which the conic is reduced
- 0 or 1 (0 if the conic is degenerate)
- The reduced equation of the conic
- A vector of the conic's parametric equations

```
reduced_conic(Expr, [Vector])
```

Example:

`reduced_conic(x^2+2*x-2*y+1)` returns

$$[[-1\ \ 0]]\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} 1\,y^2+2\cdot x \left[-1+-i\cdot\left(-\frac{1}{2}\cdot x\cdot x+i\cdot x\right) x-4\,4\,0.1\,x^2+2\cdot x-2\cdot y+1-1+(-i)\cdot\left(\frac{-1}{2}\cdot x\cdot x+(i)\cdot x\right) \quad \right]$$

ref

Performs Gaussian reduction of a matrix.

`ref(Matrix)`

Example:

$\texttt{ref}\begin{bmatrix} 3 & 1 & -2 \\ 3 & 2 & 2 \end{bmatrix}$ returns $\begin{bmatrix} 1 & \frac{1}{3} & \frac{-2}{3} \\ 0 & 1 & 2 \end{bmatrix}$

remove

Given a vector or list, removes the occurrences of Value or removes the values that make Test true and returns the resulting vector or list.

`remove(Value, List)` or `remove(Test, List)`

Example:

`remove(5,{1,2,5,6,7,5})` returns {1,2,6,7}

`remove(x→x≥5, [1 2 5 6 7 5])` returns [1 2]

reorder

Given an expression and a vector of variables, reorders the variables in the expression according to the order given in the vector.

`reorder(Expr, Vector)`

Example:

$\texttt{reorder}(x^2+2*x+y^2,[y,x])$ gives y^2+x^2+2*x

residue

Returns the residue of an expression at a value.

`residue(Expr, Var, Value)`

Example:

`residue(1/z,z,0)` returns 1

restart

Purges all the variables.

`restart(NULL)`

resultant

Returns the resultant (i.e. the determinant of the Sylvester matrix) of two polynomials.

`resultant(Poly1, Poly2, Var)`

Example:

`resultant(x^3+x+1, x^2-x-2,x)` returns -11

revlist

Reverses the order of the elements in a list or vector.

`revlist(List)` or `revlist(Vector)`

Example:

`revlist([1,2,3])` returns [3,2,1]

romberg

Uses Romberg's method to return the approximate value of a definite integral.

`romberg(Expr, Var, Val1, Val2)`

Example:

`romberg(exp(x^2),x,0,1)` gives 1.46265174591

row

Given a matrix and an integer n, returns the row n of the matrix. Given a matrix and an interval, returns a vector containing the rows of the matrix indicated by the interval.

`row(Matrix, Integer)` or `row(Matrix, Interval)`

Example:

`row` $\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2\right)$ returns [4 5 6]

rowAdd

Given a matrix and two integers, returns the matrix obtained from the given matrix after the row indicated by the second integer is replaced by the sum of the rows indicated by the two integers.

`rowAdd(Matrix, Integer1, Integer2)`

Example:

`rowAdd` $\left(\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix},1,2\right)$ returns $\begin{bmatrix}1&2\\4&6\\5&6\end{bmatrix}$

rowDim

Returns the number of rows of a matrix.

`rowDim(Matrix)`

Example:

`rowDim` $\left(\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}\right)$ gives 2

rowSwap

Given a matrix and two integers, returns the matrix obtained from the given matrix after swapping the two rows indicated by the two integers.

```
rowSwap(Matrix,Integer1,Integer2)
```

Example:

`rowSwap` $\left(\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}, 1, 2\right)$ returns $\begin{bmatrix}3 & 4\\ 1 & 2\\ 5 & 6\end{bmatrix}$

rsolve

Given an expression defining a recurrence relation, a variable, and an initial condition, returns the closed form solution (if possible) of the recurrent sequence. Given three lists, each containing multiple items of the above nature, solves the system of recurrent sequences.

```
rsolve(Expr, Var, Condition) or rsolve(List1, List2, List3)
```

Example:

`rsolve(u(n+1)=2*u(n)+n,u(n),u(0)=1)` returns [-n+2*2^n-1]

select

Given a test expression in a single variable and a list or vector, tests each element in the list or vector and returns a list or vector containing the elements that satisfy the test.

```
select(Test, List) or select(Test, Vector)
```

Example:

`select(x→x>=5,[1,2,6,7])` returns [6,7]

seq

Given an expression, a variable defined over an interval, and a step value, returns a vector containing the sequence obtained when the expression is evaluated within the given interval using the given step. If no step is provided, the step used is 1.

```
seq(Expr, Var=Interval, [Step])
```

Example:

`seq(`2^k`,k=0..8)` gives [1,2,4,8,16,32,64,128,256]

seqsolve

Similar to rsolve. Given an expression defining a recurrence relation in terms of n and/or the previous term (x), followed by a vector of variables and an initial condition for x (the 0th term), returns the closed form solution (if possible) for the recurrent sequence. Given three lists, each containing multiple items of the above nature, solves the system of recurrent sequences.

`seqsolve(Expr, Vector, Condition)` or `seqsolve(List1, List2, List3)`

Example:

`seqsolve(2x+n,[x,n],1)` gives -n-1+2*2^n

shift

Given a list or vector and an integer n, moves the elements of that list or vector either n places to the left, if n>0, or n places to the right, if n<0. If no integer is provided, n=-1 by default and all elements are moved one place to the left.

Elements that move off the list to one side are replaced by 0 on the opposite side.

Given a first integer and a second integer n, moves the first integer bitwise either n bits to the left, if n>0, or n bits to the right, if n<0.

`shift(list, integer)` or `shift(vector, integer)` or `shift(integer1, integer2)`

Example:

`shift({1,2,3},2)` returns {3, 0, 0}

shift_phase

Returns the result of applying a phase shift of pi/2 to a trigonometric expression.

`shift_phase(Expr)`

Example:

`shift_phase(sin(x))` gives -cos((pi+2*x)/2)

signature

Returns the signature of a permutation.

`signature(Vector)`

Example:

`signature([2 1 4 5 3])` returns –1

simult

Returns the solution to a system of linear equations or several systems of linear equations presented in matrix form. In the case of one system of linear equations, takes a matrix of coefficients and a column matrix of constants, and returns the column matrix of the solution.

`simult(Matrix1, Matrix2)`

Example:

`simult` $\left(\begin{bmatrix}3 & 1\\3 & 2\end{bmatrix},\begin{bmatrix}-2\\2\end{bmatrix}\right)$ returns $\begin{bmatrix}-2\\4\end{bmatrix}$

sincos

Returns an expression with the complex exponentials rewritten in terms of sin and cos.

`sincos(Expr)`

Example:

`sincos(exp(i*x))` gives cos(x)+(i)*sin(x)

spline

Given two lists or vectors (one for the x-values and one for the y-values), as well as a variable and an integer degree, returns the natural spline through the points given by the two lists. The polynomials in the spline are in terms of the given variable and are of the given degree.

`spline(ListX, ListY, Var, Integer)` or `spline(VectorX, VectorY, Var, Integer)`

Example:

`spline({0,1,2},{1,3,0},x,3)` returns

$$\left[\frac{-5}{4}\cdot x^{3}+\frac{13}{4}\cdot x+1 \quad \frac{5}{4}\cdot(x-1)^{3}+\frac{-15}{4}\cdot(x-1)^{2}-\frac{1}{2}\cdot(x-1)+3\right]$$

sqrfree

Returns the factorization of the argument, gathering the terms with the same exponent.

`sqrfree(Expr)`

Example:

`sqrfree((x-2)^7*(x+2)^7*(x^4-2*x^2+1))` returns (x^2-1)^2*(x^2-4)^7

sqrt

Returns the square root of an expression.

`sqrt(Expr)`

Example:

`sqrt(50)` gives 5*sqrt(2)

srand

Returns an integer and initializes the sequence of random numbers for CAS-based functions that generate random numbers.

`srand` or `srand(Integer)`

stddev

Returns the standard deviation of the elements of a list or a list of the standard deviations of the columns of a matrix. The optional second list is a list of weights.

`stddev(List1, [List2])` or `stddev(Vector1, [Vector2])` or `stddev(Matrix)`

Example:

`stddev({1,2,3})` returns $\frac{\sqrt{6}}{3}$

stddevp

Returns the population standard deviation of the elements of a list or a list of the population standard deviations of the columns of a matrix. The optional second list is a list of weights.

`stddevp(List1, [List2])` or `stddevp(Vector1, [Vector2])` or `stddevp(Matrix)`

Example:

`stddevp({1,2,3})` gives 1

sto

Stores a real or string in a variable.

```
sto((Real or Str),Var)
```

sturmseq

Returns the Sturm sequence for a polynomial or a rational fraction.

```
sturmseq(Poly,[Var])
```

Example:

`sturmseq(x^3-1,x)` gives [1 [[1 0 0 -1] [3 0 0] 9] 1]

subMat

Extracts from a matrix a sub matrix whose diagonal is defined by four integers. The first two integers define the row and column of the first element and the last two integers define the row and column of the last element of the sub matrix.

```
subMat(Matrix, Int1, Int2, Int3, Int4)
```

Example:

`subMat` $\left(\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}, 2, 1, 3, 2\right)$ returns $\begin{bmatrix}3 & 4\\ 5 & 6\end{bmatrix}$

suppress

Given a list and an element, deletes the first occurrence of the element in the list (if there is one) and returns the result.

```
suppress(List, Element)
```

Example:

`suppress([0 1 2 3 2],2)` returns [0 1 3 2]

surd

Given an expression and an integer n, returns the expression raised to the power 1/n.

```
surd(Expr, Integer)
```

Example:

`surd(8,3)` gives -2

sylvester

Returns the Sylvester matrix of two polynomials.

```
sylvester(Poly1, Poly2, Var)
```

Example:

`sylvester(x`^{2}`-1,x`^{3}`-1,x)` gives $\begin{bmatrix}1&0&-1&0&0\\0&1&0&-1&0\\0&0&1&0&-1\\1&0&0&-1&0\\0&1&0&0&-1\end{bmatrix}$

table

Defines an array where the indexes are strings or real numbers.

`table(SeqEqual(index_name=element_value))`

tail

Given a list, string, or sequence of objects, returns a vector with the first element deleted.

`tail(List)` or `tail(Vector)` or `tail(String)` or `tail(Obj1, Obj2,…)`

Example:

`tail([3 2 4 1 0])` gives [2 4 1 0]

tan2cossin2

Returns an expression with tan(x) rewritten as (1–cos(2*x))/ sin(2*x).

`tan2cossin2(Expr)`

Example:

`tan2cossin2(tan(x))` gives (1-cos(2*x))/sin(2*x)

tan2sincos2

Returns an expression with tan(x) rewritten as sin(2*x)/ (1+cos(2*x)).

`tan2sincos2(Expr)`

Example:

`tan2sincos2(tan(x))` gives sin(2*x)/(1+cos(2*x)

transpose

Returns a matrix transposed (without conjugation).

`transpose(Matrix)`

Example:

`transpose`$\left(\begin{bmatrix}1&2\\3&4\end{bmatrix}\right)$ returns $\begin{bmatrix}1&3\\2&4\end{bmatrix}$

trunc

Given a value or list of values, as well as an integer n, returns the value or list truncated to n decimal places. If n is not provided, it is taken as 0. Accepts complex numbers.

`trunc(Real, Integer)` or `trunc(List, Integer)`

Example:

`trunc(4.3)` gives 4

tsimplify

Returns an expression with transcendentals rewritten as complex exponentials.

`tsimplify(Expr)`

Example:

`tsimplify(exp(2*x)+exp(x))` gives exp(x)^2+exp(x)

type

Returns the type of an expression (e.g. list, string).

`type(Expr)`

Example:

`type("abc")` gives DOM_STRING

unapply

Returns the function defined by an expression and a variable.

`unapply(Expr,Var)`

Example:

`unapply(2*x^2,x)` gives (x)→2*x^2

uniform

The discrete uniform probability density function. Computes the probability density of the uniform distribution at x, given parameters a and b.

`uniform(a, b, x)`

Example:

`uniform(1.2, 3.5, 3)` returns 0.434782608696

uniform_cdf

The cumulative uniform probability density function. Returns the lower-tail probability of the uniform probability density function for the value x, given parameters a and b. With the optional parameter x_2, returns the area under the uniform probability density function between x and x_2.

`uniform_cdf(a, b, x, [x2])`

Examples:

`uniform_cdf(1.2, 3.5, 3)` returns 0.782608695652

`uniform_cdf(1.2, 3.5, 2, 3)` returns 0.434782608696

uniform_icdf

The inverse cumulative uniform probability density function. Returns the value x such that the uniform lower-tail probability of x, given parameters a and b, is p.

```
uniform_icdf(a, b, p)
```

Example:

`uniform_icdf(3.2, 5.7, 0.48)` returns 4.4

UNION

Concatenates the inputs in a list where all duplicates have been removed.

Example:

`UNION({1,2,3}, {2,4,8}, 10)` returns {1, 2, 3, 4, 8, 10}

valuation

Returns the valuation (degree of the term of lowest degree) of a polynomial. With only a polynomial as argument, the valuation returned is for x. With a variable as second argument, the valuation is performed for it.

```
valuation(Poly,[Var])
```

Example:

`valuation(x^4+x^3)` gives 3

variance

Returns the variance of a list or the list of variances of the columns of a matrix. The optional second list is a list of weights.

`variance(List1, [List2])` or `variance(Matrix)`

Example:

`variance({3, 4, 2})` returns 2/3

vpotential

Given a vector V and a vector of variables, returns the vector U such that curl(U)=V.

```
vpotential(Vector1, Vector2)
```

Example:

`vpotential([2*x*y+3,x2-4*z,-2*y*z],[x,y,z])` returns

$$\left[0 \quad -2\cdot x\cdot y\cdot z \quad 4\cdot x\cdot z-\frac{1}{3}\cdot x^{3}+3\cdot y\right]$$

VERSION

Returns a string containing the version numbers of the various components of the system, as displayed in the "About HP Prime calculator" help page. Given an integer n, returns only the version number for that specific component. The components are identified by the following integers:

- **1**—Software version
- **2**—Hardware version
- **3**—CAS version
- **4**—Product serial number
- **5**—Operating system version

weibull

The Weibull probability density function. Computes the probability density of the Weibull distribution at x, given parameters k, n, and t. By default, t=0.

```
weibull(k, n, [t], x)
```

Example:

`weibull(2.1, 1.2, 1.3)` returns 0.58544681204, as does `weibull(2.1, 1.2, 0, 1.3)`

weibull_cdf

The cumulative probability density function for the Weibull distribution. Returns the lower-tail probability of the Weibull probability density function for the value x, given parameters k, n, and t. By default, t=0. With the optional parameter x_2, returns the area under the Weibull probability density function between x and x_2.

`weibull_cdf(k, n, [t], x, [x`$_2$`])`

Examples:

`weibull_cdf(2.1, 1.2, 1.9)` returns 0.927548261801

`weibull_cdf(2.1, 1.2, 0, 1.9)` returns 0.927548261801

`weibull_cdf(2.1, 1.2, 1, 1.9)` returns 0.421055367782

weibull_icdf

The inverse cumulative probability density function for the Weibull distribution. Returns the value x such that the Weibull lower-tail probability of x, given parameters k, n, and t, is p. By default, t=0.

```
weibull_icdf(k, n, [t], x)
```

Examples:

`weibull_icdf(4.2, 1.3, 0.95)` returns 1.68809330364

`weibull_icdf(4.2, 1.3, 0, 0.95)` returns 1.68809330364

when

Used to introduce a conditional statement.

XOR

Exclusive or. Returns 1 if the first expression is true and the second expression is false or if the first expression is false and the second expression is true. Returns 0 otherwise.

```
Expr1 XOR Expr2
```

Example:

`0 XOR 1` returns 1

zip

Applies a bivariate function to the elements of two lists or vectors and returns the results in a vector. Without the default value the length of the vector is the minimum of the lengths of the two lists; with the default value, the shorter list is padded with the default value.

`zip(‘function’List1, List2, Default)` or `zip(‘function’, Vector1, Vector2, Default)`

Example:

`zip('+',[a,b,c,d], [1,2,3,4])` returns [a+1 b+2 c+3 d+4]

ztrans

z transform of a sequence.

`ztrans(Expr,[Var],[ZtransVar])`

Example:

`ztrans(a^n,n,z)` returns –z/(a-z)

|

Found in the Catalog menu and the Template menu, the where command has several uses associated with variable declarations. For one, it is used to substitute values for one or more variables in an expression. It can also be used to define the domain of a variable.

`Expr|Var=Val` or `Expr|{Var1=Val1, Var2=Val2…Varn=Valn}` or `Expr|Var>n` or `Expr|Var<n` and so on.

Examples:

`(X+Y)|{X=2, Y=6}` returns 8

`int((1-x)^p|p>0,x,0,1)` returns ((-x+1)^(p+1))/(-p-1)

^{2}

Returns the square of an expression.

`(Expr)`^{2}

π

Inserts pi.

∂

Inserts a template for a partial derivative expression.

Σ

Inserts a template for a summation expression.

–

Inserts a minus sign.

√

Inserts a square root sign.

∫

Returns the integral of an expression.

When one expression is used as an argument, this command returns the indefinite integral with respect to x.

Optionally, you can specify the variable of integration and the bounds of a definite integral using three additional arguments.

Examples:

`int(1/x)` returns ln(abs(x))

`int(sin(x),x,0,π)` returns 2

`int(1/(1-x^4),x,2,3))` returns -1/4*(2*atan(2)+ln(3))+1/4*(2*atan(3)-ln(2)+ln(4))

≠

Inequality test. Returns 1 if the left and right sides are not equal and 0 if they are equal.

≤

Less than or equal inequality test. Returns 1 if the left side of the inequality is less than the right side or if the two sides are equal, and 0 otherwise.

≥

Greater than or equal inequality test. Returns 1 if the left side of the inequality is greater than the right side or if the two sides are equal, and 0 otherwise.

▶

Evaluates the expression then stores the result in variable var. Note that ▯ cannot be used with the graphics G0–G9. See the command BLIT.

`expression ▶ var`

i

Inserts the imaginary number *i*.

–1

Returns the inverse of an expression.

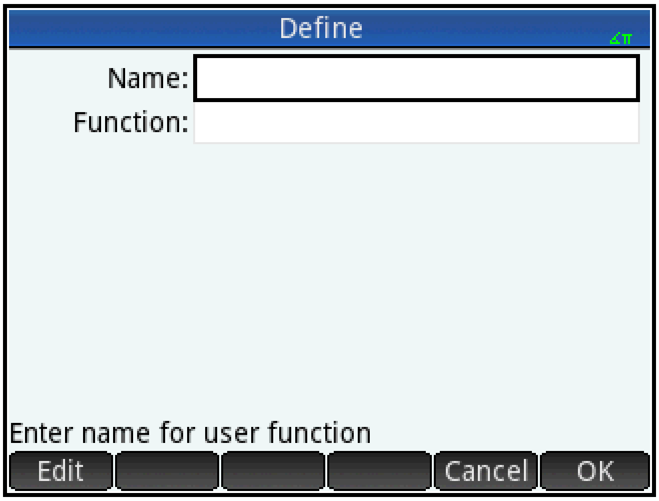
`(Expr)`$^{-1}$

Creating your own functions

You can create your own function by writing a program (see chapter 5) or by using the simpler `DEFINE` functionality. Functions you create yourself appear on the User menu (one of the Toolbox menus).

Suppose you wanted to create the function SINCOS(A,B)=SIN(A)+COS(B)+C.

1. Press Shift xtθn (Define).



Define

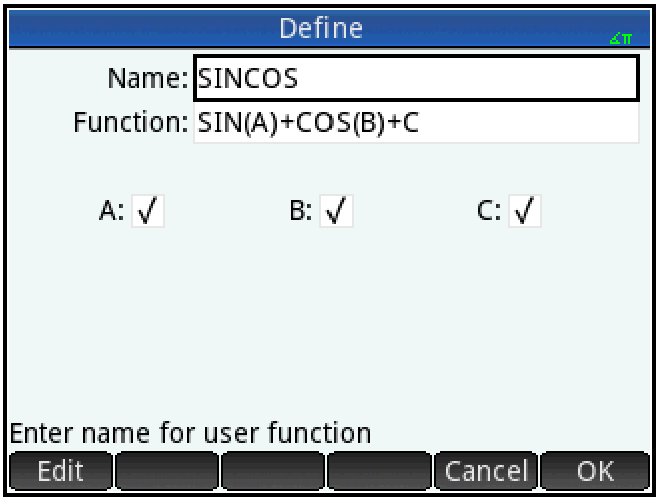
Name:

Function:

Enter name for user function

2. In the **Name** field, enter a name for the function—for example, SINCOS—and tap OK.

3. In the **Function** field, enter the function. SIN ALPHA A ▶ + COS ALPHA B ▶ ALPHA C OK



Define

Name: SINCOS

Function: SIN(A)+COS(B)+C

A: ✓ B: ✓ C: ✓

Enter name for user function

New fields appear below your function, one for each variable used in defining it. You need to decide which ones are to be input arguments for your functions and which ones are global variables whose values are not input within the function. In this example, we'll make A and B input variables, so our new function takes two arguments. The value of C will be provided by global variable C (which by default is zero).

4. Make sure that `A` and `B` are selected and `C` is not.

5. Tap OK.

   You can run your function by entering it on the entry line in Home view, or be selecting it from the USER menu. You enter the value for each variable you chose to be a parameter. In this example. we chose A and B to be parameters. Thus you might enter SINCOS(0.5, 0.75). With C=0 and in radians mode, this would return 1.211…

23 Variables

Variables are objects that have names and contain data. They are used to store data, either for later use or to control settings in the Prime system. There are four types of variables, all of which can be found in the **Vars** menu by pressing Vars:

- Home variables
- CAS variables
- App variables
- User variables

The Home and app variables all have names reserved for them. They are also typed; that is, they can contain only certain types of objects. For example, the Home variable A can only contain a real number. You use Home variables to store data that is important to you, such as matrices, lists, real numbers, etc. You use app variables to store data in apps or to change app settings. You can accomplish these same tasks via the user interface of an app, but app variables give you a quick way of doing these tasks, either from Home or within a program. For example, you can store the expression "`SIN(X)`" in the Function app variable `F1` in Home View, or you could open the Function app, navigate to `F1(X)`, and enter `SIN(X)` in that field.

CAS and user variables can be created by the user and they have no particular type. Their names may be of any length as well. Thus, `diff(t2,t)` returns `2*t` and `diff((bt)2, bt)` returns `2*bt` for the CAS variables `t` and `bt`. Further evaluation of `2*bt` will only return `2*bt`, unless an object has been stored in `bt`. For example, if you enter `bt:={1,2,3}` and then enter `diff((bt)2, bt)`, the CAS will still return `2*bt`. But if you evaluate that result (using the `EVAL` command), the CAS will now return `{2,4,6}`.

User variables are explicitly created by the user. You create user variables either in a program or by assignment in Home view. User variables created in a program are either declared as local or exported as global. User variables created by assignment or exported from a program will show up in the Vars User menu. Local variables exist only within their own program.

The following sections describe the various processes associated with variables, such as creating them, storing objects in them, and retrieving their contents. The rest of the chapter contains tables that list all the Home and app variable names.

Working with variables

Working with Home variables

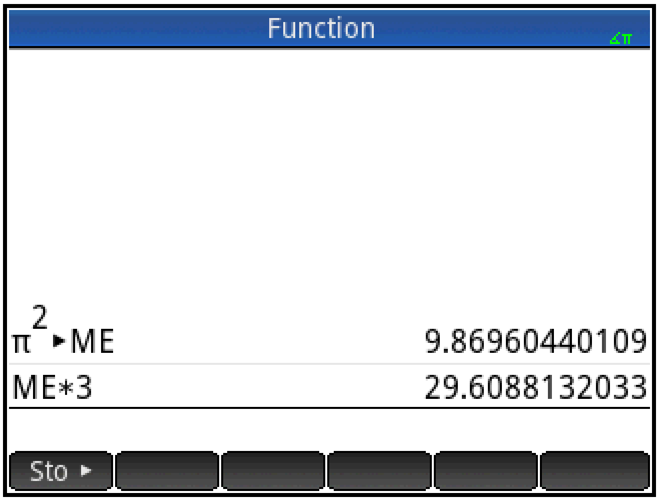
Example 1: Assign π^2 to the Home variable A and then calculate 5*A

1. Press Settings to display Home view.

2. Assign π^2 to A:

   Shift 3 x^2 Sto ▸ ALPHA Vars Enter

3. Multiply A by 5:5 × Vars Enter


Function

π^2 ▸ME	9.86960440109
ME*3	29.6088132033

This example illustrates the process for storing and using any Home variable, not just the Real Home variables A–Z. It is important to match the object you want to store to the correct type of Home variable. See Home variables on page 473 for details.

Working with user variables

Example 2: Create a variable called ME and assign π^2 to it.

1. Press ⌂ Settings to display Home view.

2. Assign π^2 to `ME`:

   Shift 3 x^2 Sto ▸ ALPHA +/- ALPHA a b/c Enter

3. A message appears asking if you want to create a variable called `ME`. Tap OK or press Enter to confirm your intention.

   You can now use that variable in subsequent calculations: `ME*3` will yield `29.6`..., for example.

Example 3: You can also store objects in variables using the assignment operator: `Name:=Object`. In this example, we'll store `{1,2,3}` in the user variable `YOU`.

1. Assign the list to the variable using the assignment operator:=.

   ALPHA ALPHA 1 Eval 4 − ALPHA Shift . Shift 8 1 Eval 2 Eval 3 a b/c Enter

2. A message appears asking if you want to create a variable called `YOU`. Tap OK or press Enter to confirm your intention.

   The variable YOU is created and contains the list `{1,2,3}`. You can now use that variable in subsequent calculations: For example, `YOU+60` will return `{61,62,63}`.

Working with app variables

Just as you can assign values to Home and user variables, you can assign values to app variables. You can modify Home settings on the Home Settings screen (Shift Settings). But you can also modify a Home setting from Home view by assigning a value to the variable that represents that setting. For example, entering `Base:=0` Enter in Home view forces the Home settings field **Integer** (for the integer base) to binary. A value of 1 would force it to octal, 2 to decimal, and 3 to hex. Another example: you can change the angle measure setting from radians to degrees by entering `HAngle:=1` Enter in Home view.

Entering `HAngle:=0` Enter forces the setting to return to radians.

You can see what value has been assigned to a variable—whether Home, app, or user—by entering its name in Home view and pressing Enter. You can enter the name letter by letter, or choose the variable from the Variables menu by pressing Vars.

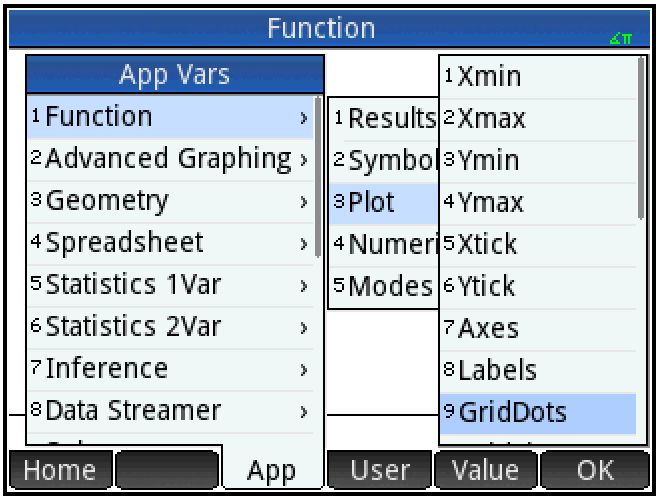
More about the Vars menu

Besides the four variable menus, the **Vars** menu contains a toggle. If you want the value of a variable instead of its name when you choose it from the **Vars** menu, tap Value. A white dot will appear next to the menu button label to indicate that it is active and that variable values rather than names will be returned upon selection.

For the Home and app variables, use the **Vars** menu to get help on the purpose of any of these variables. Select the variable of interest and press Help. Suppose, for example, that you wanted to get help on the Function app variable `GridDots`:

1. Press Vars to open the **Vars** menu.

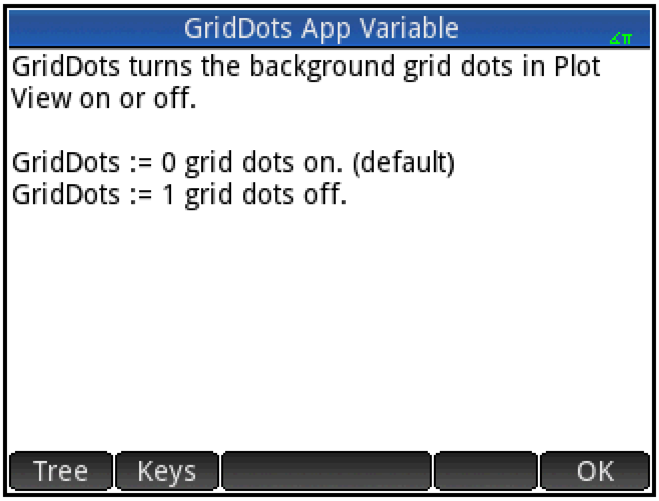
2. Tap App to open the app variables menu. (If you were interested in a Home variable instead, you would tap Home instead.)



3. Use the cursor keys to navigate to the variable of interest.

4. Press Help User to see the help about that variable.

5. Tap OK to exit or Esc Clear to return to the current **Vars** submenu.


GridDots App Variable

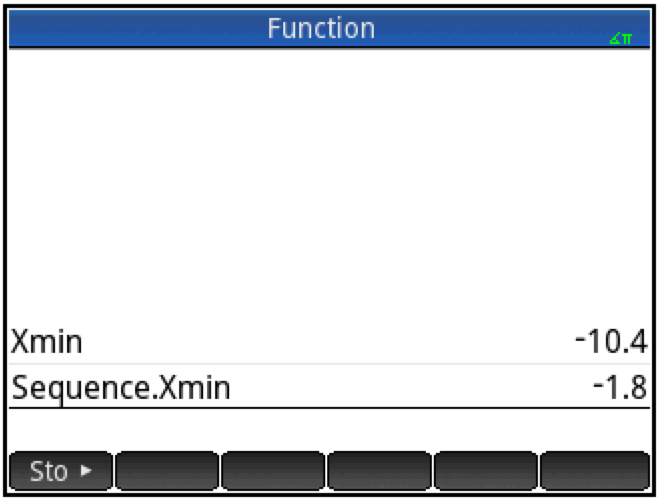
GridDots turns the background grid dots in Plot View on or off.

GridDots := 0 grid dots on. (default)
GridDots := 1 grid dots off.

Qualifying variables

Some app variable names are shared by multiple apps. For example, the Function app has a variable named `Xmin`, but so too does the Polar app, the Parametric app, the Sequence app, and the Solve app. Although named identically, these variables usually hold different values. If you attempt to retrieve the contents of a variable that is used in more than one app by entering just its name in Home view, you will get the contents of that version of the variable in the current app. For example, if the Function app is active and you enter `Xmin` in Home view, you will get the value of `Xmin` from the Function app. If you want the value of Xmin from, say, the Sequence app, you must qualify the variable name. Enter `Sequence.Xmin` to retrieve the value of Xmin from the Sequence app.

In the following figure, the value of `Xmin` from the Function app was retrieved first (–10.4...). The qualified variable name entered second retrieved the value of `Xmin` from the Sequence app (–1.8).



```
Function
Xmin                -10.4
Sequence.Xmin        -1.8
Sto ▸
```

Note the syntax required: `app_name.variable_name`.

The app can be any of the 18 HP apps, or one you have created based on a built-in app. The name of the app variable must match a name listed in the app variables tables below. Spaces are not allowed in an app name and must be represented by the underscore character: Shift _ .

TIP: Non-standard characters in variables name—such as Σ and σ—can be entered by selecting them from the special symbols palette (Shift 9) or from the characters menu (Shift Vars).

Home variables

The Home variables are accessed by pressing Vars and tapping Home.

Category	Names
Real	A to Z and θ For example, 7.45 Sto ▸ A
Complex	Z0 to Z9 For example, 2+3×i Sto ▸ Z1 or(2,3) Sto ▸ Z1 (depending on your Complex number settings)
List	L0 to L9 For example, {1,2,3} Sto ▸ L1.
Matrix	M0 to M9 Store matrices and vectors in these variables. For example, [[1,2],[3,4]] Sto ▸ M1.
Graphics	G0 to G9

Category	Names
Settings	HAngle
	HFormat
	HSeparator
	HDigits
	HComplex
	Entry
	Base
	Bits
	Signed
System	Date
	Time
	Language
	Notes
	Programs
	TOff
	HVars
	DelHVars

App variables

The app variables are accessed by pressing a and tapping . They are grouped below by app. Note that if you have customized a built-in app, your app will appear on the App variables menu under the name you gave it. You access the variables in a customized app in the same way that you access the variables in built-in apps.

Function app variables

Category	Names	
Results (explained below)	SignedArea	Root
	Extremum	Slope
	Isect	
Symbolic	`F1`	`F6`
	`F2`	`F7`
	`F3`	`F8`
	`F4`	`F9`
	`F5`	`F0`
Plot	`Axes`	`Xmin`
	`Cursor`	`Xtick`
	`GridDots`	`Xzoom`

Category	Names	
	GridLines	Ymax
	Labels	Ymin
	Method	Ytick
	Recenter	Yzoom
	Xmax	
Numeric	NumStart	NumType
	NumStep	NumZoom
	NumIndep	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Results variables

Extremum

Contains the value from the last use of the Extremum function from the Fcn menu in the Plot view of the Function app. The app function EXTREMUM does not store results to this variable.

Isect

Contains the value from the last use of the Isect function from the Fcn menu in the Plot view of the Function app. The app function ISECT does not store results to this variable.

Root

Contains the value from the last use of the Root function from the Fcn menu in the Plot view of the Function app. The app function ROOT does not store results to this variable.

SignedArea

Contains the value from the last use of the Signed Area function from the Fcn menu in the Plot view of the Function app. The app function AREA does not store results to this variable.

Slope

Contains the value from the last use of the Slope function from the Fcn menu in the Plot view of the Function app. The app function SLOPE does not store results to this variable.

Geometry app variables

Category	Names	
Plot	Axes	GridDots
	GridLines	Labels
	PixSize	ScrollText
	Xmax	Xmin
	Ymax	Ymin
	XTick	Ytick
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Spreadsheet app variables

Category	Names	
Numeric	ColWidth	RowHeight
	Row	Col
	Cell	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Solve app variables

Category	Names	
Results (explained below)	SignedArea	Root
	Extremum	Slope
	Isect	
Symbolic	E1	E6
	E2	E7
	E3	E8

Category	Names	
	E4	E9
	E5	E0
Plot	Axes	Xmin
	Cursor	Xtick
	GridDots	Xzoom
	GridLines	Ymax
	Labels	Ymin
	Method	Ytick
	Recenter	Yzoom
	Xmax	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Advanced Graphing app variables

Category	Names	
Symbolic	V1	V6
	V2	V7
	V3	V8
	V4	V9
	V5	V0
Plot	Axes	Xmin
	Cursor	Xtick
	GridDots	Xzoom
	GridLines	Ymax
	Labels	Ymin
	Recenter	Ytick
	Xmax	Yzoom
Numeric	NumXStart	NumIndep
	NumYStart	NumType
	NumXStep	NumXZoom
	NumYStep	NumYZoom
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Statistics 1Var app variables

Category	Names	
Results (explained below)	NbItem	ΣX
	MinVal	ΣX2
	Q1	MeanX
	MedVal	sX
	Q3	σX
	MaxVal	serrX
		ssX
Symbolic	H1	H4
	H2	H5

Category	Names	
	H3	
Plot	Axes	Xmax
	Cursor	Xmin
	GridDots	Xtick
	GridLines	Xzoom
	Hmin	Ymax
	Hmax	Ymin
	Hwidth	Ytick
	Labels	Yzoom
	Recenter	
Numeric	D1	D6
	D2	D7
	D3	D8
	D4	D9
	D5	D0
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Results

NbItem

Contains the number of data points in the current 1-variable analysis (H1–H5).

MinVal

Contains the minimum value of the data set in the current 1-variable analysis (H1–H5).

Q1

Contains the value of the first quartile in the current 1-variable analysis (H1–H5).

MedVal

Contains the median in the current 1-variable analysis (H1–H5).

Q3

Contains the value of the third quartile in the current 1-variable analysis (H1–H5).

MaxVal

Contains the maximum value in the current 1-variable analysis (H1–H5).

ΣX

Contains the sum of the data set in the current 1-variable analysis (H1–H5).

ΣX2

Contains the sum of the squares of the data set in the current 1-variable analysis (H1–H5).

MeanX

Contains the mean of the data set in the current 1-variable analysis (H1–H5).

sX

Contains the sample standard deviation of the data set in the current 1-variable analysis (H1–H5).

σX

Contains the population standard deviation of the data set in the current 1-variable analysis (H1–H5).

serrX

Contains the standard error of the data set in the current 1-variable analysis (H1–H5).

ssX

Contains the sum of the squared deviations of x for the current statistical analysis (H1–H5).

Statistics 2Var app variables

Category	Names	
Results (explained below)	NbItem	σX
	Corr	serrX
	CoefDet	ssX
	sCov	MeanY
	σCov	ΣY
	ΣXY	ΣY2
	MeanX	sY
	ΣX	σY
	ΣX2	serrY
	sX	ssY
Symbolic	S1	S4
	S2	S5
	S3	
Plot	Axes	Xmin
	Cursor	Xtick

Category	Names	
	GridDots	Xzoom
	GridLines	Ymax
	Labels	Ymin
	Recenter	Ytick
	Xmax	Yzoom
Numeric	C1	C6
	C2	C7
	C3	C8
	C4	C9
	C5	C0
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Results

NbItem

Contains the number of data points in the current 2-variable analysis (S1–S5).

Corr

Contains the correlation coefficient from the latest calculation of summary statistics. This value is based on the linear fit only, regardless of the fit type chosen.

CoefDet

Contains the coefficient of determination from the latest calculation of summary statistics. This value is based on the fit type chosen.

sCov

Contains the sample covariance of the current 2-variable statistical analysis (S1–S5).

σCov

Contains the population covariance of the current 2-variable statistical analysis (S1–S5).

ΣXY

Contains the sum of the X·Y products for the current 2-variable statistical analysis (S1–S5).

MeanX

Contains the mean of the independent values (X) of the current 2-variable statistical analysis (S1–S5).

ΣX

Contains the sum of the independent values (X) of the current 2-variable statistical analysis (S1–S5).

ΣX2

Contains the sum of the squares of the independent values (X) of the current 2-variable statistical analysis (S1–S5).

sX

Contains the sample standard deviation of the independent values (X) of the current 2-variable statistical analysis (S1–S5).

σX

Contains the population standard deviation of the independent values (X) of the current 2-variable statistical analysis (S1–S5).

serrX

Contains the standard error of the independent values (X) of the current 2-variable statistical analysis (S1–S5).

ssX

Contains the sum of the squared deviations of x for the current statistical analysis (S1–S5).

MeanY

Contains the mean of the dependent values (Y) of the current 2-variable statistical analysis (S1–S5).

ΣY

Contains the sum of the dependent values (Y) of the current 2-variable statistical analysis (S1–S5).

ΣY2

Contains the sum of the squares of the dependent values (Y) of the current 2-variable statistical analysis (S1–S5).

sY

Contains the sample standard deviation of the dependent values (Y) of the current 2-variable statistical analysis (S1–S5).

σY

Contains the population standard deviation of the dependent values (Y) of the current 2-variable statistical analysis (S1–S5).

serrY

Contains the standard error of the dependent values (Y) of the current 2-variable statistical analysis (S1–S5).

ssY

Contains the sum of the squared deviations of y for the current statistical analysis (S1–S5).

Inference app variables

Category	Names	
Results (explained below)	ContribList	ContribMat
	Slope	Inter
	Corr	CoefDet
	serrLine	serrSlope
	serrInter	Yval
	serrY	CritScore
	Result	CritVal1
	TestScore	CritVal2
	TestValue	DF
	Prob	
Symbolic	AltHyp	InfType
	Method	
Numeric	Alpha	Pooled
	Conf	s1
	ExpList	s2
	Mean1	σ1
	Mean2	σ2
	n1	x1
	n2	x2
	μ0	Xlist
	π0	Ylist
	ObsList	Xval
	ObsMat	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Results

CoefDet

Contains the value of the coefficient of determination.

ContribList

Contains a list of the chi-square contributions by category for the chi-square goodness of fit test.

ContribMat

Contains a matrix of the chi-square contributions by category for the chi-square two-way test.

Corr

Contains the value of the correlation coefficient.

CritScore

Contains the value of the Z- or t-distribution associated with the input α-value

CritVal1

Contains the lower critical value of the experimental variable associated with the negative `TestScore` value which was calculated from the input α-level.

CritVal2

Contains the upper critical value of the experimental variable associated with the positive `TestScore` value which was calculated from the input α-level.

DF

Contains the degrees of freedom for the t-tests.

ExpList

Contains a list of the expected counts by category for the chi-square goodness of fit test.

ExpMat

Contains the matrix of expected counts by category for the chi-square two-way test.

Inter

Contains the value of the intercept of the regression line for either the linear t-test or the confidence interval for the intercept

Prob

Contains the probability associated with the `TestScore` value.

Result

For hypothesis tests, contains 0 or 1 to indicate rejection or failure to reject the null hypothesis.

serrInter

Contains the standard error of the intercept for either the linear t-test or the confidence interval for the intercept.

serrLine

Contains the standard error of the line for the linear t-test.

serrSlope

Contains the standard error of the slope for either the linear t-test or the confidence interval for slope.

serrY

Contains the standard error of ŷ for either the confidence interval for a mean response or the prediction interval for a future response.

Slope

Contains the value of the slope of the regression line for either the linear t-test or the confidence interval for slope.

TestScore

Contains the Z- or t-distribution value calculated from the hypothesis test or confidence interval inputs.

TestValue

Contains the value of the experimental variable associated with the `TestScore`.

Yval

Contains the value of ŷ for either the confidence interval for a mean response or the prediction interval for a future response.

Parametric app variables

Category	Names	
Symbolic	X1	X6
	Y1	Y6
	X2	X7
	Y2	Y7
	X3	X8
	Y3	Y8
	X4	X9
	Y4	Y9
	X5	X0
	Y5	Y0
Plot	Axes	Tstep
	Cursor	Xmax

Category	Names	
	GridDots	Xmin
	GridLines	Xtick
	Labels	Xzoom
	Method	Ymax
	Recenter	Ymin
	Tmin	Ytick
	Tmax	Yzoom
Numeric	NumStart	NumType
	NumStep	NumZoom
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Polar app variables

Category	Names	
Symbolic	R1	R6
	R2	R7
	R3	R8
	R4	R9
	R5	R0
Plot	θmin	Recenter
	θmax	Xmax
	θstep	Xmin
	Axes	Xtick
	Cursor	Xzoom
	GridDots	Ymax
	GridLines	Ymin
	Labels	Ytick
	Method	Yzoom
Numeric	NumStart	NumType
	NumStep	NumZoom
Modes	AAngle	AComplex

Category	Names	
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Finance app variables

Category	Names	
Numeric	CPYR	NbPmt
	BEG	PMT
	FV	PPYR
	IPYR	PV
	GSize	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Linear Solver app variables

Category	Names	
Numeric	LSystem	LSolution[a]
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

[a] Contains a vector with the last solution found by the Linear Solver app.

Triangle Solver app variables

Category	Names	
Numeric	SideA	AngleA
	SideB	AngleB
	SideC	AngleC
	TriType	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Linear Explorer app variables

Category	Names	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Quadratic Explorer app variables

Category	Names	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

Trig Explorer app variables

Category	Names	
Modes	AAngle	AComplex

Category	Names	
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

[a2] Contains a vector with the last solution found by the Linear Solver app.

Sequence app variables

Category	Names	
Symbolic	U1	U6
	U2	U7
	U3	U8
	U4	U9
	U5	U0
Plot	Axes	Xmax
	Cursor	Xmin
	GridDots	Xtick
	GridLines	Xzoom
	Labels	Ymax
	Nmin	Ymin
	Nmax	Ytick
	Recenter	Yzoom
Numeric	NumIndep	NumType
	NumStart	NumZoom
	NumStep	
Modes	AAngle	AComplex
	ADigits	AFiles
	AFilesB	AFormat
	ANote	AProgram
	AVars	DelAFiles
	DelAVars	

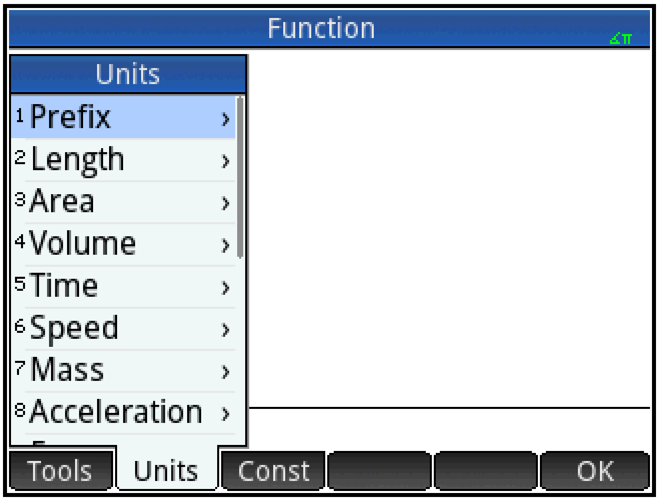
24 Units and constants

Units

A unit of measurement—such as inch, ohm, or Becquerel—enables you to give a precise magnitude to a physical quantity.

You can attach a unit of measurement to any number or numerical result. A numerical value with units attached is referred to as a measurement. You can operate on measurements just as you do on numbers without attached units. The units are kept with the numbers in subsequent operations.

The units are on the **Units** menu. Press Shift Units (Units) and, if necessary, tap Units.



The menu is organized by category. Each category is listed at the left, with the units in the selected category listed at the right.

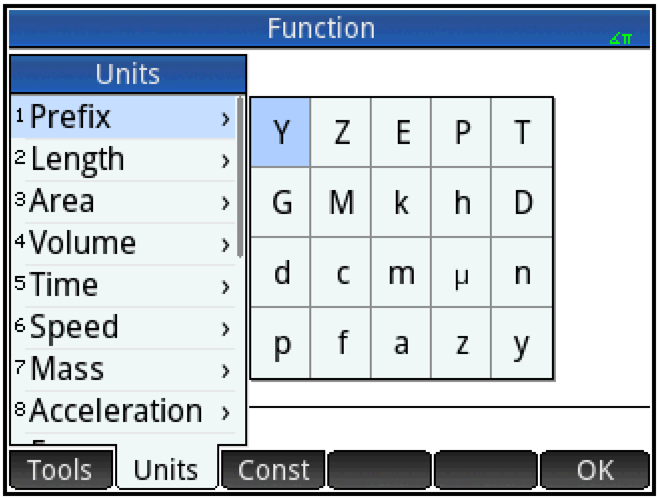
Unit categories

- length
- area
- volume
- time
- speed
- mass
- acceleration
- force
- energy
- power
- pressure

- temperature
- electricity
- light
- angle
- viscosity
- radiation

Prefixes

The **Units** menu includes an entry that is not a unit category, namely, Prefix. Selecting this option displays a palette of prefixes.



Y: yotta	Z: zetta	E: exa	P: peta	T: tera
G: giga	M: mega	k: kilo	h: hecto	D: deca
d: deci	c: centi	m: milli	μ: micro	n: nano
p: pico	f: femto	a: atto	z: zepto	y: octo

Unit prefixes provide a handy way of entering large or small numbers. For example, the speed of light is approximately 300,000 m/s. If you wanted to use that in a calculation, you could enter it as 300_km/s, with the prefix k selected from the prefix palette.

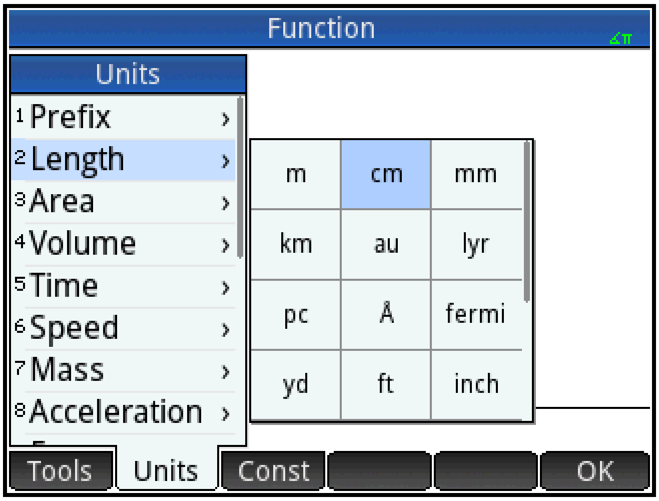
Select the prefix you want before selecting the unit.

Unit calculations

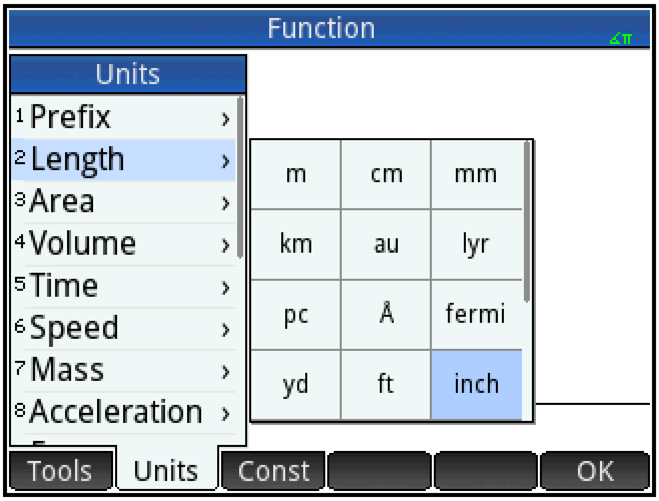
A number plus a unit is a measurement. You can perform calculations with multiple measurements providing that the units of each measurement are from the same category. For example, you can add two measurements of length (even lengths of different units, as illustrated in the following example). But you cannot add, say, a length measurement to a volume measurement.

Suppose you want to add 20 centimeters and 5 inches and have the result displayed in centimeters.

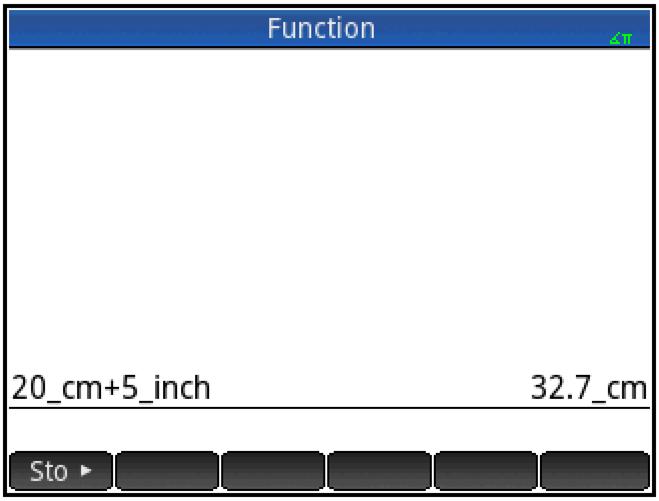
1. If you want the result in cm, enter the centimeter measurement first. 20 Shift Units (Units). Select **Length**. Select **cm**.



2. Now add 5 inches. + 5 Shift Units . Select **Length**. Select **in** Enter

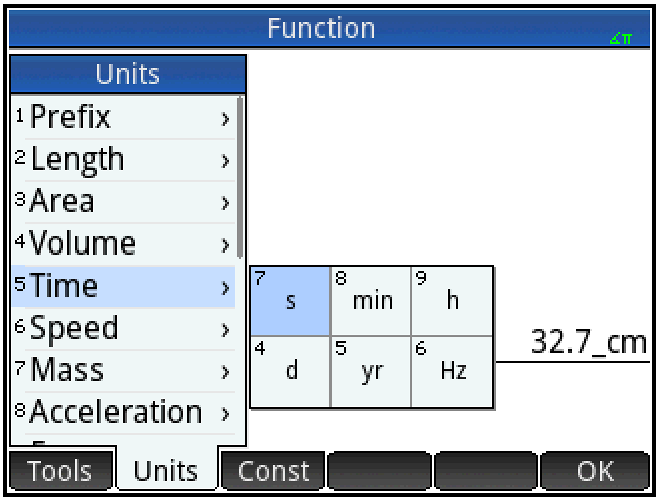


The result is shown as 32.7 cm. If you had wanted the result in inches, then you would have entered the 5 inches first.

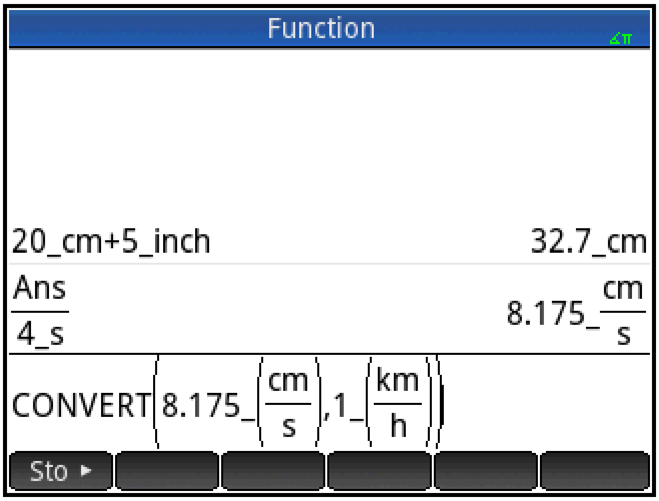


```
20_cm+5_inch                32.7_cm
```

3. To continue the example, let's divide the result by 4 seconds. ÷ 4 Shift Units. Select **Time**. Select **s** Enter.

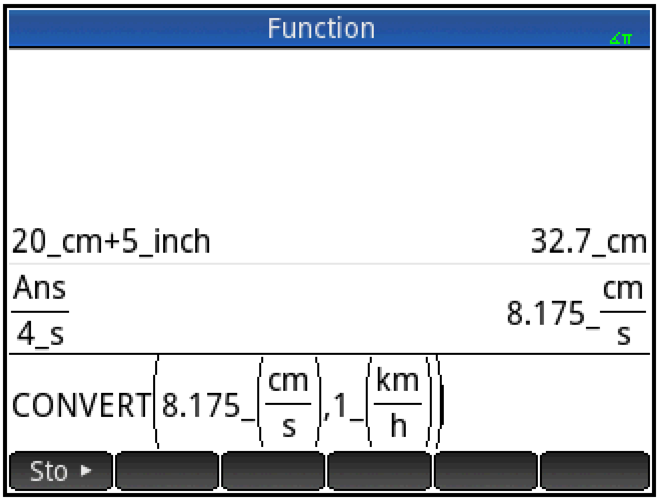


The result is shown as 8.175 cm*s^{-1}.



```
20_cm+5_inch                          32.7_cm
Ans                                        cm
———                                8.175_ ——
4_s                                         s
CONVERT(8.175_(cm/s),1_(km/h))
```

4. Now convert the result to kilometers per hour. Sto ▸ Shift Units. Select **Speed**. Select **km/h** Enter ≈.



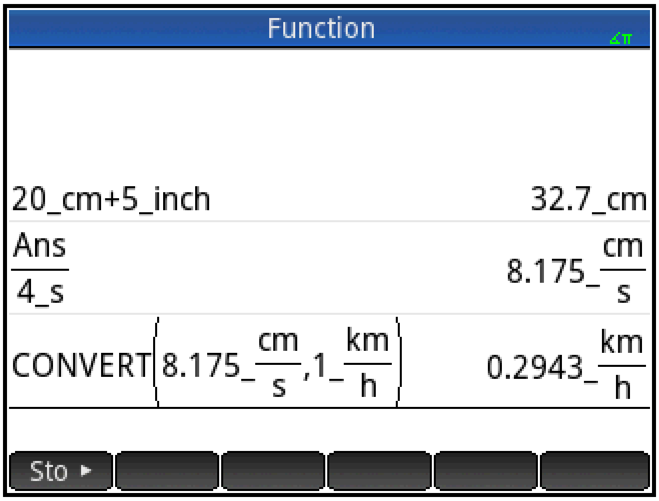
Function

20_cm+5_inch 32.7_cm

$\frac{\text{Ans}}{4\_s}$ $8.175\_\frac{cm}{s}$

$\text{CONVERT}\left(8.175\_\left(\frac{cm}{s}\right),1\_\left(\frac{km}{h}\right)\right)$

The result is shown as 0.2943 kilometers per hour.



Function

20_cm+5_inch 32.7_cm

$\frac{\text{Ans}}{4\_s}$ $8.175\_\frac{cm}{s}$

$\text{CONVERT}\left(8.175\_\frac{cm}{s},1\_\frac{km}{h}\right)$ $0.2943\_\frac{km}{h}$

This shortcut does not work in CAS view.

Unit tools

There are a number of tools for managing and operating on units. These are available by pressing Shift Units and tapping Tools.

Convert

Converts one unit to another unit of the same category.

`CONVERT(5_m,1_ft)` returns `16.4041994751_ft`

You can also use the last answer as the first argument in a new conversion calculation. Pressing Shift + Ans places the last answer on the entry line. You can also select a value from history and tap Copy to copy it to the entry line. Sto ► with a measurement calls the convert command as well and converts to whatever unit follows the Store symbol.

The Convert tool also converts bases on either single values or arrays of values.

`convert(123,base,8)` returns `[3, 7, 1]`

This result means that 123 in decimal notation equals 173 in octal notation, because the result always reverses the digits.

`convert([3, 7, 1],base,8)` returns `123`

The Convert tool can also be used to convert real numbers or ratios to continued fractions.

Example:

`convert(pi,confrac)` returns `[3,7,15,1,292,1,1,1,2]`

MKSA

Meters, kilograms, seconds, amperes. Converts a complex unit into the base components of the MKSA system.

`MKSA(8.175_cm/s)` returns `.08175_m/s`

UFACTOR

Unit factor conversion. Converts a measurement using a compound unit into a measurement expressed in constituent units. For example, a Coulomb—a measure of electric charge—is a compound unit derived from the SI base units of Ampere and second: 1 C = 1 A * 1 s. Thus:

`UFACTOR(100_C,1_A))` returns `100_A*s`

USIMPLIFY

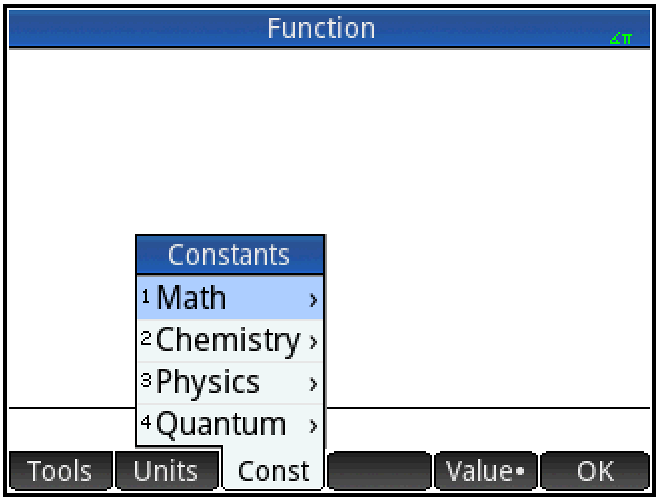
Unit simplification. For example, a Joule is defined as one kg*m^2/s^2. Thus:

`USIMPLIFY(5_kg*m^2/s^2)` returns `5_J`

Physical constants

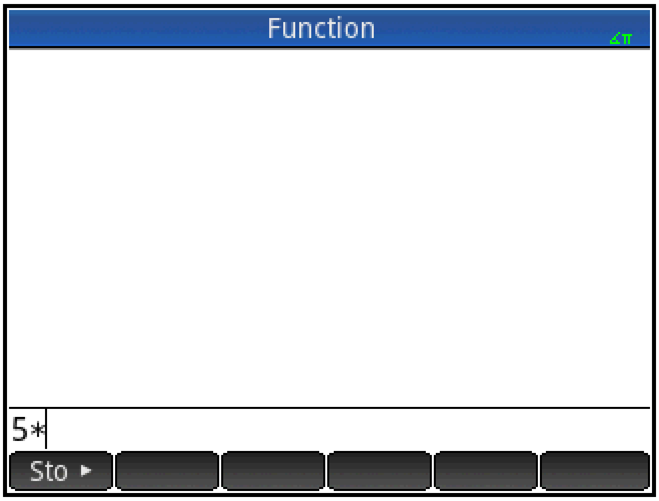
The values of 34 math and physical constants can be selected (by name or value) and used in calculations. These constants are grouped into four categories: math, chemistry, physics and quantum mechanics. A list of all these constants is given in List of constants on page 499.

To display the constants, press Shift Units and then tap Const.



Suppose you want to know the potential energy of a mass of 5 units according to the equation *E = mc2*.

1. Enter the mass and the multiplication operator: 5 ×



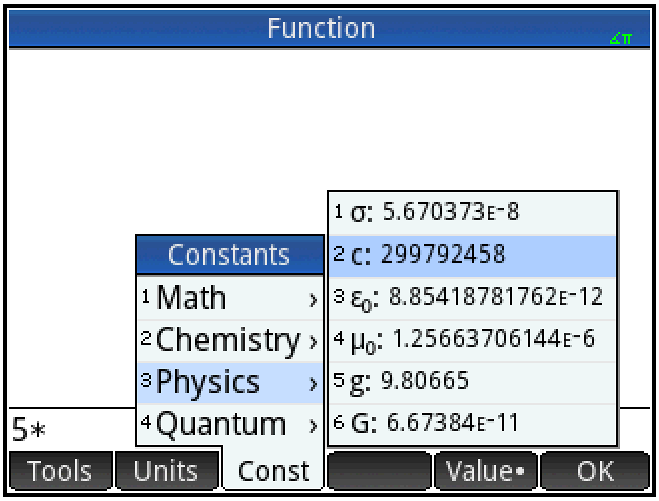
5*

2. Open the constants menu.

Shift Units Const

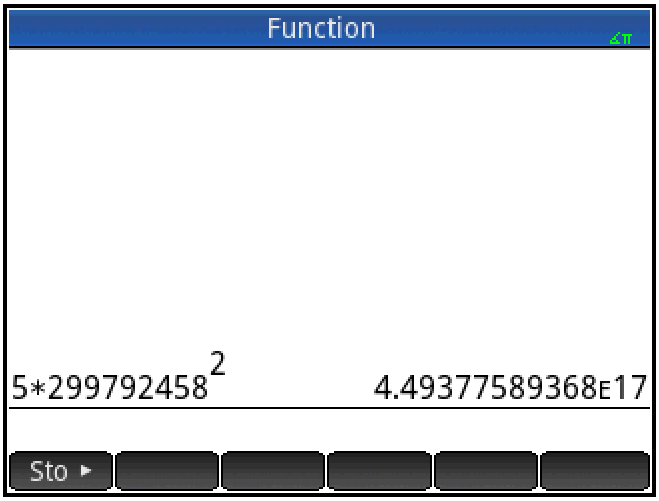
3. Select **Physics**.

4. Select **c: 299792458**.



5*

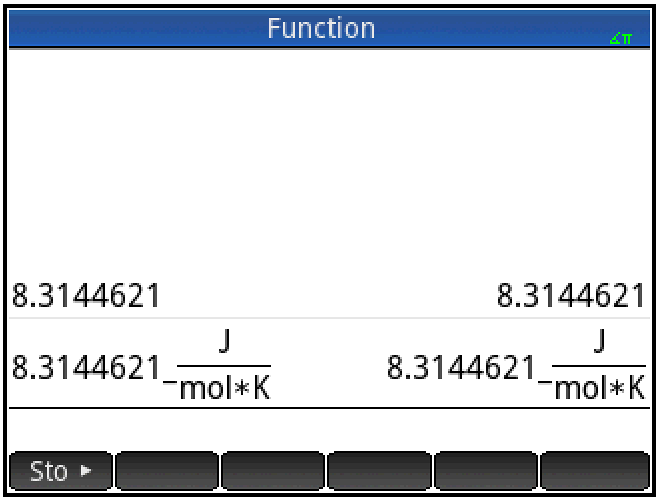
5. Square the speed of light and evaluate the expression. x² Enter



```
5*299792458²          4.49377589368E17
```

You can enter just the value of a constant or the constant and its units (if it has units). If Value• is showing on the screen, the value is inserted at the cursor point. If Value is showing on the screen, the value and its units are inserted at the cursor point.

In the previous figure, the first entry shows the Universal Gas Constant after it was chosen with Value• showing. The second entry shows the same constant, but chosen when Value was showing.



```
8.3144621                              8.3144621
8.3144621_J/(mol*K)          8.3144621_J/(mol*K)
```

Tapping Value displays Value•, and vice versa.

List of constants

Category	Name and symbol
Math	e
	MAXREAL
	MINREAL
	π
	I
Chemistry	Avogadro, NA
	Boltmann, k
	molar volume, Vm
	universal gas, R
	standard temperature, StdT
	standard pressure, StdP
Physics	Stefan-Boltzmann, σ
	speed of light, c
	permittivity, ϵ_0
	permeability, μ_0
	acceleration of gravity, g
	gravitation, G
Quantum	Planck, h
	Dirac,Ћ
	electronic charge, q
	electron mass, me
	q/me ratio, qme

Category	Name and symbol
	proton mass, mp
	mp/me ratio, mpme
	fine structure, α
	magnetic flux, ϕ
	Faraday, F
	Rydberg, R_∞
	Bohr radius, a_0
	Bohr magneton, μ
	nuclear magneton, μ_N
	photon wavelength, λ_0
	photon frequency, f_0
	Compton wavelength, λ_c

25 Lists

A list consists of comma-separated real or complex numbers, expressions, or matrices, all enclosed in braces. A list may, for example, contain a sequence of real numbers such as `{1,2,3}`. Lists represent a convenient way to group related objects.

You can do list operations in Home and in programs.

There are ten list variables available, named `L0` to `L9`, or you can create your own list variable names. You can use them in calculations or expressions in Home or in a program. Retrieve a list name from the Vars menu (Vars), or just type its name from the keyboard.

You can create, edit, delete, send, and receive named lists in the List Catalog: Shift 7 List Q (List). You can also create and store lists—named or unnamed—in Home view.

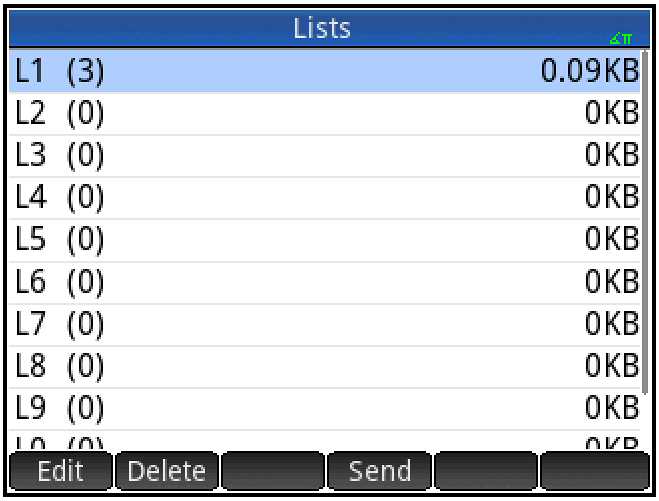
List variables are identical in behavior to the columns `C1`–`C0` in the Statistics 2Var app and the columns `D1`–`D0` in the Statistics 1Var app. You can store a statistics column as a list (or vice versa) and use any of the list functions on the statistics columns, or the statistics functions on the list variables.

Creating a list in the List Catalog

1. Open the List Catalog.

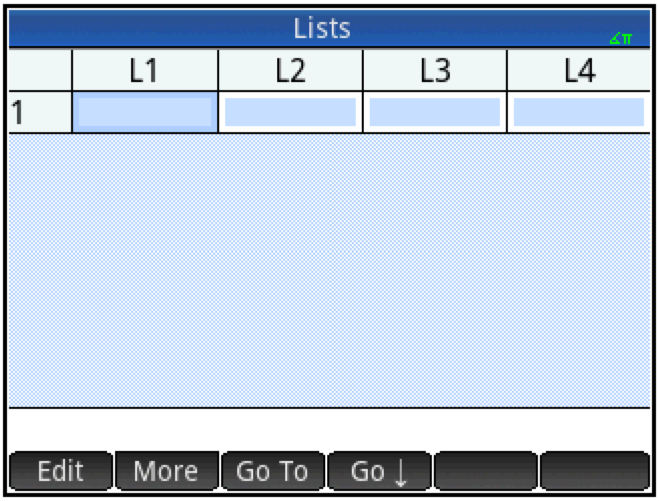
   Shift 7 List Q (List)

   The number of elements in a list is shown beside the list name.



```
Lists
L1  (3)        0.09KB
L2  (0)           0KB
L3  (0)           0KB
L4  (0)           0KB
L5  (0)           0KB
L6  (0)           0KB
L7  (0)           0KB
L8  (0)           0KB
L9  (0)           0KB
L0  (0)           0KB
```

2. Tap on the name you want to assign to the new list (`L1`, `L2`, etc.). The list editor appears.

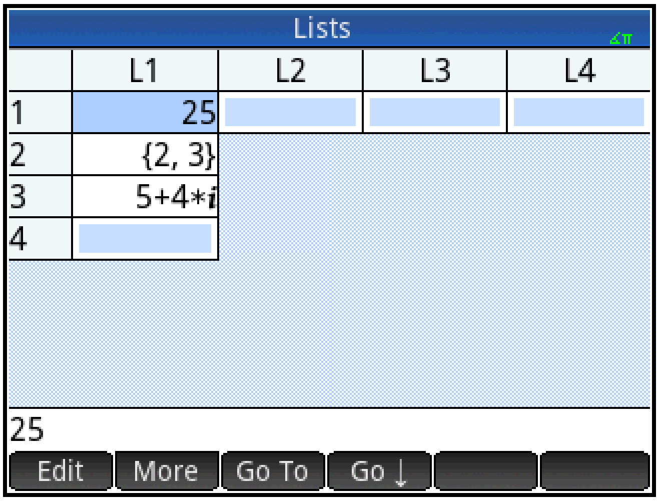


	L1	L2	L3	L4
1				

If you are creating a new list rather than editing a list that already contains elements, make sure you choose a list without any elements in it.

3. Enter the values you want in the list, pressing Enter after each one.

Values can be real or complex numbers (or an expression). If you enter an expression, it is evaluated and the result is inserted in the list.



	L1	L2	L3	L4
1	25			
2	{2, 3}			
3	5+4*i			
4				

25

4. When done, press Shift 7 List (List) to return to the List catalog, or press ⌂ Settings to go to Home view.

The buttons and keys in the List Catalog are:

Button or Key	Purpose
Edit	Opens the highlighted list for editing. You can also just tap on a list name.
Delete or Del	Deletes the contents of the selected list.

Button or Key	Purpose
Send	Transmits the highlighted list to another HP Prime calculator, if available.
Shift Esc Clear (Clear)	Clears all lists.
Shift ▲ or ▼	Moves to the top or bottom of the catalog, respectively.

The List Editor

The List Editor is a special environment for entering data into lists. There are two ways to open the List Editor once the List Catalog is open:

- Highlight the list and tap Edit or
- Tap the name of the list.

List Editor: Buttons and keys

When you open a list, the following buttons and keys are available to you:

Button or Key	Purpose
Edit	Copies the highlighted list item into the entry line.
More	Opens a menu with options for editing the list.
Go To	Moves the cursor to the specified element in the list. This option is especially useful for very large lists.
Go	Sets how the cursor moves after you press Enter ≈. The options are **Down**, **Right**, and **None**.
Shift Esc Clear (Clear)	Clears all items from the list.
Shift ▲ or ▼	Moves the cursor to the start or the end of the list.

List Editor: More menu

The List Editor More menu contains options for editing a list. These options are described in the following table.

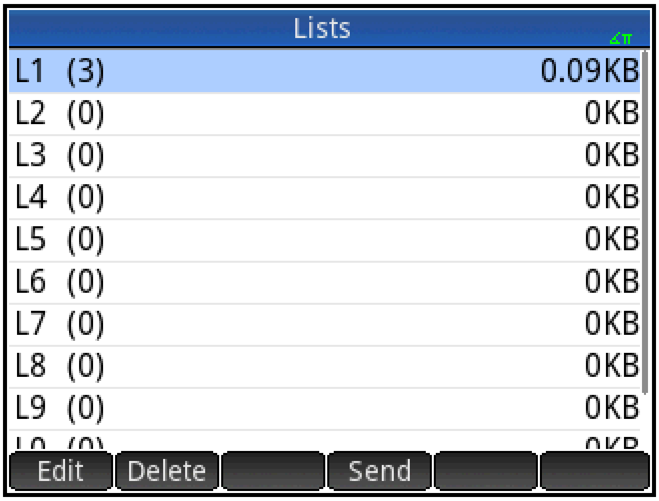
Category	Option	Description
Insert	Row	Inserts a new row above the current row in the list. The new row contains a zero.

Category	Option	Description
Delete	Column	Deletes the contents of the current list (column). To delete a single element, select it and then press Del .
Select	Row	Selects the current row. After it is selected, the row can be copied.
	Column	Selects the current column. After it is selected, the column can be copied.
	Box	Opens a dialog box to select a rectangular array defined by a starting location and a final location. You can also tap and hold a cell to select it as the starting location, and then drag your finger to select a rectangular array of elements. After it is selected, the rectangular array can be copied.
Selection		Turns selection mode on and off. You can also tap and hold a cell and then drag your finger to select multiple cells.
Swap	Column	Transposes the values of the selected columns.

Editing a list

1. Open the List Catalog.

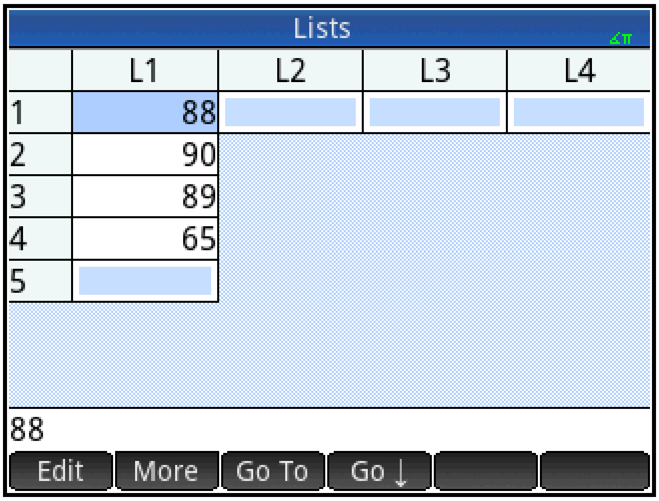
   Shift 7 (List)



Lists

```
L1  (3)     0.09KB
L2  (0)        0KB
L3  (0)        0KB
L4  (0)        0KB
L5  (0)        0KB
L6  (0)        0KB
L7  (0)        0KB
L8  (0)        0KB
L9  (0)        0KB
L0  (0)        0KB
```

2. Tap the name of the list (**L1**, **L2**, and so on). The List Editor appears.



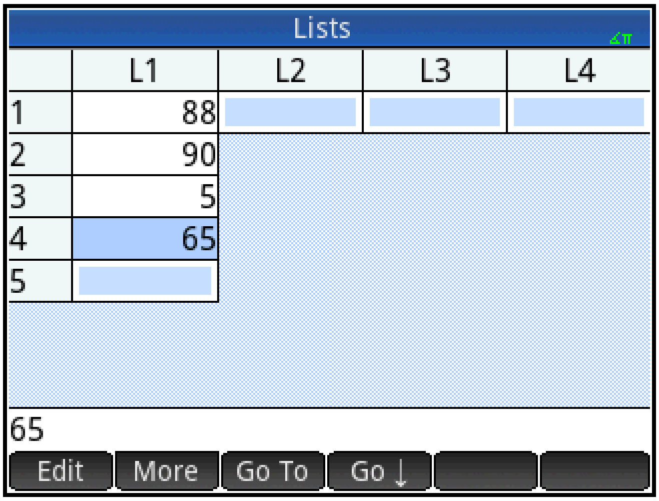
Lists

	L1	L2	L3	L4
1	88			
2	90			
3	89			
4	65			
5				

88

3. Tap the element you want to edit. (Alternatively, press ▲ or ▼ until the element you want to edit is highlighted.) In this example, edit the third element so that it has a value of 5.

5 OK



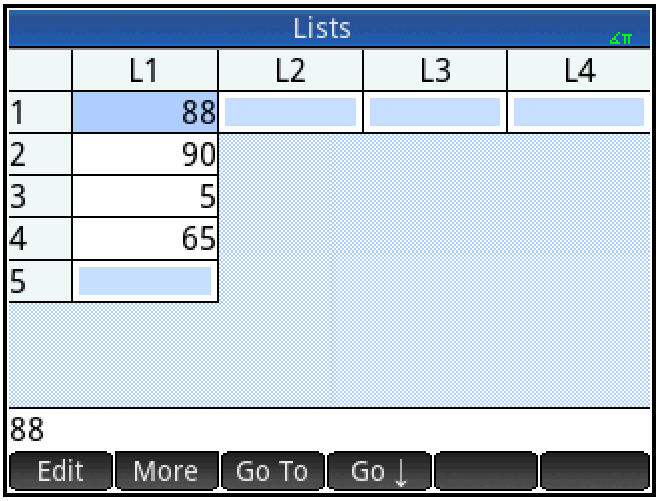
Lists

	L1	L2	L3	L4
1	88			
2	90			
3	5			
4	65			
5				

65

Inserting an element in a list

Suppose you want to insert a new value, 9, in L1(2) in the list L1 shown in the following figure.


Lists

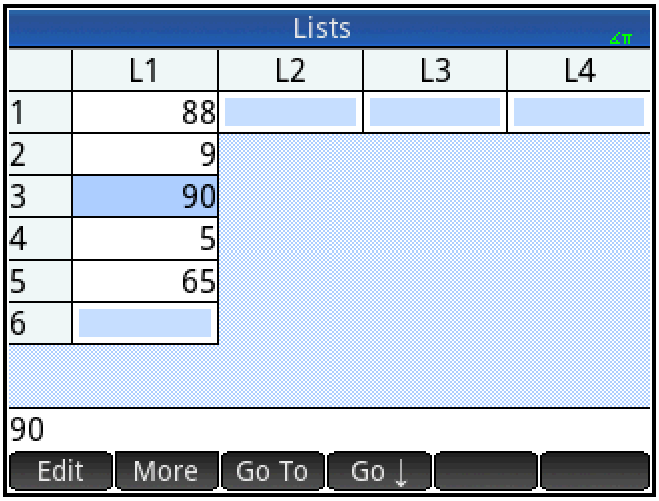
	L1	L2	L3	L4
1	88			
2	90			
3	5			
4	65			
5				

88

1. Select L1(2); that is, select the second element in the list.

2. Tap More, select **Insert**, and then select **Row**.

3. Enter 9 and then tap OK.


Lists

	L1	L2	L3	L4
1	88			
2	9			
3	90			
4	5			
5	65			
6				

90

Deleting lists

To delete a list

In the List Catalog, use the cursor keys to highlight the list and press Del. You are prompted to confirm your decision. Tap OK or press Enter ≈.

If the list is one of the reserved lists L0-L9, then only the contents of the list are deleted. The list is simply stripped of its contents. If the list is one you have named (other than L0-L9), then it is deleted entirely.

To delete all lists

In the List Catalog, press Shift Esc Clear (Clear).

The contents of the lists L0-L9 are deleted and any other named lists are deleted entirely.

Lists in Home view

You can enter and operate on lists directly in Home view. The lists can be named or unnamed.

To create a list

1. Press Shift 8 ({}).

   A pair of braces appears on the entry line. All lists must be enclosed in braces.

2. Enter the first element in the list followed by a comma: [element] ,

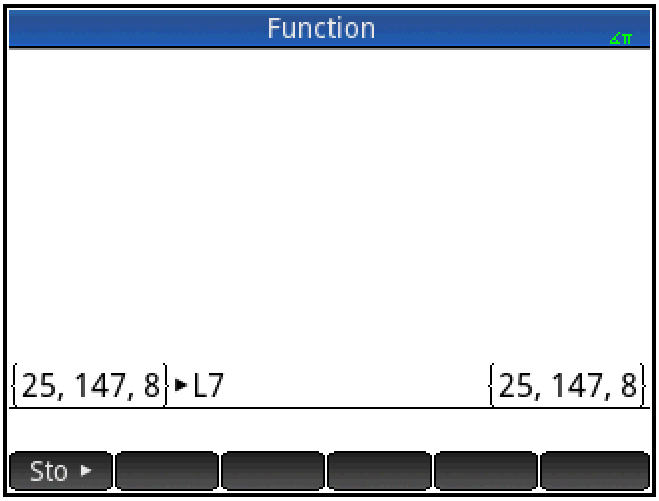
3. Continue adding elements, separating each with a comma.

4. When you have finished entering the elements, press Enter. The list is added to History (with any expressions among the elements evaluated).

To store a list

You can store a list in a variable. You can do this before the list is added to History, or you can copy it from History. When you've entered a list in the entry line or copied it from History to the entry line, tap Sto ▸, enter a name for the list and press Enter. The reserved list variable names available to you are `L0` through `L9`; however, you can create a list variable name of your own as well.

For example, to store the list {25,147,8} in `L7`:



```
Function
{25, 147, 8}▸L7                    {25, 147, 8}
```

1. Create the list on the entry line.
2. Press ▶ to move the cursor outside the list.
3. Tap Sto ▸.
4. Enter the name:

   ALPHA x^2 7

5. Complete the operation: Enter.

To display a list

To display a list in Home view, type its name and press Enter.

If the list is empty, a pair of empty braces is returned.

To display one element

To display one element of a list in Home view, enter *listname (element#)*. For example, if L6 is {3,4,5,6}, then `L6(2)` Enter returns 4.

To store one element

To store a value in one element of a list in Home view, enter *value* Sto ▸ *listname (element#)*. For example, to store 148 as the second element in L2, type `148` Sto ▸ `L2(2)` Enter.

List references

Suppose L1:={5, "abcde", {1,2,3,4,5}, 11}. `L1(1)` returns 5 and `L1(2)` returns "abcde". `L1(2, 4)` returns 100 (the ASCII code for d) and `L1(2,4,1)` returns "d". `L1({2,4})` returns {"abcde", {1,2,3,4,5},11}, extracting a sublist of all the elements from 2 through 4.

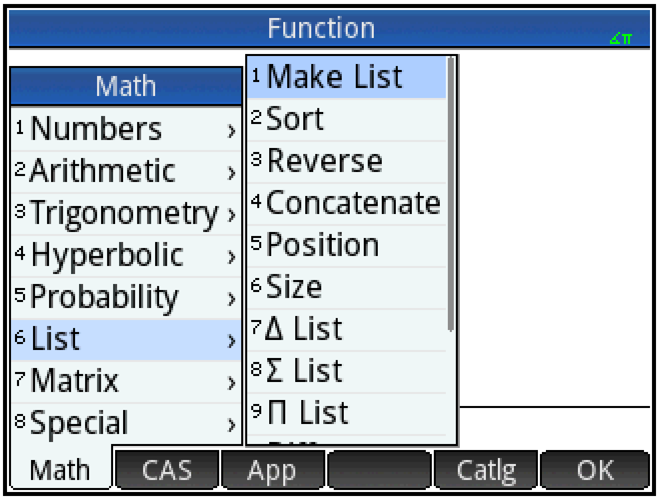
To send a list

You can send lists to another calculator or a PC just as you can apps, programs, matrices, and notes.

List functions

List functions are found on the Math menu. You can use them in Home and in programs.

You can type in the name of the function, or you can copy the name of the function from the List category of the Math menu.



Press Mem B 6 to select the **List** category in the left column of the **Math** menu. (**List** is the sixth category on the **Math** menu, which is why pressing 6 will take you straight to the **List** category.) Tap a function to select it, or use the direction keys to highlight it and either tap OK or press Enter ≈ .

List functions are enclosed in parentheses. They have arguments that are separated by commas, as in `CONCAT(L1,L2)`. An argument can be either a list variable name or the actual list; for example, `REVERSE(L1)` or `REVERSE({1,2,3})`.

Common operators like +, –, ×, and ÷ can take lists as arguments. If there are two arguments and both are lists, then the lists must have the same length, since the calculation pairs the elements. If there are two arguments and one is a real number, then the calculation operates on each element of the list.

Example:

`5*{1,2,3}` returns `{5,10,15}`.

Besides the common operators that can take numbers, matrices, or lists as arguments, there are commands that can only operate on lists.

Menu format

By default, a List function is presented on the Math menu using its descriptive name, not its common command name. Thus the shorthand name `CONCAT` is presented as **Concatenate** and `POS` is presented as **Position**.

If you prefer the **Math** menu to show command names instead, deselect the **Menu Display** option on page 2 of the Home Settings screen.

Difference

Returns the list of non-common elements of two lists.

`DIFFERENCE({1,2,3,4}, {1,3,5,7})` returns {2,4,5,7}

Intersect

Returns the list of the elements common to two lists.

`INTERSECT({1,2,3,4}, {1,3,5,7})` returns {1,3}

Make List

Calculates a sequence of elements for a new list using the syntax:

```
MAKELIST(expression,variable,begin,end,increment)
```

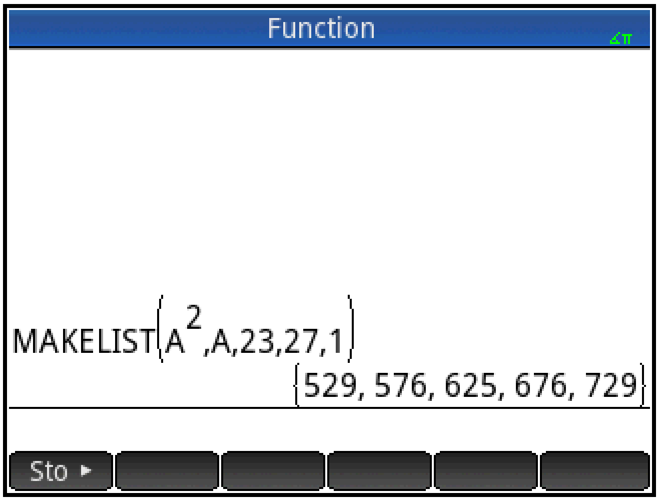
Evaluates *expression* with respect to *variable*, as *variable* takes on values from *begin* to *end* values, taken at increment steps.

Example:

In Home, generate a series of squares from 23 to 27:

Mem Select **List**. Select **Make List** (or **MAKELIST**) ALPHA Vars x² , ALPHA Vars

, 23 , 27 , 1 Enter



Function

MAKELIST(A^2,A,23,27,1)

{529, 576, 625, 676, 729}

Sort

Sorts the elements in a list in ascending order.

```
SORT(list)
```

Example:

`SORT({2,5,3})` returns `{2,3,5}`

Reverse

Creates a list by reversing the order of the elements in a list.

`REVERSE(list)`

Example:

`REVERSE({1,2,3})` returns `{3,2,1}`

Concatenate

Concatenates two lists into a new list.

`CONCAT(list1,list2)`

Example:

`CONCAT({1,2,3},{4})` returns `{1,2,3,4}`.

Position

Returns the position of an element within a list. The element can be a value, a variable, or an expression. If there is more than one instance of the element, the position of the first occurrence is returned. A value of 0 is returned if there is no occurrence of the specified element.

`POS(list, element)`

Example:

`POS ({3,7,12,19},12)` returns `3`

Size

Returns the number of elements in a list or a list containing the dimensions of a vector or matrix.

`SIZE(list)` or `SIZE(Vector)` or `SIZE(Matrix)`

Examples:

`SIZE({1,2,3})` returns `3`

`SIZE([[1 2 3], [4 5 6]])` returns `{2, 3}`

ΔLIST

Creates a new list composed of the first differences of a list; that is, the differences between consecutive elements in the list. The new list has one less element than the original list. The differences for {x_1, x_2, x_3,... x_{n-1}, x_n} are {$x_2–x_1$, $x_3–x_2$,... $x_n–x_{n–1}$}.

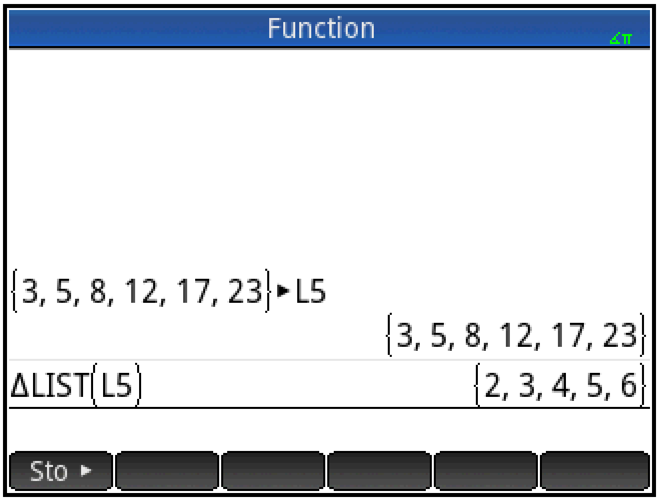
```
ΔLIST(list1)
```

Example:

In Home view, store {3,5,8,12,17,23} in L5 and find the first differences for the list.

Shift 8 3,5,8,12,17,23 ▶ Sto ▸ ALPHA x² 5 Enter Mem. Select **List**.

Select **ΔListA**. ALPHA x² 5 Enter



Function

```
{3, 5, 8, 12, 17, 23}▸L5
                          {3, 5, 8, 12, 17, 23}
ΔLIST(L5)                       {2, 3, 4, 5, 6}
```

ΣLIST

Calculates the sum of all elements in a list.

```
ΣLIST(list)
```

Example:

`ΣLIST({2,3,4})` returns `9`.

πLIST

Calculates the product of all elements in list.

```
πLIST(list)
```

Example:

`πLIST({2,3,4})` returns `24`.

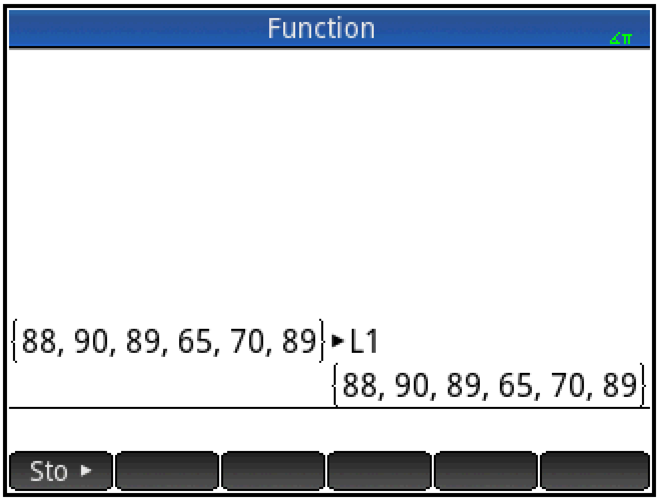
Finding statistical values for lists

To find statistical values—such as the mean, median, maximum, and minimum of a list—you create a list, store it in a data set and then use the Statistics 1Var app.

In this example, use the Statistics 1Var app to find the mean, median, maximum, and minimum values of the elements in the list L1, being 88, 90, 89, 65, 70, and 89.

1. In Home view, create L1.

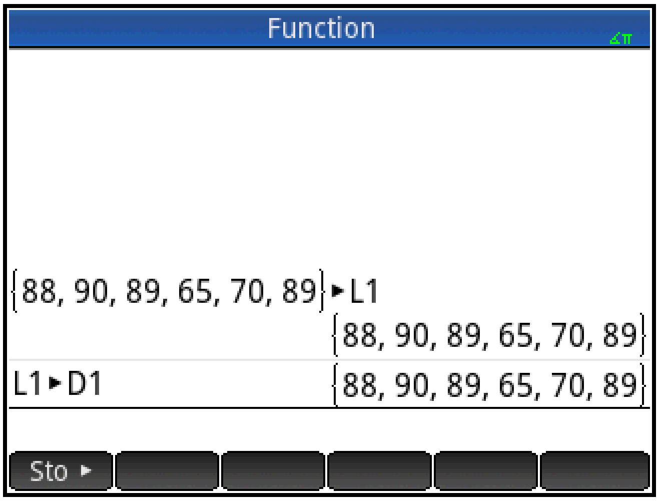
   Shift 8 88, 90, 89, 65, 70,89 ▶ Sto ▸ ALPHA x^2 1 Enter

   ```
   Function
   {88, 90, 89, 65, 70, 89}▸L1
                                {88, 90, 89, 65, 70, 89}
   ```

2. In Home view, store L1 in D1.

   ALPHA x^2 1 Sto ▸ ALPHA $xt\theta n$ 1 Enter

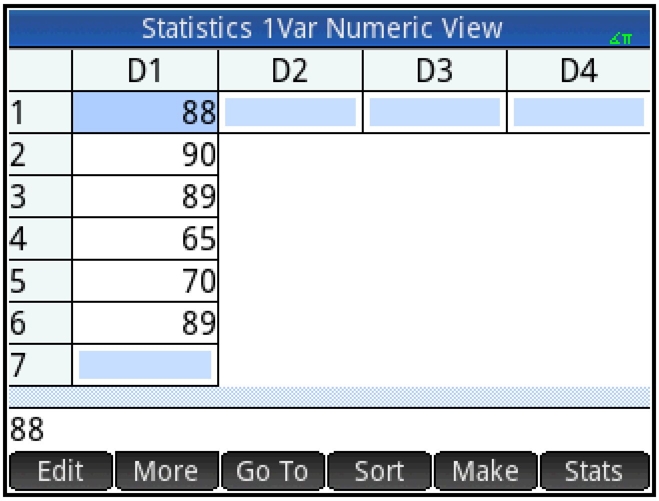
   

   ```
   Function
   {88, 90, 89, 65, 70, 89}▸L1
                                {88, 90, 89, 65, 70, 89}
   L1▸D1                        {88, 90, 89, 65, 70, 89}
   ```

   You will now be able to see the list data in the Numeric view of the Statistics 1Var app.

3. Start the Statistics 1Var app.

Apps Info Select **Statistics 1VarNotice** that your list elements are in data set D1.



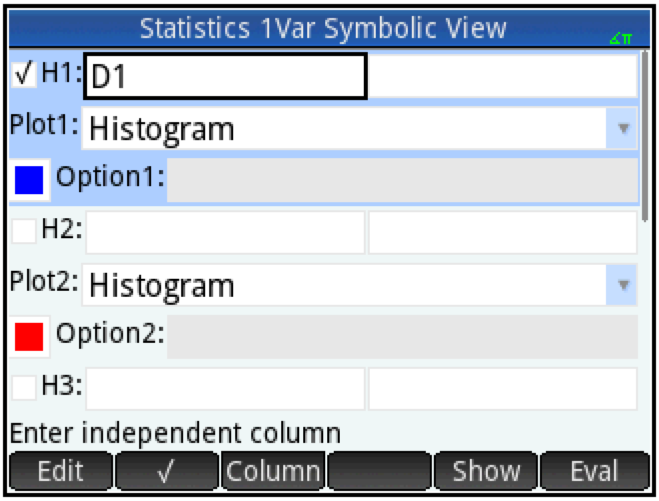
Statistics 1Var Numeric View

	D1	D2	D3	D4
1	88			
2	90			
3	89			
4	65			
5	70			
6	89			
7				

88

4. In the Symbolic view, specify the data set whose statistics you want to find.

Symb Setup



Statistics 1Var Symbolic View

✓ H1: D1

Plot1: Histogram

Option1:

□ H2:

Plot2: Histogram

Option2:

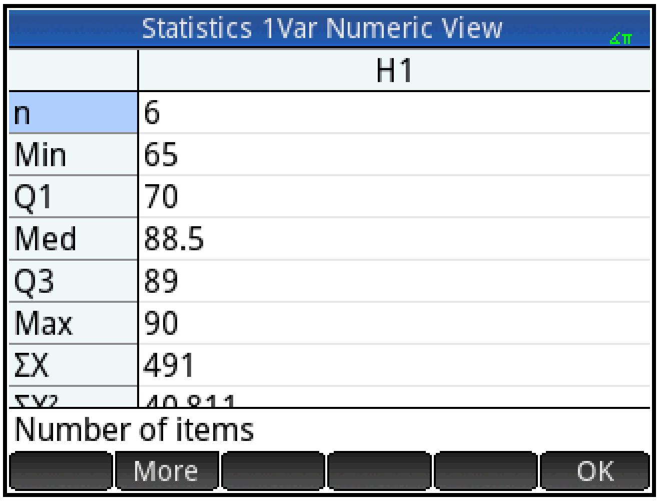
□ H3:

Enter independent column

By default, H1 will use the data in D1, so nothing further needs to be done in Symbolic view. However, if the data of interest were in D2, or any column other than D1, you would have to specify the desired data column here.

5. Calculate the statistics.

Num Setup Stats



Statistics 1Var Numeric View

	H1
n	6
Min	65
Q1	70
Med	88.5
Q3	89
Max	90
ΣX	491
ΣX²	40,811

Number of items

6. Tap OK when you are done.

26 Matrices

You can create, edit, and operate on matrices and vectors in the Home view, CAS, or in programs. You can enter matrices directly in Home or CAS, or use the Matrix Editor.

Vectors

Vectors are one-dimensional arrays. They are composed of just one row. A vector is represented by single brackets; for example, [1 2 3]. A vector can be a real number vector, or a complex number vector such as[1+2*i 7+3*i].

Matrices

Matrices are two-dimensional arrays. They are composed of at least two rows and at least one column. Matrices may contain any combination of or real and complex numbers, such as:

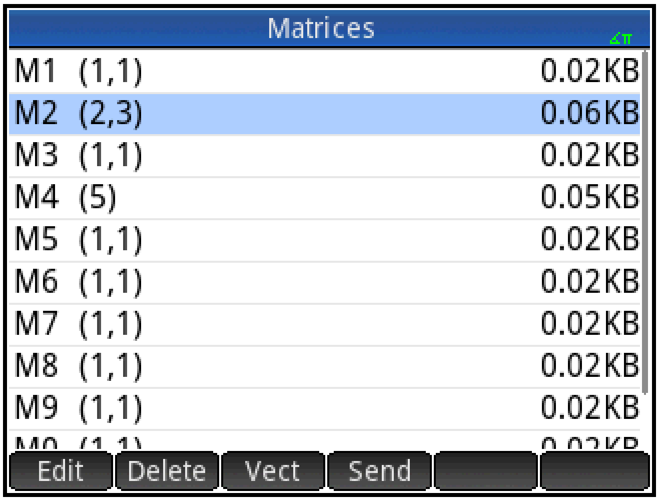
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ or $\begin{bmatrix} 1+2i \\ 3-4i \\ 7 \end{bmatrix}$

Matrix variables

There are ten reserved matrix variables available, named `M0` to `M9`; however, you can save a matrix in a variable name you define. You can then use them in calculations in Home or CAS views or in a program. You can retrieve matrix names from the Vars menu, or just type their names from the keyboard.

Creating and storing matrices

The Matrix Catalog contains the reserved matrix variables M0-M9, as well as any matrix variables you have created in Home or CAS views (or from a program if they are global).



Matrices

```
M1  (1,1)    0.02KB
M2  (2,3)    0.06KB
M3  (1,1)    0.02KB
M4  (5)      0.05KB
M5  (1,1)    0.02KB
M6  (1,1)    0.02KB
M7  (1,1)    0.02KB
M8  (1,1)    0.02KB
M9  (1,1)    0.02KB
M0  (1,1)    0.02KB
```

Once you select a matrix name, you can create, edit, and delete matrices in the Matrix Editor. You can also send a matrix to another HP Prime.

To open the Matrix Catalog, press Shift 4 (Matrix).

In the Matrix Catalog, the size of a matrix is shown beside the matrix name. (An empty matrix is shown as 1*1.) The number of elements in it is shown beside a vector.

You can also create and store matrices—named or unnamed—in Home view. For example, the command:

```
POLYROOT([1,0,-1,0])▶M1
```

stores the roots of the complex vector of length 3 into the variable M1. M1 will thus contain the three roots of : 0, 1 and –1.

Matrix Catalog: buttons and keys

The buttons and keys available in the Matrix Catalog are:

Button or Key	Purpose
Edit	Opens the highlighted matrix for editing.
Delete or ⌫ Del	Deletes the contents of the selected matrix.
Vect	Changes the selected matrix into a one-dimensional vector.
Send	Transmits the highlighted matrix to another HP Prime calculator, if available.
Shift Esc Clear (Clear)	Clears the contents of the reserved matrix variables `M0`–`M9` and deletes any user-named matrices.

Working with matrices

To open the Matrix Editor

To create or edit a matrix, go to the Matrix Catalog, and tap on a matrix. (You could also use the cursor keys to highlight the matrix and then press Edit). The Matrix Editor opens.

Matrix Editor: Buttons and keys

The buttons and keys available in the Matrix Editor are as follows:

Button or Key	Purpose
Edit	Copies the selected element to the entry line, where it can be edited. This item is visible only when an element in the matrix or vector is selected.
More	Opens an editing options menu.
Go To	Moves the cursor to the specified element in the matrix. This option is especially useful for very large matrices.
Go	Sets how the cursor moves after you press Enter ≈. The options are **Down**, **Right**, and **None**.

Button or Key	Purpose
Shift Esc Clear (Clear)	Deletes the highlighted row, or column, or the entire matrix. (You are prompted to make a choice.)
Shift ▲ ▼ ◀ ▶	Moves the cursor to the first row, last row, first column, or last column respectively.

Matrix Editor: More menu

The Matrix Editor More menu contains options similar to those in the List Editor More menu, but with additional options used only for editing matrices. These options are described in the following table.

Category	Option	Description
Insert	Row	Inserts a new row above the current row in the matrix. The new row contains zeroes.
	Column	Inserts a new column to the left of the current column in the matrix. The new column contains zeroes.
Delete	Row	Deletes the current row of the matrix.
	Column	Deletes the current column of the matrix.
	All	Deletes the contents of the matrix.
Select	Row	Selects the current row. After it is selected, the row can be copied.
	Column	Selects the current column. After it is selected, the column can be copied.
	Box	Opens a dialog box to select a rectangular array defined by a starting location and a final location. You can also tap and hold a cell to select it as the starting location, and then drag your finger to select a rectangular array of elements. After it is selected, the rectangular array can be copied.
Selection		Turns selection mode on and off. You can also tap and hold a cell and then drag your finger to select multiple cells.
Swap	Row	Transposes the values of the selected rows.
	Column	Transposes the values of the selected columns.

Creating a matrix in the Matrix Editor

1. Open the Matrix Catalog:

   Shift 4 Matrix U (Matrix)

2. If you want to create a vector, press ▲ or ▼ until the matrix you want to use is highlighted, tap Vect, and then press Enter ≈. Continue from step 4 below.

3. If you want to create a matrix, either tap on the name of the matrix (M0–M9), or press ▲ or ▼ until the matrix you want to use is highlighted and then press Enter.

   Note that an empty matrix will be shown with a size of `1*1` beside its name.

4. For each element in the matrix, type a number or an expression, and then tap OK or press Enter.

   You can enter complex numbers in complex form, that is, (*a, b*), where *a* is the real part and *b* is the imaginary part. You can also enter them in the form *a+bi.*

5. By default, on entering an element the cursor moves to the next column in the same row. You can use the cursor keys to move to a different row or column. You can also change the direction the cursor automatically moves by tapping Go. The Go button toggles between the following options:

   - Go →: the cursor moves to the cell to the right of the current cell when you press Enter.
   - Go ↓: the cursor moves to the cell below the current cell when you press Enter.
   - Go: the cursor stays in the current cell when you press Enter.

6. When done, press Shift 4 (Matrix) to return to the Matrix Catalog, or press ⌂ Settings to return to Home view. The matrix entries are automatically saved.

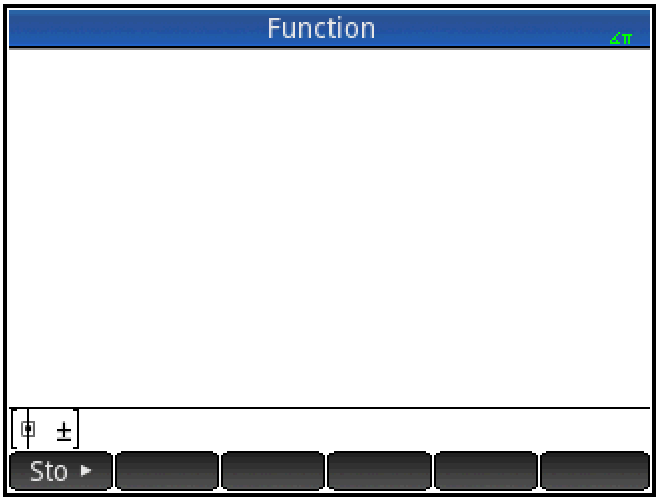
Matrices in Home view

You can enter and operate on matrices directly in Home view. The matrices can be named or unnamed.

Enter a vector or matrix in Home or CAS views directly in the entry line.

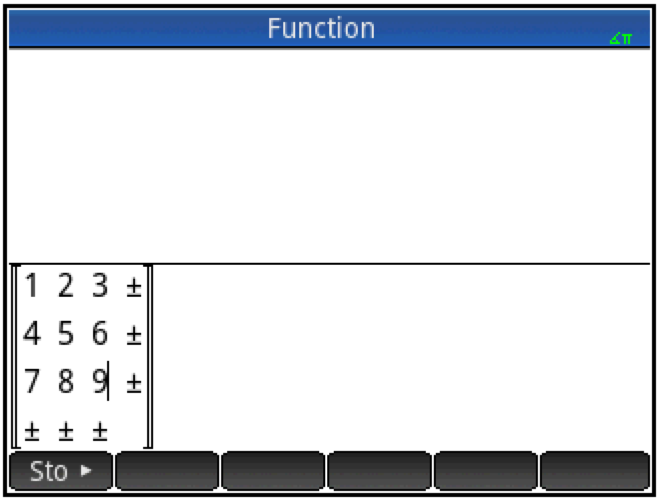
1. Press Shift 5 to start a vector; then press Shift 5 again to start a matrix.

   Alternately, you can press Units to open the Template menu and select either the vector template or one of the matrix templates. In the following figure, a vector has been started, with a dark square placeholder for the first value.

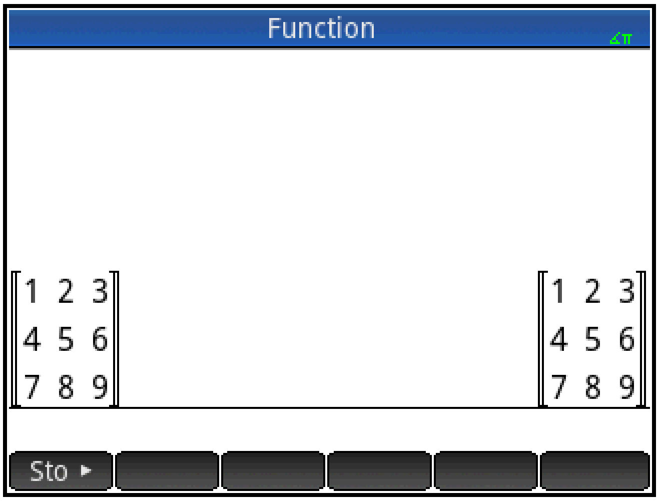
2. Enter a value in the square. Then press ▶ to enter a second value in the same row, or press + to add a row. The matrix will grow with you as you enter values, adding rows and columns as needed.

3. You can increase your matrix at any time, adding columns and rows as you please. You can also delete an entire row or column. Just place the cursor on the ± symbol at the end of a row or column. Then press + to insert a new row or column, or − to delete the row or column. You can also press Del to delete a row or column. In the figure above, pressing Del would delete the second row of the matrix.



$$\begin{bmatrix} 1 & 2 & 3 & \pm \\ 4 & 5 & 6 & \pm \\ 7 & 8 & 9 & \pm \\ \pm & \pm & \pm & \end{bmatrix}$$

4. When you are finished, press Enter and the matrix will be displayed in the History. You can then use or name your matrix.



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

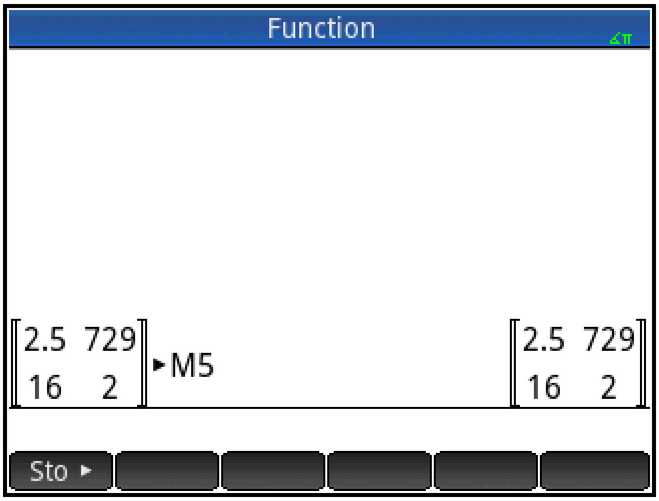
Storing a matrix

You can store a vector or matrix in a variable. You can do this before it is added to History, or you can copy it from History. When you've entered a vector or matrix in the entry line or copied it from History to the entry line, tap Sto ▸, enter a name for it and press Enter ≈. The variable names reserved for vectors and matrices are M0 through M9. You can always use a variable name you devise to store a vector or matrix. The new variable will appear in the **Vars** menu under User.

The following screen shows the matrix

$$\begin{bmatrix}2.5 & 729\\ 16 & 2\end{bmatrix}$$

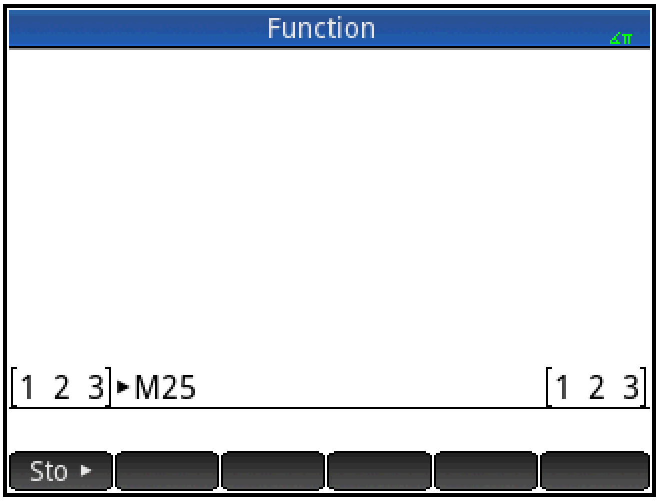
being stored in M5. Note that you can enter an expression (like 5/2) for an element of the matrix, and it will be evaluated upon entry.



Function

$\begin{bmatrix}2.5 & 729\\ 16 & 2\end{bmatrix}$▸M5 $\begin{bmatrix}2.5 & 729\\ 16 & 2\end{bmatrix}$

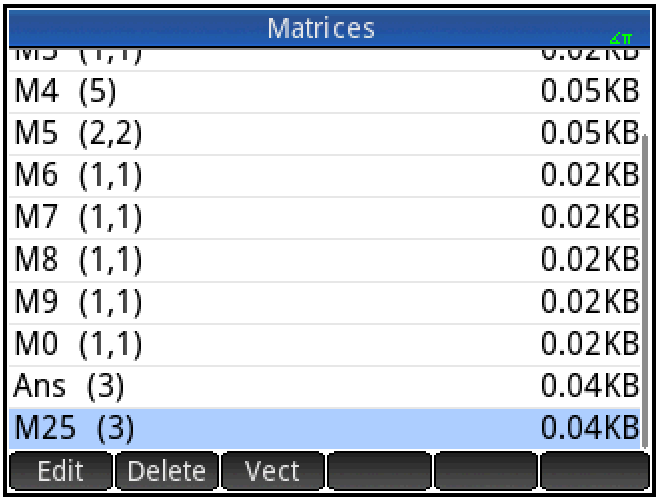
The following figure shows the vector [1 2 3] being stored in the user variable M25. You will be prompted to confirm that you want to create your own variable. Tap OK to proceed or Cancel to cancel.



Function

[1 2 3]▸M25 [1 2 3]

Once you tap OK, your new matrix will be stored under the name M25. This variable will show up in the User section of the **Vars** menu. You will also see your new matrix in the Matrix Catalog.



Matrices

```
M4  (5)       0.05KB
M5  (2,2)     0.05KB
M6  (1,1)     0.02KB
M7  (1,1)     0.02KB
M8  (1,1)     0.02KB
M9  (1,1)     0.02KB
M0  (1,1)     0.02KB
Ans  (3)      0.04KB
M25  (3)      0.04KB
```

Displaying a matrix

In Home view, enter the name of the vector or matrix and press Enter. If the vector or matrix is empty, zero is returned inside double square brackets.

Displaying one element

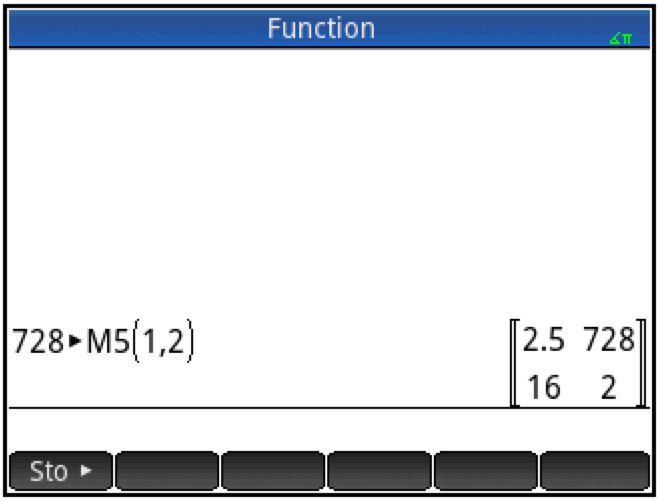
In Home view, enter matrixname(row,column). For example, if `M2` is `[[3,4],[5,6]]`, then `M2(1,2)` Enter returns `4`.

Storing one element

In Home view, enter value, tap Sto ▸, and then enter *matrixname(row,column)*.

For example, to change the element in the first row and second column of M5 to 728 and then display the resulting matrix:

728 Sto ▸ ALPHA +/− 5 ( ) 1 , 2 Enter



Function

728▸M5(1,2)

$$\begin{bmatrix} 2.5 & 728 \\ 16 & 2 \end{bmatrix}$$

An attempt to store an element to a row or column beyond the size of the matrix results in resizing the matrix to allow the storage. Any intermediate cells will be filled with zeroes.

Matrix references

`M1(1,2)` returns the value in the first row and second column of matrix M1. `M1(1)` returns the first row of M1 as a vector. In Home view, `M1(-1)` returns the first column of M1 as a vector. In CAS view, this command cannot be used with negative arguments.

`M1({1,2})` returns the first two rows of M1. `M1({1,1},{2,2}})` extracts a sub-matrix from the element in the first row and column to the element in the second row and column. If M1 is a vector, then `M1({1,3})` extracts a sub-vector of the first three elements.

Sending a matrix

You can send matrices between calculators just as you can send apps, programs, lists, and notes. See "Sharing data" for instructions.

Matrix arithmetic

You can use the arithmetic functions (+, –, ×, ÷, and powers) with matrix arguments. Division left-multiplies by the inverse of the divisor. You can enter the matrices themselves or enter the names of stored matrix variables. The matrices can be real or complex.

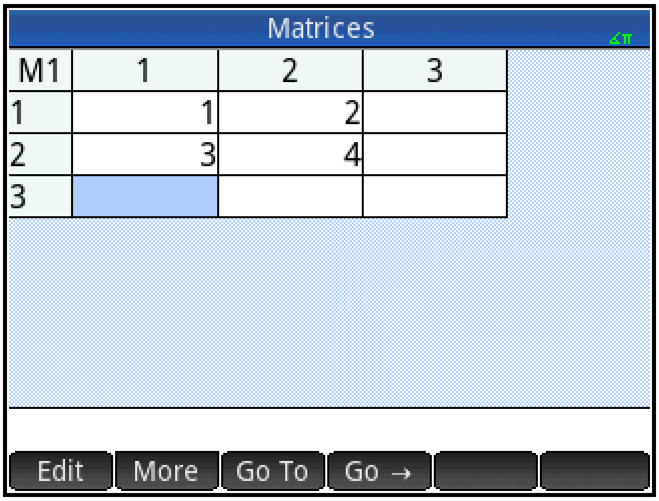
For the next examples, store [[1,2],[3,4]] in M1 and [[5,6],[7,8]] in M2.

1. Select the first matrix:

   Shift 4 Matrix U (Matrix)

2. Enter the matrix elements:

   Go → 1 Enter ≈ 2 Enter ≈ 3 Enter ≈ 4 Enter ≈



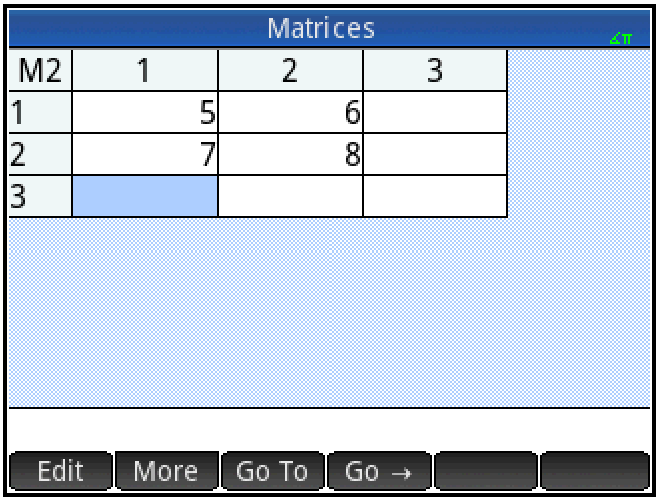
Matrices

M1	1	2	3
1	1	2	
2	3	4	
3			

3. Select the second matrix:

   Shift 4 (Matrix)

   Tap **M2** or highlight it and press Enter.



Matrices

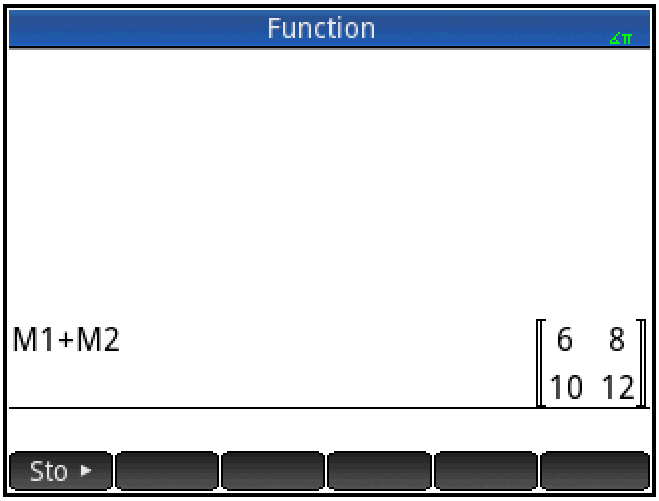
M2	1	2	3
1	5	6	
2	7	8	
3			

4. Enter the matrix elements:

   5 Enter 6 Enter 7 Enter 8 Enter

5. In Home view, add the two matrices you have just created.

   ⌂ ALPHA +/− 1 + ALPHA +/− 2 Enter



Function

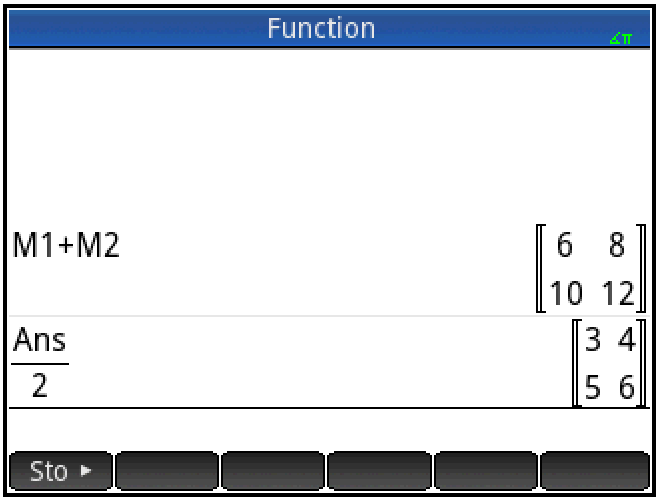
M1+M2 $\begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$

Multiplying and dividing by a scalar

For division by a scalar, enter the matrix first, then the operator, then the scalar. For multiplication, the order of the operands does not matter.

The matrix and the scalar can be real or complex. For example, to divide the result of the previous example by 2, press the following keys:

÷ 2 Enter



Function

M1+M2 $\begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$

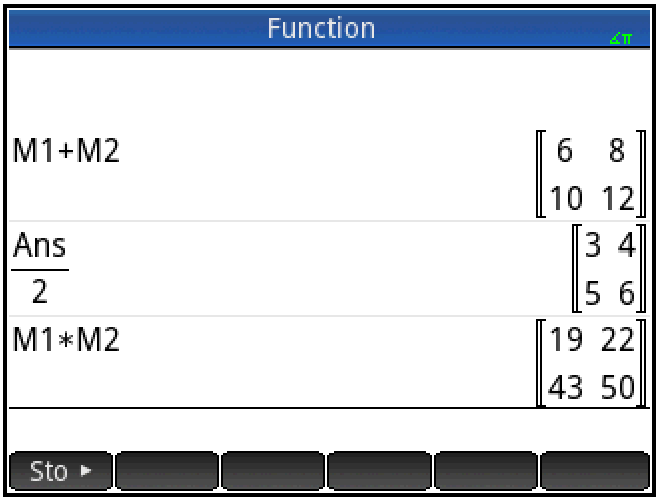
$\frac{\text{Ans}}{2}$ $\begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$

Multiplying two matrices

To multiply the two matrices that you created for the previous example, press the following keys:

ALPHA alpha +/- M 1 × ALPHA alpha +/- M 2 Enter ≈

To multiply a matrix by a vector, enter the matrix first, then the vector. The number of elements in the vector must equal the number of columns in the matrix.



M1+M2 $\begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$

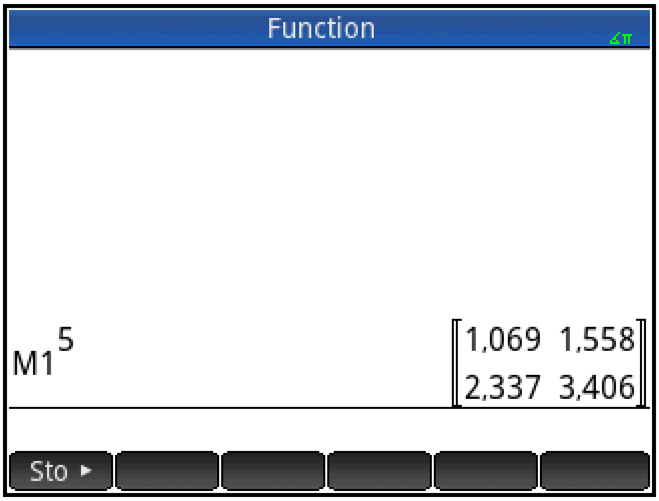
$\frac{\text{Ans}}{2}$ $\begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$

M1*M2 $\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$

Raising a matrix to a power

You can raise a matrix to any power as long as the power is an integer. The following example shows the result of raising matrix M1, created earlier, to the power of 5.

ALPHA alpha +/- M 1 x^2 5 Enter ≈

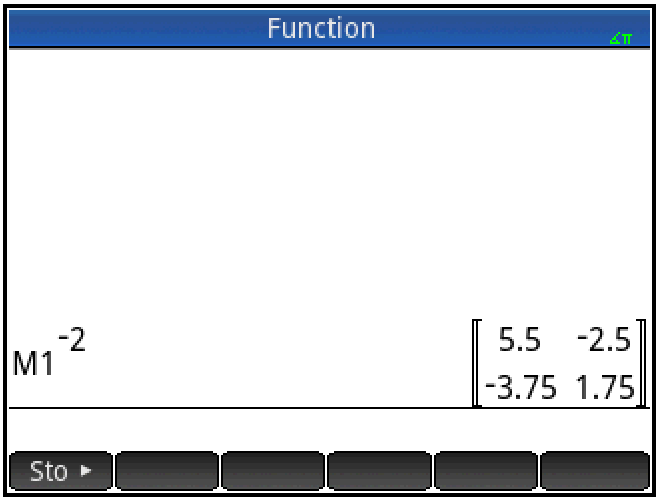


$M1^5$ $\begin{bmatrix} 1{,}069 & 1{,}558 \\ 2{,}337 & 3{,}406 \end{bmatrix}$

You can also raise a matrix to a power without first storing it as a variable.

Matrices can also be raised to negative powers. In this case, the result is equivalent to 1/[matrix]^ABS(power). In the following example, M1 is raised to the power of –2.

ALPHA +/− 1 x^2 +/− 2 Enter



$M1^{-2}$

$$\begin{bmatrix} 5.5 & -2.5 \\ -3.75 & 1.75 \end{bmatrix}$$

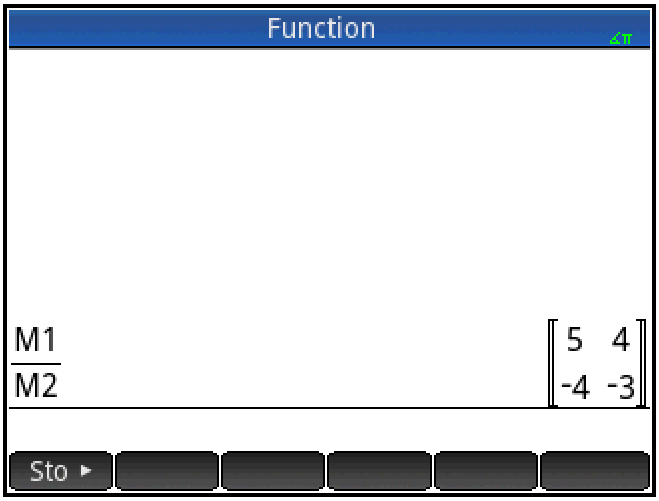
Dividing by a square matrix

For division of a matrix or a vector by a square matrix, the number of rows of the dividend (or the number of elements, if it is a vector) must equal the number of rows in the divisor.

This operation is not a mathematical division: it is a left multiplication by the inverse of the divisor. M1/M2 is equivalent to $M2^{-1}$ * M1.

To divide the two matrices you created for the previous example, press the following keys:

ALPHA +/− 1 ÷ ALPHA +/− 2



$\frac{M1}{M2}$

$$\begin{bmatrix} 5 & 4 \\ -4 & -3 \end{bmatrix}$$

Inverting a matrix

You can invert a square matrix in Home view by typing the matrix (or its variable name) and pressing Shift ÷ Enter. You can also use the `INVERSE` command in the Matrix category of the Math menu.

Negating each element

You can change the sign of each element in a matrix by pressing +/−, entering the matrix name, and pressing Enter.

Solving systems of linear equations

You can use matrices to solve systems of linear equations, such as the following:

2x+3y+4z=5

x+y−z=7

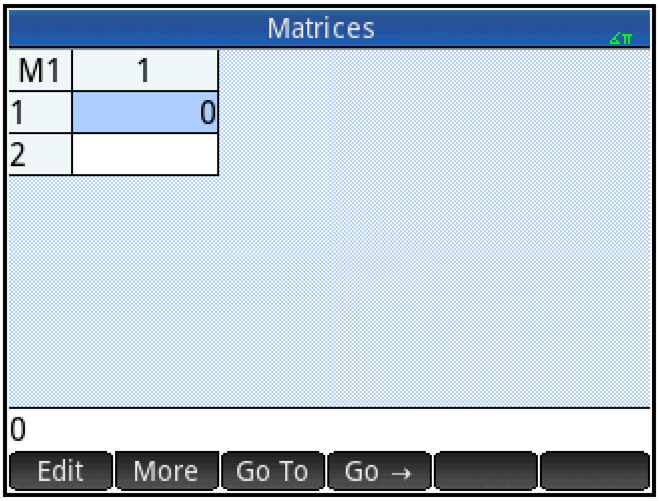
4x−y+2z=1

In this example we will use matrices M1 and M2, but you could use any available matrix variable name.

In this example we will use matrices M1 and M2, but you could use any available matrix variable name.

1. Open the Matrix Catalog, clear M1, choose to create a vector, and open the Matrix Editor:

   Shift 4 [press ▲ or ▼ to select M1 ⌫ OK Vect Enter



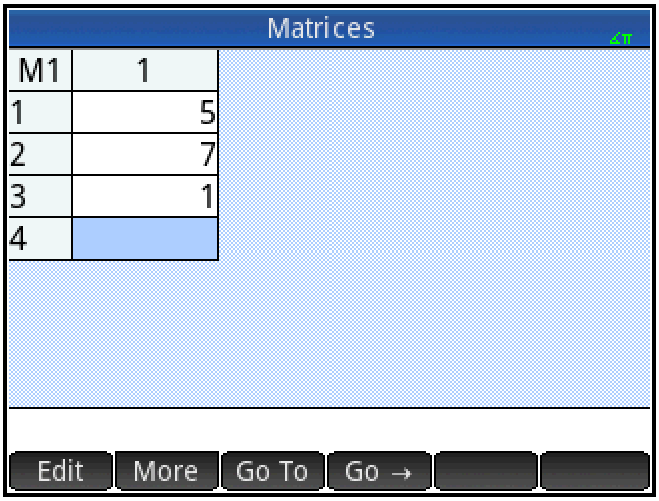
Matrices

M1	1
1	0
2	

0

2. Create the vector of the three constants in the linear system.

5 Enter 7 Enter 1 Enter



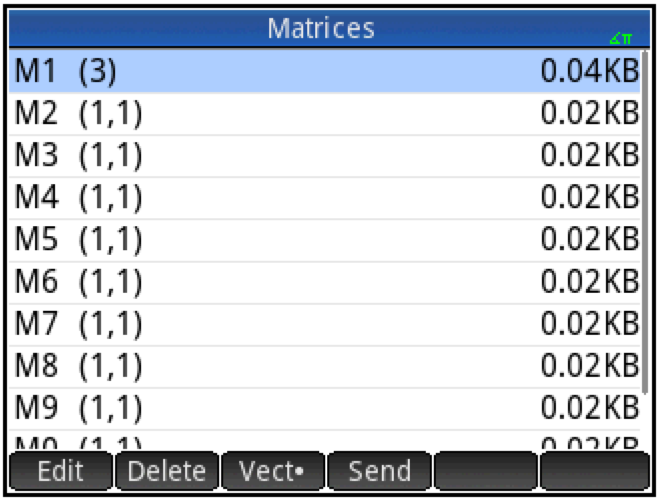
Matrices

M1	1
1	5
2	7
3	1
4	

3. Return to the Matrix Catalog.

Shift 4

The size of M1 should be showing as 3.

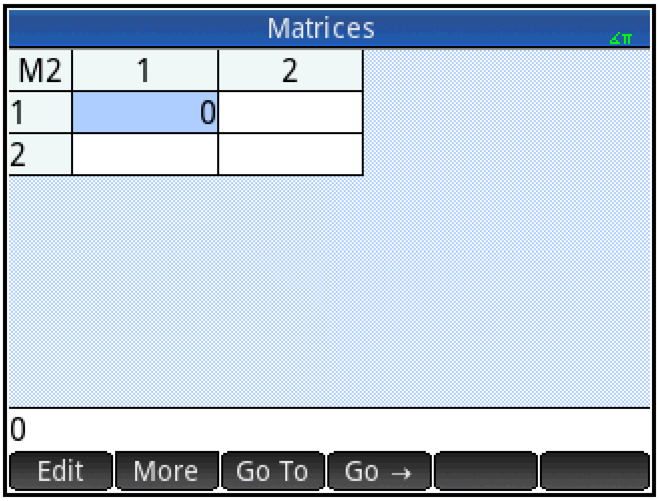


Matrices

```
M1  (3)      0.04KB
M2  (1,1)    0.02KB
M3  (1,1)    0.02KB
M4  (1,1)    0.02KB
M5  (1,1)    0.02KB
M6  (1,1)    0.02KB
M7  (1,1)    0.02KB
M8  (1,1)    0.02KB
M9  (1,1)    0.02KB
M0  (1,1)    0.02KB
```

4. Select and clear M2, and reopen the Matrix Editor:

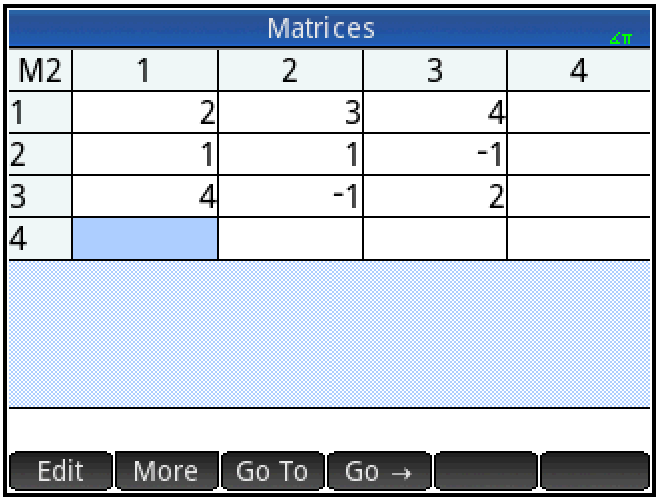
[Press ▼ or ▲ to select M2] Del OK Enter



M2	1	2
1	0	
2		

5. Enter the equation coefficients.

2 Enter 3 Enter [Tap in cell R1, C3.] 4 Enter 1 Enter 1 Enter +/− 1 Enter 4 Enter +/− 1 Enter 2 Enter

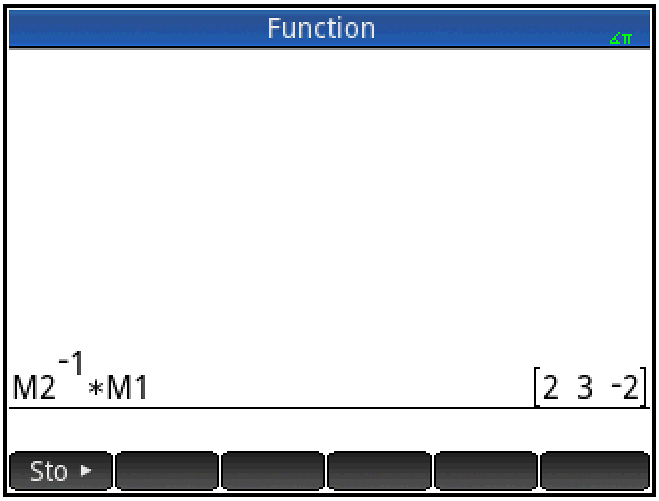


M2	1	2	3	4
1	2	3	4	
2	1	1	-1	
3	4	-1	2	
4				

6. Return to Home view and left-multiply the constants vector by the inverse of the coefficients matrix:

⌂ Settings ALPHA alpha +/– 2 Shift ÷ × ALPHA alpha +/– 1 Enter

The result is a vector of the solutions: x = 2, y = 3 and z = –2.



Function

M2^{-1}∗M1 [2 3 -2]

An alternative method is to use the RREF function (see the "RREF" section).

Matrix functions and commands

Functions

Functions can be used in any app or in Home view. They are listed on the Math menu under the Matrix category. They can be used in mathematical expressions—primarily in Home view—as well as in programs.

Functions always produce and display a result. They do not change any stored variables, such as a matrix variable.

Functions have arguments that are enclosed in parentheses and separated by commas; for example, `CROSS`(*vector1*,*vector2*). The matrix input can be either a matrix variable name (such as `M1`) or the actual matrix data inside brackets. For example, `CROSS(M1,[1 2])`.

Menu format

By default, a Matrix function is presented on the Math menu using its descriptive name, not its common command name. Thus the shorthand name `TRN` is presented as **Transpose** and `DET` is presented as **Determinant**.

If you prefer the **Math** menu to show command names instead, deselect the **Menu Display** option on page 2 of the Home Settings screen.

Commands

Matrix commands differ from matrix functions in that they do not return a result. For this reason, these functions can be used in an expression and matrix commands cannot. The matrix commands are designed to support programs that use matrices.

The matrix commands are listed in the Matrix category of the Commands menu in the Program Editor. They are also listed in the Catalog menu, one of the Toolbox menus. Press ⌂ Settings and tap Catlg to display the commands catalog. The matrix functions are described in the following sections of this chapter; the matrix commands are described in the chapter Programming (see page 544).

Argument conventions

- For *row#* or *column#*, supply the number of the row (counting from the top, starting with 1) or the number of the column (counting from the left, starting with 1).
- The argument *matrix* can refer to either a vector or a matrix.

Matrix functions

The matrix functions are available in the Matrix category on the Math menu: Mem B Select **Matrix**. Select a function.

Matrix

Transpose

Transposes matrix. For a complex matrix, TRN finds the conjugate transpose.

```
TRN(matrix)
```

Example:

$\mathtt{TRN}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$ returns $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

Determinant

Determinant of a square matrix.

```
DET(matrix)
```

Example:

$\mathtt{DET}\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ returns -2

RREF

Reduced Row-Echelon Form. Changes a rectangular matrix to its reduced row-echelon form.

```
RREF(matrix)
```

Example:

$\mathtt{RREF}\left(\begin{bmatrix} 1 & -2 & 1 \\ 3 & 4 & -1 \end{bmatrix}\right)$ returns $\begin{bmatrix} 1 & 0 & 0.2 \\ 0 & 1 & -0.4 \end{bmatrix}$

Create

Make

Creates a matrix of dimension rows × columns, using expression to calculate each element. If expression contains the variables I and J, then the calculation for each element substitutes the current row number for I and the current column number for J. You can also create a vector by the number of elements (e) instead of the number of rows and columns.

```
MAKEMAT(expression, rows, columns)
```

```
MAKEMAT(expression, elements)
```

Examples:

`MAKEMAT(0,3,3)` returns a 3 × 3 zero matrix, [[0,0,0],[0,0,0],[0,0,0]].

`MAKEMAT(√2,2,3)` returns the 2 × 3 matrix [[√2,√2,√2],[√2,√2,√2]].

`MAKEMAT(I+J–1,2,3)` returns the 2 × 3 matrix [[1,2,3],[2,3,4]]

Note in the example above that each element is the sum of the row number and column number minus 1.

Identity

Identity matrix. Creates a square matrix of dimension size × size whose diagonal elements are 1 and offdiagonal elements are zero.

```
IDENMAT(size)
```

Random

Given two integers, n and m, and a matrix name, creates an n x m matrix that contains random integers in the range −99 through 99 with a uniform distribution and stores it in the matrix name. Given only one integer, returns a vector of that length, filled with random integers. Given an optional additional pair of integers, returns a matrix of the random numbers restricted to the interval defined by those integers.

```
randMat([MatrixName],n,[m], [lower, upper})
```

Example:

`RANDMAT(M1,2,2)` returns a 2x2 matrix with random integer elements, and stores it in M1.

Jordan

Returns a square nxn matrix with expr on the diagonal, 1 above and 0 everywhere else.

```
JordanBlock(Expr,n)
```

Example:

`JordanBlock(7,3)` returns $\begin{bmatrix} 7 & 1 & 0 \\ 0 & 7 & 1 \\ 0 & 0 & 7 \end{bmatrix}$

Hilbert

Given a positive integer, n, returns the nth order Hilbert matrix. Each element of the matrix is given by the formula 1/(j+k-1) where j is the row number and k is the column number.

```
hilbert(n)
```

Example:

In CAS view, `hilbert(4)` returns $\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$

Isometric

Matrix of an isometry given by its proper elements.

```
mkisom(vector,sign(1 or -1))
```

Example:

In CAS view, `mkisom([1,2],1)` returns $\begin{bmatrix} \cos(1) & -\sin(1) \\ \sin(1) & \cos(1) \end{bmatrix}$

Vandermonde

Returns the Vandermonde matrix. Given a vector [n1, n2 … nj], returns a matrix whose first row is [$(n1)^0$, $(n1)^1$, $(n1)^2$, …,$(n1)^{j-1}$]. The second row is [$(n2)^0$, $(n2)^1$, $(n2)^2$, …,$(n2)^{j-1}$], etc.

```
vandermonde(vector)
```

Example:

`vandermonde([1 3 5])` returns $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 9 \\ 1 & 5 & 25 \end{bmatrix}$

Basic

Norm

Returns the Frobenius norm of a matrix.

```
|matrix|
```

Example:

$\left\| \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \right\|$ returns 5.47722557505

Row Norm

Row Norm. Finds the maximum value (over all rows) for the sums of the absolute values of all elements in a row.

```
ROWNORM(matrix)
```

Example:

`ROWNORM`$\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$ returns 7

Column Norm

Column Norm. Finds the maximum value (over all columns) of the sums of the absolute values of all elements in a column.

```
COLNORM(matrix)
```

Example:

`COLNORM`$\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$ returns 6

Spectral Norm

Spectral Norm of a square matrix.

```
SPECNORM(matrix)
```

Example:

`SPECNORM`$\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$ returns 5.46498570422

Spectral Radius

Spectral Radius of a square matrix.

```
SPECRAD(matrix)
```

Example:

`SPECRAD(matrix)`$\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$ returns 5.37228132327

Condition

Condition Number. Finds the 1-norm (column norm) of a square matrix.

```
COND(matrix)
```

Example:

`COND`$\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$ returns 21

Rank

Rank of a rectangular matrix.

```
RANK(matrix)
```

Example:

`RANK`$\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)$ returns 2

Pivot

Given a matrix, a row number n, and a column number, m, uses Gaussian elimination to return a matrix with zeroes in column m, except that the element in column m and row n is kept as a pivot.

```
pivot(matrix,n,m)
```

Example:

$\text{pivot}\left(\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}, 1, 1\right)$ returns $\begin{bmatrix}1 & 2\\0 & -2\\0 & -4\end{bmatrix}$

Trace

Finds the trace of a square matrix. The trace is equal to the sum of the diagonal elements. (It is also equal to the sum of the eigenvalues.)

```
TRACE(matrix)
```

Example:

$\text{TRACE}\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\right)$ returns 5

Advanced

Eigenvalues

Displays the eigenvalues in vector form for matrix.

```
EIGENVAL(matrix)
```

Example:

$\text{EIGENVAL}\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\right)$ returns [5.37228... –0.37228...]

Eigenvectors

Eigenvectors and eigenvalues for a square matrix. Displays a list of two arrays. The first contains the eigenvectors and the second contains the eigenvalues.

```
EIGENVV(matrix)
```

Example:

$\text{EIGENVV}\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\right)$ returns the following matrices:

$$\left\{\begin{bmatrix}0.4159\ldots & -0.8369\ldots\\0.9093\ldots & 0.5742\ldots\end{bmatrix}, \begin{bmatrix}5.3722\ldots & 0\\0 & -0.3722\ldots\end{bmatrix}\right\}$$

Jordan

Returns the list made by the matrix of passage and the Jordan form of a matrix.

```
jordan(matrix)
```

Example:

`jordan` returns $\left[\begin{bmatrix} \sqrt{2} & -\sqrt{2} \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} \sqrt{2} & 0 \\ 0 & -\sqrt{2} \end{bmatrix}\right]$

Diagonal

Given a list, returns a matrix with the list elements along its diagonal and zeroes elsewhere. Given a matrix, returns a vector of the elements along its diagonal.

`diag(list)` or `diag(matrix)`

Example:

`diag`$\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$ returns `[1 4]`

Cholesky

For a numerical symmetric matrix A, returns the matrix L such that A=L*tran(L).

`cholesky(matrix)`$\left(\begin{bmatrix} 3 & 1 \\ 1 & 4 \end{bmatrix}\right)$

Example:

In CAS view, `cholesky` returns $\left(\begin{bmatrix} \sqrt{3} & 0 \\ \frac{\sqrt{3}}{3} & \frac{\sqrt{33}}{3} \end{bmatrix}\right)$ after simplification

Hermite

Hermite normal form of a matrix with coefficients in Z: returns U,B such that U is invertible in Z, B is upper triangular and B=U*A.

`ihermite(Mtrx(A))`

Example:

`ihermite`$\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)$ returns $\left[\begin{bmatrix} -3 & 1 & 0 \\ 4 & -1 & 0 \\ -1 & 2 & -1 \end{bmatrix}, \begin{bmatrix} 1 & -1 & -3 \\ 0 & 3 & 6 \\ 0 & 0 & 0 \end{bmatrix}\right]$

Hessenberg

Matrix reduction to Hessenberg form. Returns [P,B] such that B=inv(P)*A*P.

`hessenberg(Mtrx(A))`

Example:

In CAS view, `hessenberg`$\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)$ returns $\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \frac{4}{7} & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & \frac{29}{7} & 2 \end{bmatrix} & \begin{bmatrix} 7 & \frac{39}{7} & 8 \end{bmatrix} & \begin{bmatrix} 0 & \frac{278}{49} & \frac{3}{7} \end{bmatrix} \end{bmatrix}$

Smith

Smith normal form of a matrix with coefficients in Z: returns U,B,V such that U and V invertible in Z, B is diagonal, B[i,i] divides B[i+1,i+1], and B=U*A*V.

```
ismith(Mtrx(A))
```

Example:

$$\mathrm{ismith}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right) \text{ returns } \begin{bmatrix}\begin{bmatrix}1&0&0\\4&-1&0\\-1&2&-1\end{bmatrix}&\begin{bmatrix}1&0&0\\0&3&0\\0&0&0\end{bmatrix}&\begin{bmatrix}1&-2&1\\0&1&-2\\0&0&1\end{bmatrix}\end{bmatrix}$$

Factorize

LQ

LQ Factorization. Factorizes a *m × n* matrix into three matrices L, Q, and P, where *{[L[m × n lowertrapezoidal]], [Q[n × n orthogonal]],[P[m × m permutation]]}*, and P*A=L*Q.

```
LQ(matrix)
```

Examples:

$$\mathrm{LQ}\left(\begin{bmatrix}1&2\\3&4\end{bmatrix}\right) \text{ returns } \left\{\begin{bmatrix}2.2360\ldots&0\\4.9193\ldots&0.8944\ldots\end{bmatrix},\begin{bmatrix}0.4472\ldots&0.8944\ldots\\0.8944\ldots&-0.4472\ldots\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}\right\}$$

LSQ

Least Squares. Displays the minimum norm least squares matrix (or vector) corresponding to the system matrix1*X=matrix2.

```
LSQ(matrix1, matrix2)
```

Example:

$$\mathrm{LSQ}\left(\begin{bmatrix}1&2\\3&4\end{bmatrix},\begin{bmatrix}5\\11\end{bmatrix}\right) \text{ returns } \begin{bmatrix}1\\2\end{bmatrix}$$

LU

LU Decomposition. Factorizes a square matrix into three matrices L, U, and P, where *{[L[lowertriangular]], [U[uppertriangular]],[P[permutation]] }}* and P*A=L*U.

```
LU(matrix)
```

Example:

$$\mathrm{LU}\left(\begin{bmatrix}1&2\\3&4\end{bmatrix}\right) \text{ returns } \left\{\begin{bmatrix}1&0\\0.3333\ldots&1\end{bmatrix},\begin{bmatrix}3&4\\0&0.6666\ldots\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}\right\}$$

QR

QR Factorization. Factorizes an *m×n* matrix A numerically as Q*R, where Q is an orthogonal matrix and R is an upper triangular matrix, and returns R. R is stored in var2 and Q=A*inv(R) is stored in var1.

```
QR(matrix A,var1,var2)
```

Example:

$$\mathrm{QR}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right) \text{ returns } \left\{\begin{bmatrix}0.3612\ldots & 0.9486\ldots\\ 0.9486\ldots & -0.3162\ldots\end{bmatrix}, \begin{bmatrix}3.1622\ldots & 4.4271\ldots\\ 0 & 0.6324\ldots\end{bmatrix}, \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right\}$$

SCHUR

Schur Decomposition. Factorizes a square matrix into two matrices. If *matrix* is real, then the result is , *{[[orthogonal]],[[upper-quasi triangular]]}*. If *matrix* is complex, then the result is *{[[unitary]],[[upper-triangular]]}*.

```
SCHUR(matrix)
```

Example:

$$\mathrm{SCHUR}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right) \text{ returns } \left\{\begin{bmatrix}0.4159\ldots & 0.9093\ldots\\ 0.9093\ldots & 0.4159\ldots\end{bmatrix}, \begin{bmatrix}5.3722\ldots & 1\\ 5.55\times10^{-17} & -0.3722\end{bmatrix}\right\}$$

SVD

Singular Value Decomposition. Factorizes an *m* × *n* matrix into two matrices and a vector: *{[[m × m square orthogonal]], [real], [[n × n square orthogonal]]}*.

```
SVD(matrix)
```

Example:

$$\mathrm{SVD}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right) \text{ returns } \left\{\begin{bmatrix}0.4045\ldots & -0.9145\ldots\\ 0.9145\ldots & 0.4045\ldots\end{bmatrix}, \begin{bmatrix}5.4649\ldots & 0.3659\ldots\end{bmatrix}, \begin{bmatrix}0.5760\ldots & 0.8174\ldots\\ 0.8174\ldots & -0.5760\end{bmatrix}\right\}$$

SVL

Singular Values. Returns a vector containing the singular values of matrix.

```
SVL(matrix)
```

Example:

$$\mathrm{SVL}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right) \text{ returns } [5.4649\ldots \quad 0.3659\ldots]$$

Vector

Cross Product

Cross Product of *vector1* with *vector2*.

```
CROSS(vector1, vector2)
```

Example:

`CROSS ([1 2],[3 4])` returns `[0 0 -2]`

Dot Product

Dot Product of two vectors, vector1 and vector2.

```
dot(vector1, vector2)
```

Example:

`dot([1 2],[3 4]}` returns 11

L^2Norm

Returns the l^2 norm (sqrt(x1^2+x2^2+...xn^2)) of a vector.

```
l2norm(Vect)
```

Example:

`l2norm([3 4 -2])` returns `√29`

L^1Norm

Returns the l^1 norm (sum of the absolute values of the coordinates) of a vector.

```
l1norm(Vect)
```

Example:

`l1norm([3 4 -2])` returns 9

Max Norm

Returns the l∞ norm (the maximum of the absolute values of the coordinates) of a vector.

```
maxnorm(Vect or Mtrx)
```

Example:

`maxnorm([1 2 3 -4])` returns 4

Examples

Identity Matrix

You can create an identity matrix with the `IDENMAT` function. For example, `IDENMAT(2)` creates the 2×2 identity matrix [[1,0],[0,1]].

You can also create an identity matrix using the `MAKEMAT` (make matrix) function. For example, entering `MAKEMAT(I ≠J,4,4)` creates a 4 × 4 matrix showing the numeral 1 for all elements except zeros on the diagonal. The logical operator (≠) returns 0 when I (the row number) and J (the column number) are equal, and returns 1 when they are not equal. (You can insert ≠ by choosing it from the relations palette: Shift 6 ≤,≥,≠ w .)

Transposing a Matrix

The `TRN` function swaps the row–column and column–row elements of a matrix. For instance, element 1,2 (row 1, column 2) is swapped with element 2,1; element 2,3 is swapped with element 3,2; and so on.

For example, `TRN([[1,2],[3,4]])` creates the matrix `[[1,3],[2,4]].`

Reduced-Row Echelon Form

The set of equations

$x - 2y + 3z = 14$

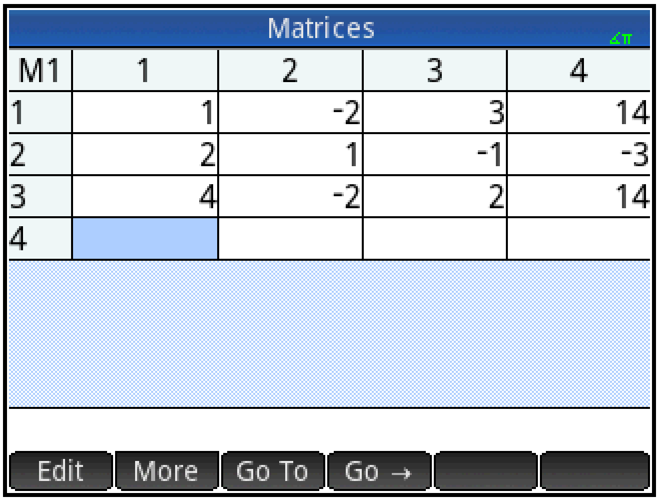
$2x + y - z = -3$

$4x - 2y + 2z = 14$

can be written as the augmented matrix

$$\left[\begin{array}{ccc|c} 1 & -2 & 3 & 14 \\ 2 & 1 & -1 & -3 \\ 4 & -2 & 2 & 14 \end{array}\right]$$

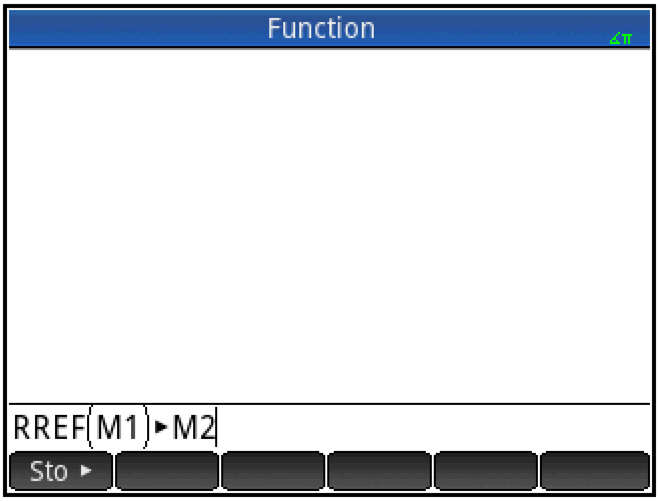
Which can then be stored as a 3 x 4 real matrix in any matrix variable. M1 is used in this example.



Matrices

M1	1	2	3	4
1	1	-2	3	14
2	2	1	-1	-3
3	4	-2	2	14
4				

You can then use the `RREF` function to change this to reduced-row echelon form, storing it in any matrix variable. M2 is used in this example.

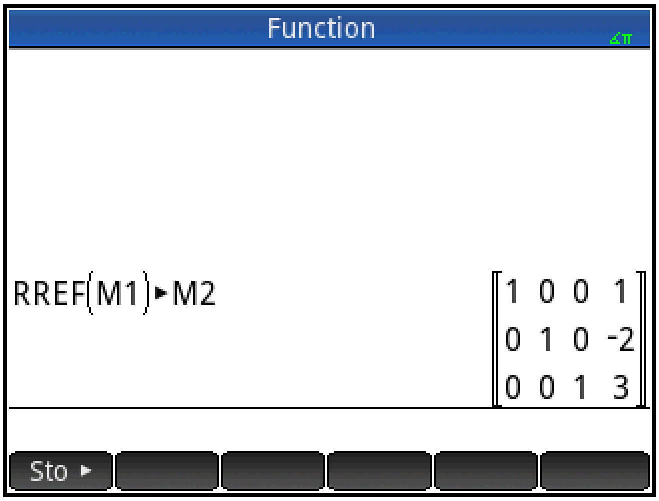


Function

RREF(M1)▸M2

The reduced row echelon matrix gives the solution to the linear equation in the fourth column.

An advantage of using the `RREF` function is that it will also work with inconsistent matrices resulting from systems of equations which have no solution or infinite solutions.



RREF(M1)▸M2

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

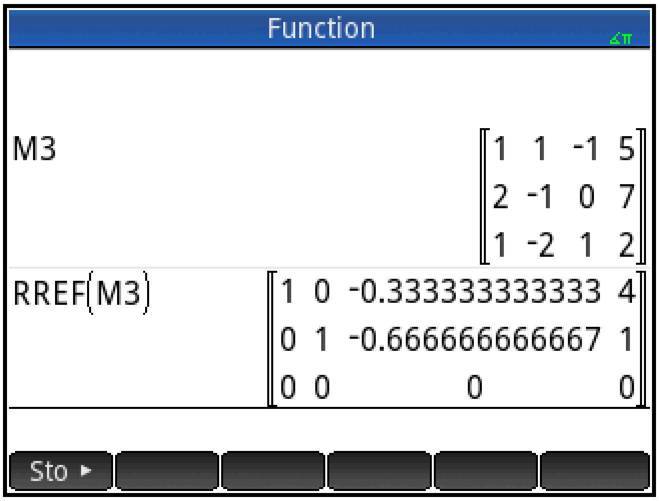
For example, the following set of equations has an infinite number of solutions:

$x + y - z = 5$

$2x - y = 7$

$x - 2y + z = 2$

The final row of zeros in the reduced-row echelon form of the augmented matrix indicates an inconsistent system with infinite solutions.



M3

$$\begin{bmatrix} 1 & 1 & -1 & 5 \\ 2 & -1 & 0 & 7 \\ 1 & -2 & 1 & 2 \end{bmatrix}$$

RREF(M3)

$$\begin{bmatrix} 1 & 0 & -0.333333333333 & 4 \\ 0 & 1 & -0.666666666667 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

27 Notes and Info

The HP Prime has two text editors for entering notes:

- The Note Editor: opens from within the Note Catalog (which is a collection of notes independent of apps).
- The Info Editor: opens from the Info view of an app. A note created in the Info view is associated with the app and stays with it if you send the app to another calculator.

The Note Catalog

Subject to available memory, you can store as many notes as you want in the Note Catalog. These notes are independent of any app. The Note Catalog lists the notes by name. This list excludes notes that were created in any app's Info view, but these can be copied and then pasted into the Note Catalog via the clipboard. From the Note Catalog, you create or edit individual notes in the Note Editor.

Note Catalog: button and keys

Press Shift 0 (Notes) to enter the Note Catalog. While you are in the Note Catalog, you can use the following buttons and keys. Note that some buttons will not be available if there are no notes in the Note Catalog.

Button or Key	Purpose
Edit	Opens the selected note for editing.
New	Begins a new note, and prompts you for a name.
More	Tap to provide the following additional features. **Save:** Creates a copy of the selected note and prompts you to save it under a new name. **Rename:** Renames the selected note. **Sort:** Sorts the list of notes (sort options are alphabetical and chronological). **Delete:** Deletes all notes. **Clear:** Creates a copy of the selected note and prompts you to save it under a new name. **Send:** Sends the selected note to another HP Prime calculator.
Del	Deletes the selected note.
Shift Esc Clear	Deletes all notes in the catalog.

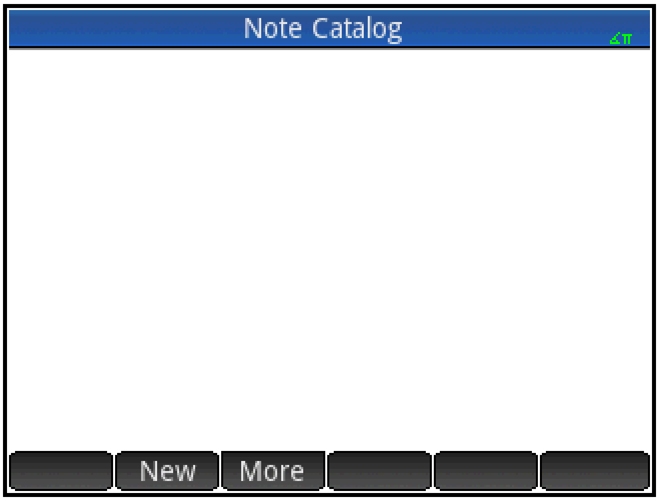
The Note Editor

The Note Editor is where you create and edit notes. You can launch the Note Editor from the Notes Catalog, and also from within an app. Notes created within an app stay with that app even if you send the app to another calculator. Such notes do not appear in the Notes Catalog. They can only be read when the associated app is open. Notes created via the Notes Catalog are not specific to any app and can be viewed at any time by opening the Notes Catalog. Such notes can also be sent to another calculator.

To create a note from the Notes catalog

1. Open the Note Catalog.

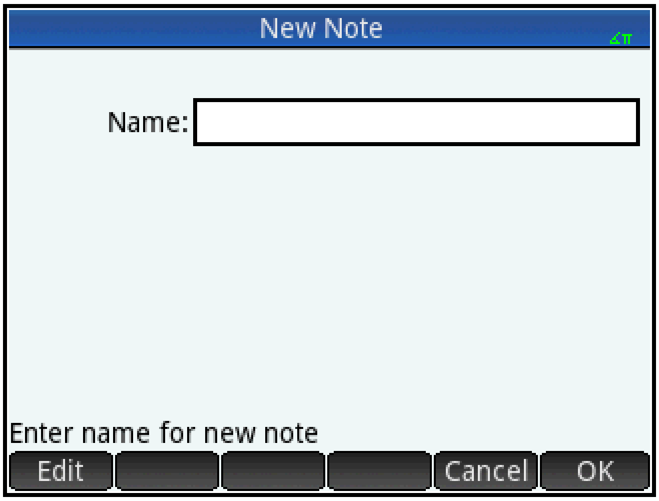
Shift 0 Notes



Note Catalog

2. Create a new note.

New



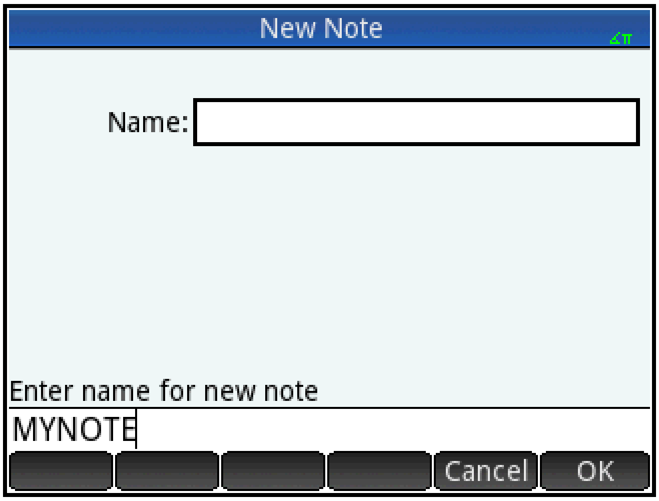
New Note

Name:

Enter name for new note

3. Enter a name for your note. In this example, we'll call the note MYNOTE. ALPHA alpha ALPHA alpha MYNOTE

OK OK



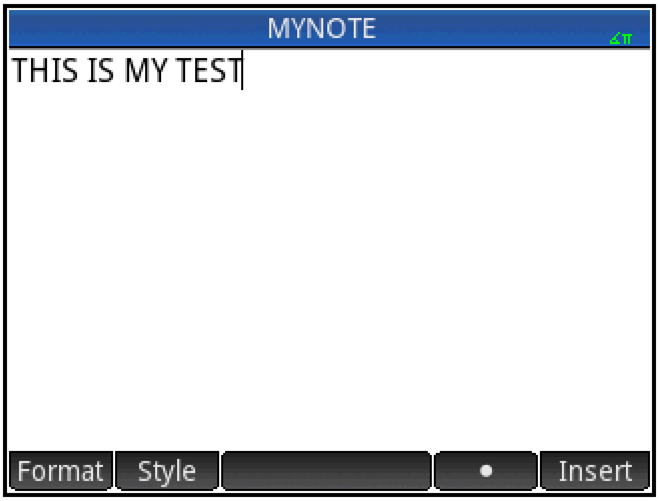
New Note

Name:

Enter name for new note

MYNOTE

4. Write your note, using the editing keys and formatting options described in the following sections. When you are finished, exit the Note Editor by pressing ⌂ Settings or pressing Apps Info and opening an app. Your work is automatically saved. To access your new note, return to the Notes Catalog.



MYNOTE

THIS IS MY TEST

Creating a note for an app

You can also create a note that is specific to an app and which stays with the app should you send the app to another calculator. Notes created this way take advantage of all the formatting features of the Note Editor (see below).

Note Editor: buttons and keys

The following buttons and keys are available while you are adding or editing a note.

Button or Key	Purpose
Format	Opens the text formatting menu. See Formatting options on page 548.
Style	Provides bold, italic, underline, full caps, superscript and subscript options. See Formatting options on page 548.
•	A toggle button that offers three types of bullet. See Formatting options on page 548.
Insert	Starts a 2D editor for entering mathematical expressions in textbook format; see Inserting mathematical expressions on page 549
␣	Enters a space during text entry.
▲ Page ▼	Moves from page to page in a multi-page note.
Shift View Copy	Shows options for copying text in a note. See below.
Begin	Copy option. Mark where to begin a text selection.
End	Copy option. Mark where to end a text selection.
All	Copy option. Select the entire note.
Cut	Copy option. Cut the selected text.
Copy	Copy option. Copy the selected text.
⌫ Del	Deletes the character to the left of the cursor.
Enter ≈	Starts a new line.
Shift Esc Clear (Clear)	Erases the entire note.
Vars Chars A	Menu for entering variable names, and the contents of variables.
Mem B	Menu for entering math commands.
Shift Vars Chars A (Clear)	Displays a palette of special characters. To type one, highlight it and tap OK or press Enter ≈. To copy a character *without* closing the Chars menu, select it and tap Echo.

Entering uppercase and lowercase characters

The following table below describes how to quickly enter uppercase and lowercase characters.

Keys	Purpose
ALPHA alpha	Make the next character upper-case
ALPHA alpha ALPHA alpha	With uppercase locked, make next character lowercase
Shift	A toggle button that offers three types of bullet. See Formatting options on page 548
Shift ALPHA alpha	With uppercase locked, make all characters lowercase until the mode is reset
ALPHA alpha	Reset uppercase lock mode
ALPHA alpha Shift	Make the next character lower-case
ALPHA alpha Shift ALPHA alpha	Lock mode: make all characters lowercase until the mode is reset
Shift	With lowercase locked, make next character uppercase
Shift ALPHA alpha	With lowercase locked, make all characters uppercase until the mode is reset
ALPHA alpha	Reset lowercase lock mode

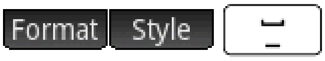
The left side of the notification area of the title bar will indicate what case will be applied to the character you next enter.

Text formatting

You can enter text in different formats in the Note Editor. Choose a formatting option before you start entering text. The formatting options are described in Formatting options on page 548.

Formatting options

Formatting options are available from three touch buttons in the Note Editor and in the Info view of an app:

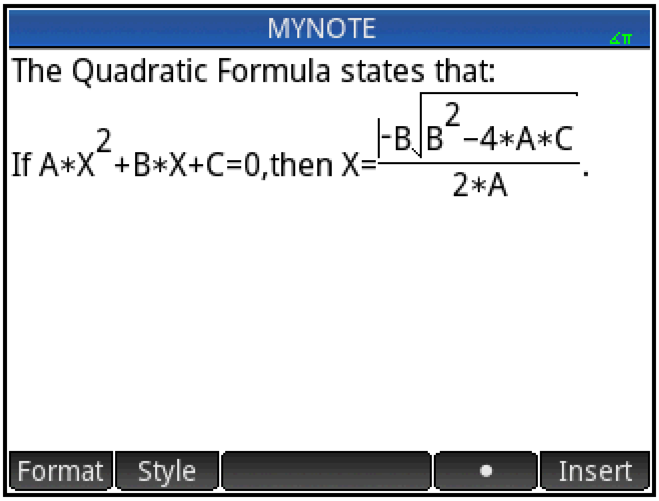


The formatting options are listed in the table below.

Category	Options
Format Font Size	10–22 pt.
Format Foreground Color	Select from 20 colors.
Format Background Color	Select from 20 colors.
Format Align (text alignment)	Left Center Right
Style Font Style	Bold Italic Underline Strikethrough Superscript Subscript
• Bullets	• —First-level bullet ° —Second-level bullet ▷ —Third-level bullet ✗ —Cancels bullet

Inserting mathematical expressions

You can insert a mathematical expression in textbook format into your note, as shown in the following figure. The Note Editor uses the same 2D editor that the Home and CAS views employ, activated via the Insert menu button.



The Quadratic Formula states that:

If $A*X^2+B*X+C=0$,then $X=\frac{-B\sqrt{B^2-4*A*C}}{2*A}$.

1. Enter the text you want. When you come to the point where you want to start a mathematical expression, tap Insert.

2. Enter the mathematical expression just as you would in Home or CAS views. You can use the math template as well as any function in the Toolbox menus.

3. When you have finished entering your mathematical expression, press ▶ 2 or 3 times (depending on the complexity of your expression) to exit the editor. You can now continue to enter text.

To import a note

You can import a note from the Note Catalog into an app's Info view and vice versa.

Suppose you want to copy a note named Assignments from the Note Catalog into the Function Info view:

1. Open the Note Catalog.

   Shift

2. Select the note **Assignments** and tap Edit.

3. Open the copy options for copying to the clipboard.

   Shift View Copy (Copy)

   The menu buttons change to give you options for copying:

   Begin: Marks where the copying or cutting is to begin.

   End: Marks where the copying or cutting is to end.

   All: Select the entire program.

   Cut: Cut the selection.

   Copy: Copy the selection.

4. Select what you want to copy or cut (using the options listed immediately above).

5. Tap Copy or Cut.

6. Open the Info view of the Function app.

   Apps Info, tap the Function app icon, press Shift Apps Info.

7. Move the cursor to the location where you want the copied text to be pasted and open the clipboard.

   Shift ☷Menu Paste

8. Select the text from the clipboard and press OK.

You can send a note to another HP Prime.

28 Programming in HP PPL

This chapter describes the HP Prime Programming Language (HP PPL). In this chapter you'll learn about:

- programming commands
- writing functions in programs
- using variables in programs
- executing programs
- debugging programs
- creating programs for building custom apps
- sending a program to another HP Prime

HP Prime Programs

An HP Prime program contains a sequence of commands that execute automatically to perform a task.

Command Structure

Commands are separated by a semicolon (;). Commands that take multiple arguments have those arguments enclosed in parentheses and separated by a comma(,). For example,

`PIXON` (*xposition, yposition*);

Sometimes, arguments to a command are optional. If an argument is omitted, a default value is used in its place. In the case of the PIXON command, a third argument could be used that specifies the color of the pixel:

`PIXON` (*xposition, yposition [,color]*);

In this manual, optional arguments to commands appear inside square brackets, as shown above. In the PIXON example, a graphics variable (G) could be specified as the first argument. The default is G0, which always contains the currently displayed screen. Thus, the full syntax for the PIXON command is:

`PIXON`(*[G,] xposition, yposition [,color]*);

Some built-in commands employ an alternative syntax whereby function arguments do not appear in parentheses. Examples include `RETURN` and `RANDOM`.

Program Structure

Programs can contain any number of subroutines (each of which is a function or procedure). Subroutines start with a heading consisting of the name, followed by parentheses that contain a list of parameters or arguments, separated by commas. The body of a subroutine is a sequence of statements enclosed within a BEGIN–END; pair. For example, the body of a simple program, called MYPROGRAM, could look like this:

```
EXPORT MYPROGAM()

BEGIN

PIXON(1,1);

END;
```

Comments

When a line of a program begins with two forward slashes, //, the rest of the line will be ignored. This enables you to insert comments in the program:

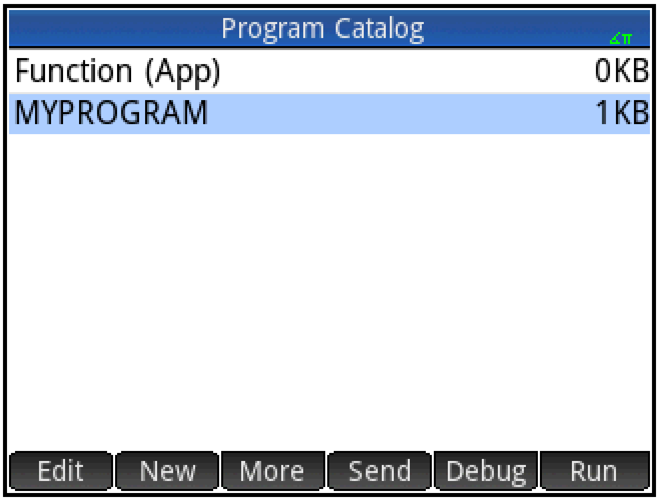
```
EXPORT MYPROGAM()

BEGIN

PIXON(1,1);

//This line is just a comment.

END;
```

The Program Catalog

The Program Catalog is where you run and debug programs, and send programs to another HP Prime. You can also rename and remove programs, and it is where you start the Program Editor. The Program Editor is where you create and edit programs. Programs can also be run from Home view or from other programs.

Open the Program Catalog

Press Shift 1 (Program) to open the Program Catalog.



Program Catalog

Function (App)	0KB
MYPROGRAM	1KB

The Program Catalog displays a list of program names. The first item in the Program Catalog is a built-in entry that has the same name as the active app. This entry is the app program for the active app, if such a program exists.

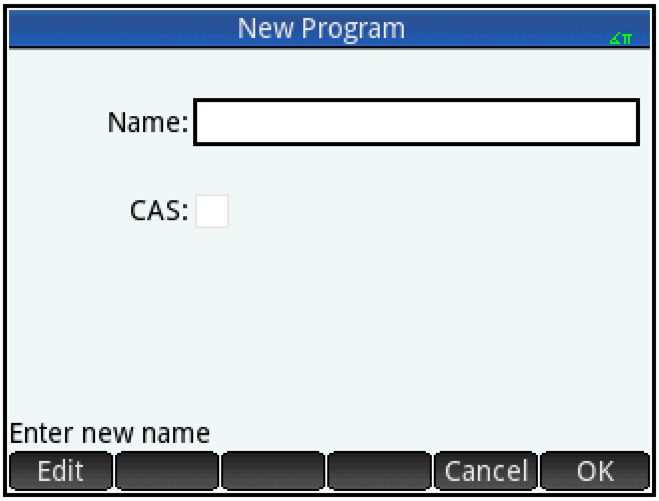
Program Catalog: buttons and keys

Button or Key	Purpose
Edit	Opens the highlighted program for editing.
New	Prompts for a new program name, then opens the Program Editor.
More	Opens further menu options for the selected program: **Save:** Creates a copy of the selected program with a new name you are prompted to give. **Rename:** Renames the selected program. **Sort:** Sorts the list of programs. (Sort options are alphabetical and chronological). **Delete:** Deletes the selected program. **Clear:** Deletes all programs. To redisplay the initial menu, press On or Esc .
Send	Transmits the highlighted program to another HP Prime.
Debug	Debugs the selected program.
Run	Runs the highlighted program.
Shift ▲ or Shift ▼	Moves to the beginning or end of the Program Catalog.
Del	Deletes the selected program.
Shift Esc	Deletes all notes in the catalog.

Creating a new program

In the following few sections, we will create a simple program that counts to three as an introduction to using the Program editor and its menus.

1. Open the Program Catalog and start a new program. Shift 1 (Program) New

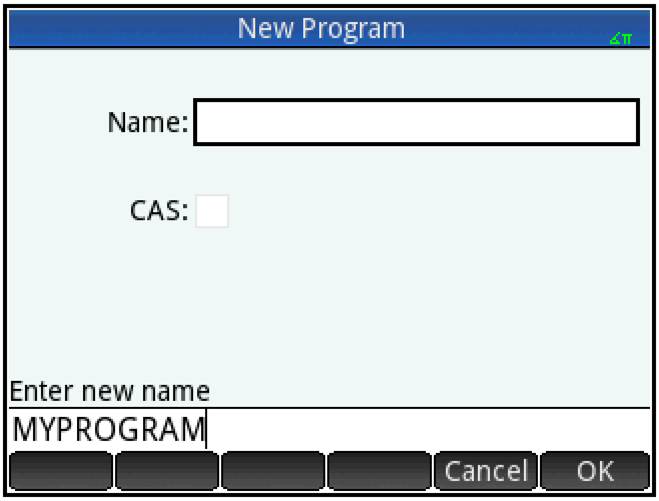

New Program

Name:

CAS: □

Enter new name

2. Enter a name for the program. ALPHA ALPHA (to lock alpha mode) MYPROGRAM OK.


New Program

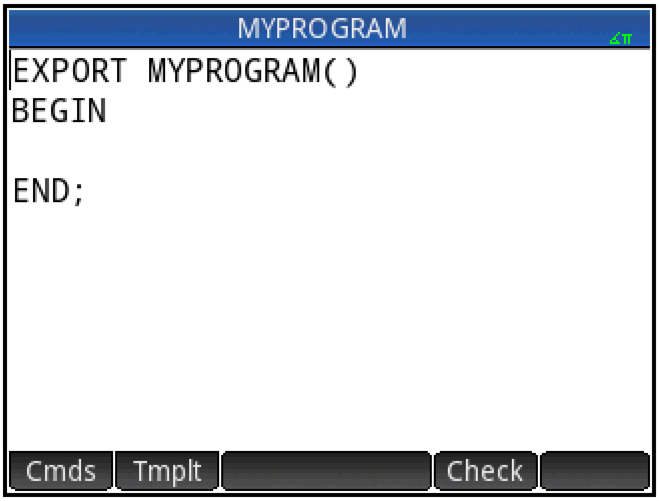
Name:

CAS: □

Enter new name

MYPROGRAM

3. Press OK again. A template for your program is then automatically created. The template consists of a heading for a function with the same name as the program, `EXPORT MYPROGRAM()`, and a `BEGIN-END`; pair that will enclose the statements for the function.



MYPROGRAM

```
EXPORT MYPROGRAM()
BEGIN

END;
```

TIP: A program name can contain only alphanumeric characters (letters and numbers) and the underscore character. The first character must be a letter. For example, `GOOD_NAME` and `Spin2` are valid program names, while `HOT STUFF` (contains a space) and `2Cool!` (starts with number and includes !) are not valid.

The Program Editor

Until you become familiar with the HP Prime commands, the easiest way to enter commands is to select them from the Catalog menu (Mem Catlg), or from the Commands menu in the Program Editor (Cmds). To enter variables, symbols, mathematical functions, units, or characters, use the keyboard keys.

Program Editor: buttons and keys

The buttons and keys in the Program Editor are described in the following table.

Button or Key	Meaning
Check	Checks the current program for errors.
▲ Page ▼ or ALPHA alpha ▲ and ALPHA alpha ▼	If your program goes beyond one screen, you can quickly jump from screen to screen by tapping either side of this button. Tap the left side of the button to display the previous page; tap the right side to display the next page. (The left tap will be inactive if you have the first page of the program displayed.)
Cmds	Opens a menu from which you can choose from common programming commands. The commands are grouped under the options:

Button or Key	Meaning
	Strings Drawing Matrix App Functions Integer I/O More Press Esc Clear to return to the main menu. The commands in this menu are described in Commands under the Cmds menu on page 589.
Tmplt	Opens a menu from which you can select common programming commands. The commands are grouped under the options: Block Branch Loop Variable Function Press Esc Clear to return to the main menu. The commands in this menu are described in Commands under the Tmplt menu on page 583.
Vars Chars A	Displays menus for selecting variable names and values.
Shift Vars Chars A (Char)	Displays a palette of characters. If you display this palette while a program is open, you can choose a character and it will be added to your program at the cursor point. To add one character, highlight it and tap OK or press Enter ≈. To add a character *without* closing the characters palette, select it and tap Echo.
Shift ▶ and Shift ◀	Moves the cursor to the end (or beginning) of the current line. You can also swipe the screen.
Shift ▲ and Shift ▼	Moves the cursor to the start (or end) of the program. You can also swipe the screen.
ALPHA alpha ▶ and ALPHA alpha ◀	Moves the cursor one screen right (or left). You can also swipe the screen.
Enter ≈	Starts a new line.

Button or Key	Meaning
Del	Deletes the character to the left of the cursor.
Shift Del	Deletes the character to the right of the cursor.
Shift Esc Clear	Deletes the entire program.

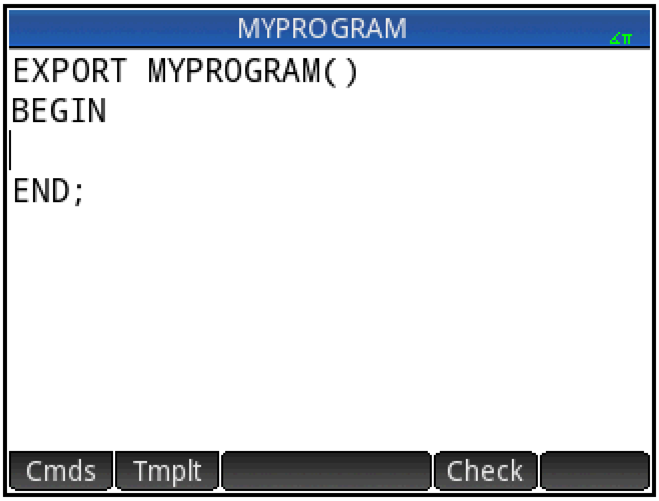
If you press Menu Paste while in the Program Editor, two more options appear:

- **Create user key**—Tap this option and then press any key to paste a template for redefining that key as a user key into your program.
- **Insert pragma**—Tap this option to paste a #pragma mode definition into your program. The #pragma mode definition is in the following form:

  ```
  #pragma mode( separator(), integer())
  ```

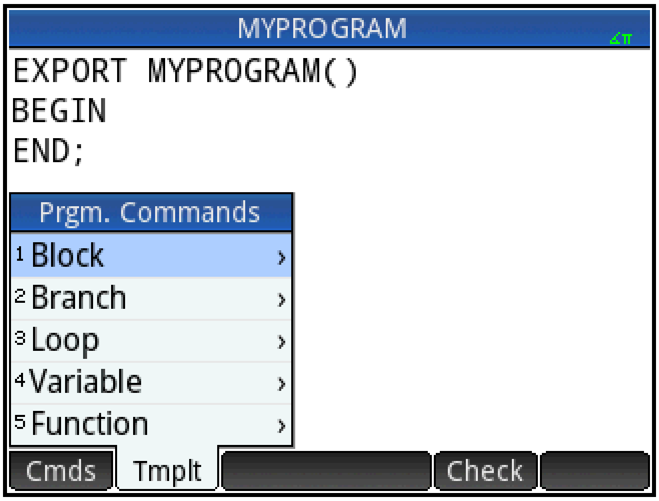
  Use the #pragma mode definition to define the set of separators used for digit grouping and the integer type. The #pragma mode definition forces the program to compile using these settings. This capability is useful for adapting a program written for a culture that uses different grouping symbols (. vs. ,) than your own.

1. To continue the MYPROGRAM example (see Programming in HP PPL on page 552), use the cursor keys to position the cursor where you want to insert a command or just tap on the desired location. In this example, you need to position the cursor between `BEGIN` and `END`.



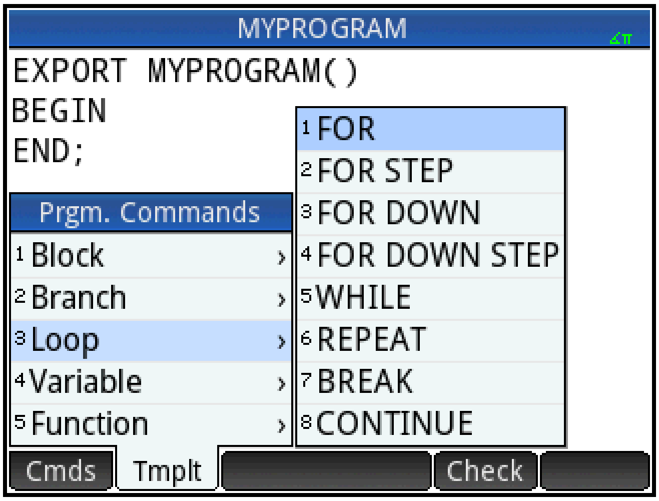
```
EXPORT MYPROGRAM()
BEGIN

END;
```

2. Tap Tmplt to open the menu of common programming commands for blocking, branching, looping, variables, and functions. In this example we'll select a LOOP command from the menu.

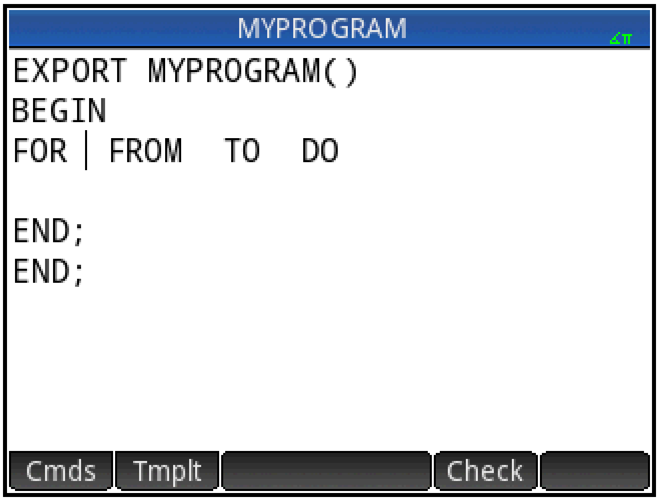


```
EXPORT MYPROGRAM()
BEGIN
END;
```

3. Select **Loop** and then select **FOR** from the sub-menu. Notice that a FOR_FROM_TO_DO_ template is inserted. All you need do is fill in the missing information.

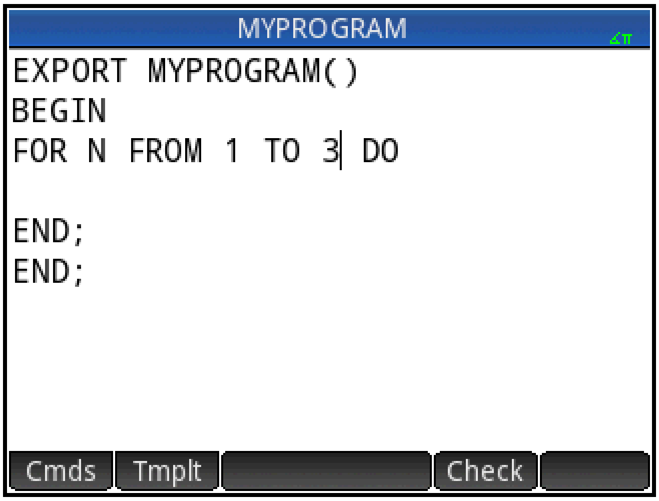


```
EXPORT MYPROGRAM()
BEGIN
END;
```



```
EXPORT MYPROGRAM()
BEGIN
FOR  FROM  TO  DO

END;
END;
```

4. Using the cursor keys and keyboard, fill in the missing parts of the command. In this case, make the statement match the following: FOR N FROM 1 TO 3 DO



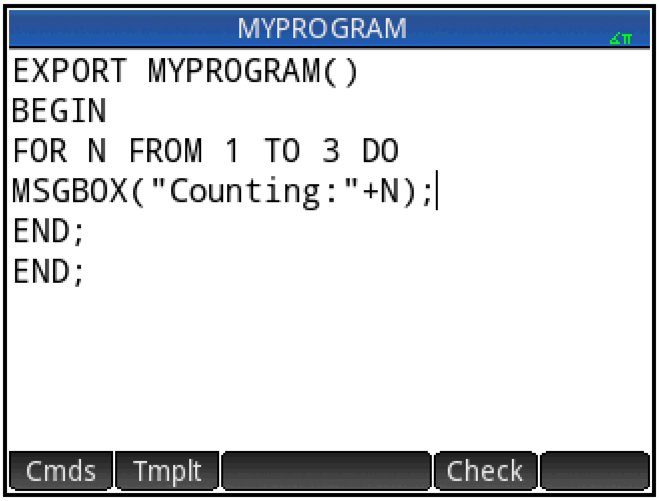
```
EXPORT MYPROGRAM()
BEGIN
FOR N FROM 1 TO 3 DO

END;
END;
```

5. Move the cursor to a blank line below the FOR statement.

6. Tap Cmds to open the menu of common programming commands.

7. Select **I/O** and then select **MSGBOX** from the sub-menu.

8. Fill in the arguments of the MSGBOX command, and type a semicolon at the end of the command (Shift + Ans ;).



```
EXPORT MYPROGRAM()
BEGIN
FOR N FROM 1 TO 3 DO
MSGBOX("Counting:"+N);
END;
END;
```

9. Tap Check to check the syntax of your program.

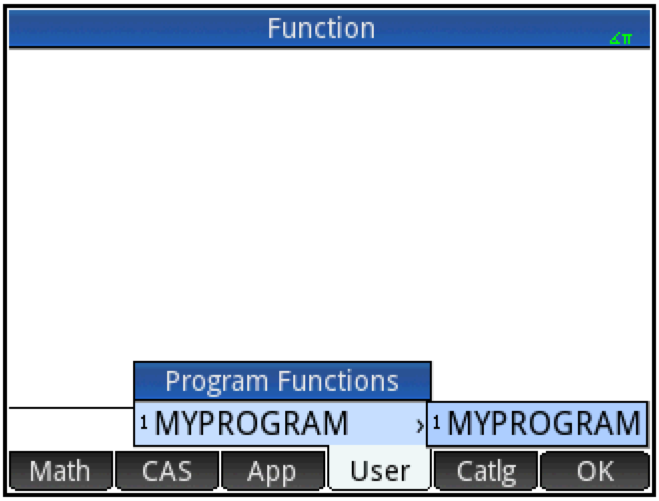
10. When you are finished, press Shift 1 Program Y to return to the Program Catalog or Settings to go to Home view. You are ready now to execute the program.

Run a Program

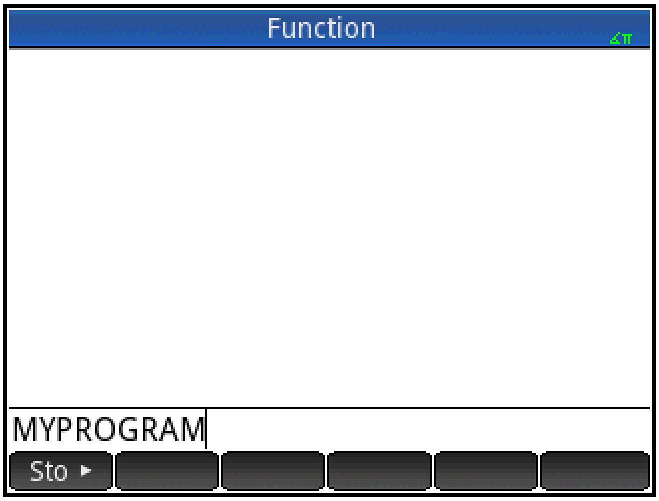
From Home view, enter the name of the program. If the program takes parameters, enter a pair of parentheses after the program name with the parameters inside them each separated by a comma. To run the program, press Enter ≈ .

From the Program Catalog, highlight the program you want to run and tap Run . When a program is executed from the catalog, the system looks for a function named `START()` (no parameters).

You can also run a program from the USER menu (one of the Toolbox menus):



1. Press Mem B and tap User .
2. Tap **MYPROGRAM >** to expand the menu and select **MYPROGRAM** .`MYPROGRAM` appears on the entry line.
3. Tap Enter ≈ and the program executes, displaying a message box.
4. Tap OK three times to step through the FOR loop. Notice that the number shown increments by 1 each time.


MYPROGRAM

After the program terminates, you can resume any other activity with the HP Prime.

If a program has arguments, when you press Run a screen appears prompting you to enter the program parameters.

Multi-function programs

▲ To create an entry that has multiple subentries in the User submenu of the Toolbox menu, enter multiple EXPORT commands in a single program.

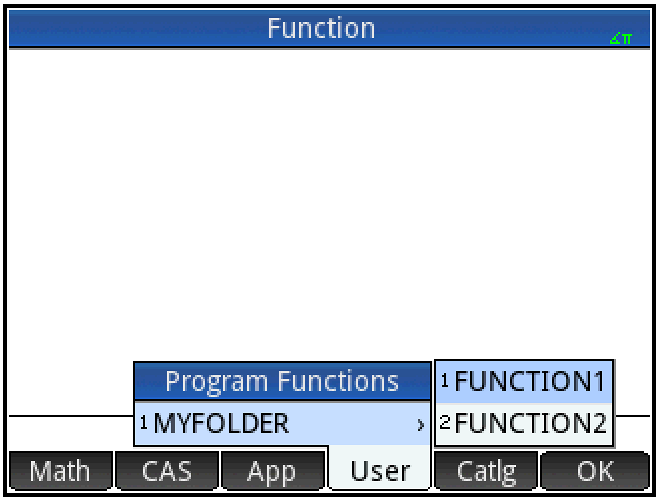
NOTE: Typically, you must delete the automatic EXPORT, BEGIN, and END commands that are created with a program.

The following example program is named MYFOLDER. It contains two user-defined functions as follows:

- FUNCTION1(X) returns X+1
- FUNCTION2(X) returns X−1

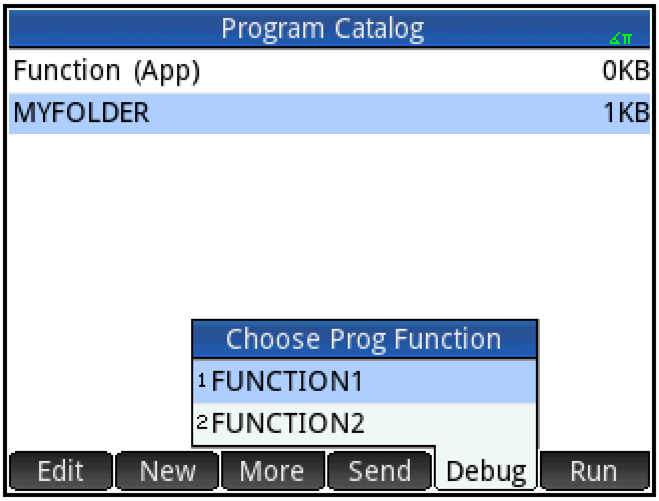
```
Program MYFOLDER
EXPORT FUNCTION1(X)
BEGIN
RETURN X+1;
END;
EXPORT FUNCTION2(X)
BEGIN
RETURN X-1;
END;
```

Now, when you press Mem and then tap User , an option named MYFOLDER appears. Tap **MYFOLDER** to see the subentries FUNCTION1 and FUNCTION2.



You can use this procedure to create custom folders that contain the functions you need and are organized optimally for your use.

When you select your program from the Program Catalog and tap Run or Debug, a list with **NAME1** and **NAME2** appears. Select which function to run or debug.



Program Catalog

Function (App)	0KB
MYFOLDER	1KB

Choose Prog Function

1 FUNCTION1

2 FUNCTION2

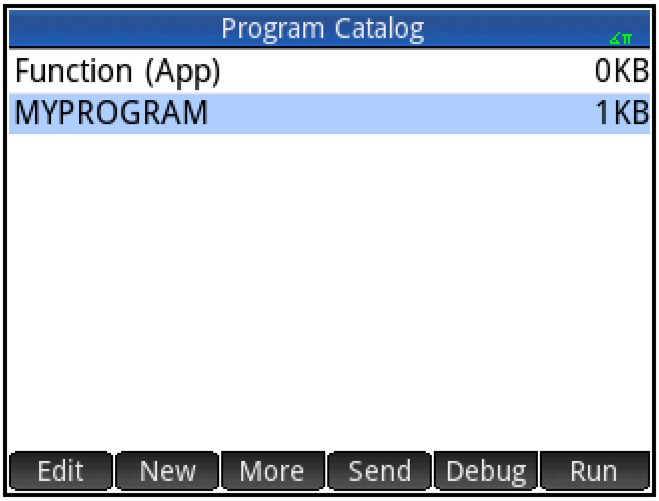
Debug a Program

You cannot run a program that contains syntax errors. If the program does not do what you expect it to do, or if there is a run-time error detected by the system, you can execute the program step by step, and look at the values of local variables.

Let’s debug the program created above: MYPROGRAM.

1. In the Program Catalog, select MYPROGRAM.

   Shift 1 Program Y

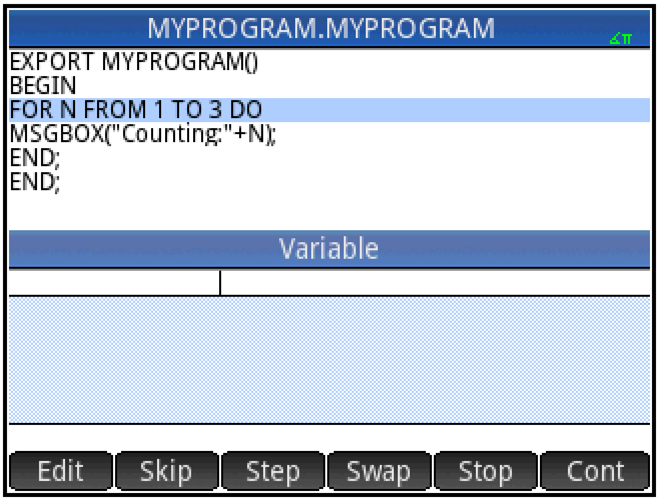


Program Catalog

Function (App)	0KB
MYPROGRAM	1KB

2. Tap Debug.

   If there is more than one EXPORT function in a file, a list appears for you to choose which function to debug.

   ```
   EXPORT MYPROGRAM()
   BEGIN
   FOR N FROM 1 TO 3 DO
   MSGBOX("Counting:"+N);
   END;
   END;
   ```

   While debugging a program, the title of the program or intra-program function appears at the top of the display. Below that is the current line of the program being debugged. The current value of each variable is visible in the main body of the screen. The following menu buttons are available in the debugger:

   Skip: Skips to the next line or block of the program

   Step: Executes the current line

   Vars: Opens a menu of variables. You can select one and add it to the list of variables so you can see how it changes as you step through the program.

   Stop: Closes the debugger

   Cont: Continues program execution without debugging

3. Execute the FOR loop command Step.

   The FOR loop starts and the top of the display shows the next line of the program (the MSGBOX command).

4. Execute the MSGBOX command Step.

   The message box appears. Note that when each message box is displayed, you still have to dismiss it by tapping OK or pressing Enter.

   Tap Step and press Enter repeatedly to execute the program step-by-step.

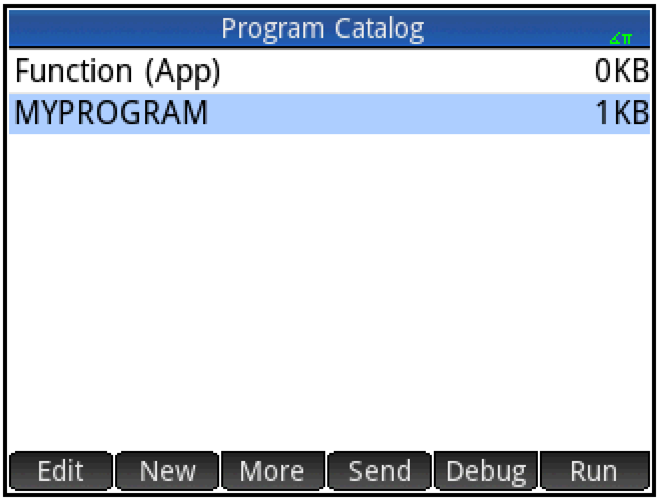
Tap Stop to close the debugger at the current line of the program, or tap Cont to run the rest of the program without using the debugger.

Edit a Program

You edit a program using the Program Editor, which is accessible from the Program Catalog.

1. Open the Program Catalog.

   Shift 1 Program Y

Program Catalog	
Function (App)	0KB
MYPROGRAM	1KB

2. Tap the program you want to edit (or use the arrow keys to highlight it and press Enter ≈).

   The HP Prime opens the Program Editor. The name of your program appears in the title bar of the display. The buttons and keys you can use to edit your program are listed in Program Editor: buttons and keys on page 556.

Copy a program or part of a program

You can use the global `Copy` and `Paste` commands to copy part or all of a program. The following steps illustrate the process:

1. Open the Program Catalog.

   Shift 1 Program Y

2. Tap the program that has the code you want to copy.

3. Press Shift View Copy (Copy).

   The menu buttons change to give you options for copying:

   Begin : Marks where the copying or cutting is to begin.

   End : Marks where the copying or cutting is to end.

   All : Select the entire program.

   Cut : Cut the selection.

Copy : Copy the selection.

4. Select what you want to copy or cut (using the options listed immediately above).

5. Tap Copy or Cut .

6. Return to the Program Catalog and open the target program.

7. Move the cursor to where you want to insert the copied or cut code.

8. Press Shift Menu Paste (Paste). The clipboard opens. What you most recently copied or cut will be first in the list and highlighted already, so just tap OK . The code will be pasted into the program, beginning at the cursor location.

Delete a program

To delete a program:

1. Open the Program Catalog.

   Shift 1 Program Y

2. Highlight a program to delete and press Del .

3. At the prompt, tap OK to delete the program or Cancel to cancel.

Delete all programs

To delete all programs at once:

1. Open the Program Catalog.

   Shift 1 Program Y

2. Press Shift Esc Clear (Clear).

3. At the prompt, tap OK to delete all programs or Cancel to cancel.

Delete the contents of a program

You can clear the contents of a program without deleting the program. The program then just has a name and nothing else.

1. Open the Program Catalog.

   Shift 1 Program Y

2. Tap the program to open it.
3. Press Shift Esc Clear (Clear).

To share a program

You can send programs between calculators just as you can send apps, notes, matrices, and lists.

The HP Prime programming language

The HP Prime programming language allows you to extend the capabilities of the HP Prime by adding programs, functions and variables to the system. The programs you write can be either standalone or attached to an app. The functions and variables you create can be either local or global. If they are declared to be global, then they appear in the User menu when you press Mem B or Vars Chars A. In the following sections, we discuss variables and functions, then create a set of short programs to illustrate the various techniques for creating programs, functions, and variables.

Variables and visibility

Variables in an HP Prime program can be used to store numbers, lists, matrices, graphics objects, and strings. The name of a variable must be a sequence of alphanumeric characters (letters and numbers), starting with a letter. Names are case-sensitive, so the variables named `MaxTemp` and `maxTemp` are different.

The HP Prime has built-in variables of various types, visible globally (that is, visible wherever you are in the calculator). For example, the built-in variables `A` to `Z` can be used to store real numbers, `Z0` to `Z9` can be used to store complex numbers, `M0` to `M9` can be used to store matrices and vectors, and so on. These names are reserved. You cannot use them for other data. For example, you cannot name a program `M1`, or store a real number in a variable named `Z8`. In addition to these reserved variables, each HP app has its own reserved variables. Some examples are `Root`, `Xmin`, and `Numstart`. Most of these app variables are local to their app, though a few are global by design. For example, `C1` is used by the Statistics 2Var app to store statistical data. This variable is global so that you can access that data from anywhere in the system. Again, these names cannot be used to name a program or store data of a type other than their design allows. (A full list of system and app variables is given in the “Variables” chapter.)

In a program you can declare variables for use only within a particular function. This is done using a `LOCAL` declaration. The use of local variables enables you to declare and use variables that will not affect the rest of the calculator. Local variables are not bound to a particular type; that is, you can store floating-point numbers, integers, lists, matrices, and symbolic expressions in a variable with any local name. Although the system will allow you to store different types in the same local variable, this is poor programming practice and should be avoided.

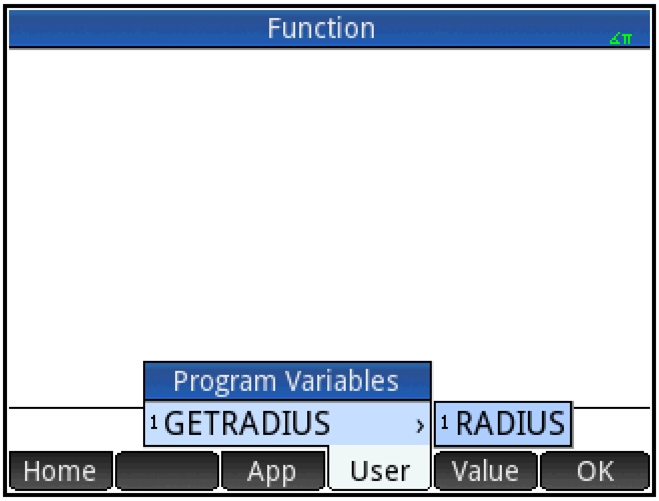
Variables declared in a program should have descriptive names. For example, a variable used to store the radius of a circle is better named `RADIUS` than `VGFTRFG`. You are more likely to remember what the variable is used for if its name matches its purpose.

If a variable is needed after the program executes, it can be exported from the program using the EXPORT command. To do this, the first command in the program (that is, on a line above the program name) would be EXPORT RADIUS. Then, if a value is assigned to RADIUS, the name appears on the variables menu (Vars Chars A) and is visible globally. This feature allows for extensive and powerful interactivity among different environments in the HP Prime. Note that if another program exports a variable with the same name, the most recently exported version will be active.

The program below prompts the user for the value of RADIUS, and exports the variable for use outside the program.

```
EXPORT RADIUS;

EXPORT GETRADIUS()

BEGIN

INPUT(RADIUS);

END;
```

Note that EXPORT command for the variable RADIUS appears before the heading of the function where RADIUS is assigned. After you execute this program, a new variable named RADIUS appears on the USER GETRADIUS section of the Variables menu.



Qualifying the name of a variable

The HP Prime has many system variables with names that are apparently the same. For example, the Function app has a variable named Xmin, but so too do the Polar app, the Parametric app, the Sequence app, and the Solve app. In a program, and in the Home view, you can refer to a particular version of these variables by qualifying its name. This is done by entering the name of the app (or program) that the variable belongs to, followed by a dot (.), and then the actual variable name. For example, the qualified variable Function.Xmin refers to the value of Xmin within the Function app. Similarly, the qualified variable Parametric.Xmin refers to the value of Xmin in the Parametric app. Despite having the same name (Xmin), the variables could have different values. You do likewise to use a local variable in a program: specify the name of the program, followed by the dot, and then the variable name.

Functions, their arguments, and parameters

You can define your own functions in a program, and data can be passed to a function using parameters. Functions can return a value (using the RETURN statement) or not. When a program is executed from Home view, the program will return the value returned by the last statement that was executed.

Furthermore, functions can be defined in a program and exported for use by other programs, in much the same way that variables can be defined and used elsewhere.

In this section, we will create a small set of programs, each illustrating some aspect of programming in the HP Prime. Each program will be used as a building block for a custom app.

Program ROLLDIE

We'll first create a program called ROLLDIE. It simulates the rolling of a single die, returning a random integer between 1 and whatever number is passed into the function.

In the Program Catalog create a new program named ROLLDIE. (For help, see Creating a new program on page 554.) Then enter the code in the Program Editor.

```
EXPORT ROLLDIE(N)

BEGIN

RETURN 1+RANDINT(N-1);

END;
```

The first line is the heading of the function. Execution of the RETURN statement causes a random integer from 1 to N to be calculated and returned as the result of the function. Note that the RETURN command causes the execution of the function to terminate. Thus any statements between the RETURN statement and END are ignored.

In Home view (in fact, anywhere in the calculator where a number can be used), you can enter ROLLDIE(6) and a random integer between 1 and 6 inclusive will be returned.

Program ROLLMANY

Because of the EXPORT command in ROLLDIE, another program could use the ROLLDIE function and generate n rolls of a die with any number of sides. In the following program, the ROLLDIE function is used to generate *n* rolls of two dice, each with the number of sides given by the local variable sides. The results are stored in list L2, so that L2(1) shows the number of times the dies came up with a combined total of 1, L2(2) shows the number of times the dies came up with a combined total of 2, etc. L2(1) should be 0 (since the sum of the numbers on 2 dice must be at least 2).

Here we use the Store operator (▶) in place of :=. Press Shift EEX Sto▸ P to retrieve this operator. The syntax is Var ▶ Value; that is, the value on the right is stored in the variable on the left.

```
EXPORT ROLLMANY(n,sides)

BEGIN

 LOCAL k,roll;

 // initialize list of frequencies

 MAKELIST(0,X,1,2*sides,1) ▶ L2;
 FOR k FROM 1 TO n DO
ROLLDIE(sides)+ROLLDIE(sides) ▶ roll;
 L2(roll)+1 ▶ L2(roll);
 END;
END;
```

By omitting the `EXPORT` command when a function is declared, its visibility can be restricted to the program within which it is defined. For example, you could define the `ROLLDIE` function inside the ROLLMANY program like this:

```
ROLLDIE();
EXPORT ROLLMANY(n,sides)
BEGIN
 LOCAL k,roll;
 // initialize list of frequencies
 MAKELIST(0,X,1,2*sides,1) ▶ L2;
 FOR k FROM 1 TO n DO
ROLLDIE(sides)+ROLLDIE(sides) ▶ roll;
 L2(roll)+1 ▶ L2(roll);
 END;
END;
ROLLDIE(n)
BEGIN
RETURN 1+RANDINT(n-1);
END;
```

In the second version of the ROLLMANY program, there is no `ROLLDIE` function exported from another program. Instead, ROLLDIE is visible only to `ROLLMANY`. The `ROLLDIE` function must be declared before it is called. The first line of the program above contains the declaration of the `ROLLDIE` function. The definition of the `ROLLDIE` function is located at the end of the program.

Finally, the list of results could be returned as the result of calling `ROLLMANY` instead of being stored directly in the global list variable, L2. This way, if the user wanted to store the results elsewhere, it could be done easily.

```
ROLLDIE();
EXPORT ROLLMANY(n,sides)
BEGIN
 LOCAL k,roll,results;
 // initialize list of frequencies
 MAKELIST(0,X,1,2*sides,1) ▶ results;
 FOR k FROM 1 TO n DO
ROLLDIE(sides)+ROLLDIE(sides) ▶ roll;
 results(roll)+1 ▶ results(roll);
 END;
RETURN results;
END;
ROLLDIE(N)
BEGIN
RETURN 1+RANDINT(N-1);
END;
```

In Home view you would enter `ROLLMANY(100,6) ▶ L5` and the results of the simulation of 100 rolls of two six-sided dice would be stored in list L5.

The User Keyboard: Customizing key presses

You can assign alternative functionality to any key on the keyboard, including to the functionality provided by the shift and alpha keys. This enables you to customize the keyboard to your particular needs. For example, you could assign SIN ASIN G to a function that is multi-nested on a menu and thus difficult to get to on a menu (such as ALOG).

A customized keyboard is called the user keyboard and you activate it when you go into user mode.

User mode

There are two user modes:

- Temporary user mode: the next key press, and only the next, enters the object you have assigned to that key. After entering that object, the keyboard automatically returns to its default operation.

  To activate temporary user mode, press Shift Help User (User). Notice that **1U** appears in the title bar. The **1** will remind you that the user keyboard will be active for just one key press.

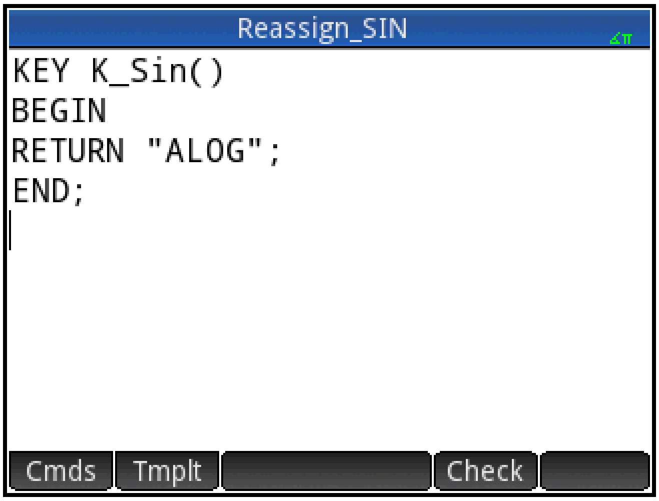
- Persistent user mode: each key press from now *until you turn off user mode* will enter whatever object you have assigned to a key.

  To activate persistent user mode, press Shift Help User Shift Help User. Notice that ↑**U** appears in the title bar. The user keyboard will now remain active until you press Shift Help User again.

If you are in user mode and press a key that hasn't been reassigned, the key's standard operation is performed.

Reassigning keys

Suppose you want to assign a commonly used function—such as ALOG—to its own key on the keyboard. Simply create a new program that mimics the syntax in the following figure.



```
KEY K_Sin()
BEGIN
RETURN "ALOG";
END;
```

The first line of the program specifies the key to be reassigned using its internal name. (The names of all the keys are given in Key names on page 572. They are case-sensitive.)

On line 3, enter the text you want produced when the key being reassigned is pressed. This text must be enclosed in quote marks.

The next time you want to insert `ALOG` at the position of your cursor, you just press Shift Help User SIN ASIN G.

You can enter any string you like in the RETURN line of your program. For example, if you enter "Newton", that text will be returned when you press the reassigned key. You can even get the program to return user-defined functions as well as system functions, and user-defined variables as well as system variables.

You can also reassign a shifted key combination. So, for example, ALPHA alpha Shift ÷ x⁻¹ T could be reassigned to produce `SLOPE(F1(X),3)` rather than the lowercase t. Then if ALPHA alpha Shift ÷ x⁻¹ T is entered in Home view and Enter ≈ pressed, the gradient at X = 3 of whatever function is currently defined as F1(X) in the Function app would be returned.

TIP: A quick way to write a program to reassign a key is to press Menu Paste and select **Create user key** when you are in the Program Editor. You will then be asked to press the key (or key combination) you want to reassign. A program template appears, with the internal name of the key (or key combination) added automatically.

Key names

The first line of a program that reassigns a key must specify the key to be reassigned using its internal name. The table below gives the internal name for each key. Note that key names are case-sensitive.

Internal name of keys and key states

Key	Name	Shift + key	ALPHA alpha + key	ALPHA alpha Shift + key
Apps Info	K_Apps	KS_Apps	KA_Apps	KSA_Apps
Symb Setup	K_Symb	KS_Symb	KA_Symb	KSA_Symb
▲	K_Up	KS_Up	KA_Up	KSA_Up
Help User	K_Help	—	KA_Help	KSA_Help
Esc Clear	K_Esc	KS_Esc	KA_Esc	KSA_Esc
⌂ Settings	K_Home	KS_Home	KA_Home	KSA_Home
Plot Setup	K_Plot	KS_Plot	KA_Plot	KSA_Plot
◀	K_Left	KS_Left	KA_Left	KSA_Left
▶	K_Right	KS_Right	KA_Right	KSA_Right
View Copy	K_View	KS_View	KA_View	KSA_View
CAS Settings	K_Cas	KS_Cas	KA_Cas	KSA_Cas
Num Setup	K_Num	KS_Num	KA_Num	KSA_Num
▼	K_Down	KS_Down	KA_Down	KSA_Down
Menu Paste	K_Menu	KS_Menu	KA_Menu	KSA_Menu
Vars Chars A	K_Vars_	KS_Vars_	KA_Vars_	KSA_Vars_
Mem B	K_Math	KS_Math	KA_Math	KSA_Math
Units C	K_Templ	KS_Templ	KA_Templ	KSA_Templ

Internal name of keys and key states				
Key	Name	Shift + key	ALPHA alpha + key	ALPHA alpha Shift + key
xtθn Define D	K_Xttn	KS_Xttn	KA_Xttn	KSA_Xttn
a b/c °′″ E	K_Abc	KS_Abc	KA_Abc	KSA_Abc
⌫ Del	K_Bksp	KS_Bksp	KA_Bksp	KSA_Bksp
x^y $\sqrt[y]{x}$ F	K_Power	KS_Power	KA_Power	KSA_Power
SIN ASIN G	K_Sin	KS_Sin	KA_Sin	KSA_Sin
COS ACOS H	K_Cos	KS_Cos	KA_Cos	KSA_Cos
TAN ATAN I	K_Tan	KS_Tan	KA_Tan	KSA_Tan
LN e^x J	K_Ln	KS_Ln	KA_Ln	KSA_Ln
LOG 10^x K	K_Log	KS_Log	KA_Log	KSA_Log
x^2 √ L	K_Sq	KS_Sq	KA_Sq	KSA_Sq
+/− |x| M	K_Neg	KS_Neg	KA_Neg	KSA_Neg
( ) '□' N	K_Paren	KS_Paren	KA_Paren	KSA_Paren
, x Eval O	K_Comma	KS_Comma	KA_Comma	KSA_Comma
Enter ≈	K_Ente	KS_Enter	KA_Enter	KSA_Enter
EEX Sto ▸ P	K_Eex	KS_Eex	KA_Eex	KSA_Eex
7 List Q	K_7	KS_7	KA_7	KSA_7
8 { } R	K_8	KS_8	KA_8	KSA_8

Internal name of keys and key states				
Key	Name	Shift + key	ALPHA alpha + key	ALPHA alpha Shift + key
9 !,∞,→ S	K_9	KS_9	KA_9	KSA_9
÷ x^{-1} T	K_Div	KS_Div	KA_Div	KSA_Div
ALPHA alpha	K_Alpha	KS_Alpha	KA_Alpha	KSA_Alpha
4 Matrix U	K_4	KS_4	KA_4	KSA_4
5 [ ] V	K_5	KS_5	KA_5	KSA_5
6 ≤,≥,≠ W	K_6	KS_6	KA_6	KSA_6
× ∡ X	K_Mul	KS_Mul	KA_Mul	KSA_Mul
Shift	—	—	—	—
1 Program Y	K_1	KS_1	KA_1	KSA_1
2 i Z	K_2	KS_2	KA_2	KSA_2
3 π #	K_3	KS_3	KA_3	KSA_3
− Base ±	K_Minus	KS_Minus	KA_Minus	KSA_Minus
On Off	K_On	—	KA_On	KSA_On
0 Notes " "	K_0	KS_0	KA_0	KSA_0
. =	K_Dot	KS_Dot	KA_Dot	KSA_Dot

Internal name of keys and key states				
Key	Name	Shift + key	ALPHA alpha + key	ALPHA alpha Shift + key
␣ _	K_Space	KS_Space	KA_Space	KSA_Space
+ Ans ;	K_Plus	KS_Plus	KA_Plus	KSA_Plus

App programs

An app is a unified collection of views, programs, notes, and associated data. Creating an app program allows you to redefine the app’s views and how a user will interact with those views. This is done with (a) dedicated program functions with special names and (b) by redefining the views in the **View** menu.

Using dedicated program functions

There are nine dedicated program function names, as shown in the table below. These functions are called when the corresponding keys shown in the table are pressed. These functions are designed to be written into a program that controls an app and used in the context of that app.

Program	Name	Equivalent Keystrokes
Symb	Symbolic view	Symb Setup
SymbSetup	Symbolic Setup	Shift Symb Setup
Plot	Plot view	Plot Setup
PlotSetup	Plot Setup	Shift Plot Setup
Num	Numeric view	Num Setup
NumSetup	Numeric Setup	Shift Num Setup
Info	Info view	Shift Apps Info
START	Starts an app	Start
RESET	Resets or initializes an app	Reset

Redefining the View menu

The **View** menu allows any app to define views in addition to the standard seven views shown in the table above. By default, each HP app has its own set of additional views contained in this menu. The `VIEW` command allows you to redefine these views to run programs you have created for an app. The syntax for the `VIEW` command is:

```
VIEW "text", function()
```

By adding `VIEW "text", function()` before the declaration of a function, you will override the list of views for the app. For example, if your app program defines three views—"SetSides", "RollDice" and "PlotResults"—when you press View Copy you will see SetSides, RollDice, and PlotResults instead of the app's default view list.

Customizing an app

When an app is active, its associated program appears as the first item in the Program Catalog. It is within this program that you put functions to create a custom app. A useful procedure for customizing an app is illustrated below:

1. Decide on the HP app that you want to customize. The customized app inherits all the properties of the HP app.
2. Go to the Applications Library (Apps Info), highlight the HP app, tap Save and save the app with a unique name.
3. Customize the new app if you need to (for example, by configuring the axes or angle measure settings).
4. Open the Program Catalog, select your new app program, and tap Edit .
5. Develop the functions to work with your customized app. When you develop the functions, use the app naming conventions described above.
6. Put the `VIEW` command in your program to modify the app's View menu.
7. Decide if your app will create new global variables. If so, you should `EXPORT` them from a separate user program that is called from the `Start()` function in the app program. This way they will not have their values lost.
8. Test the app and debug the associated programs.

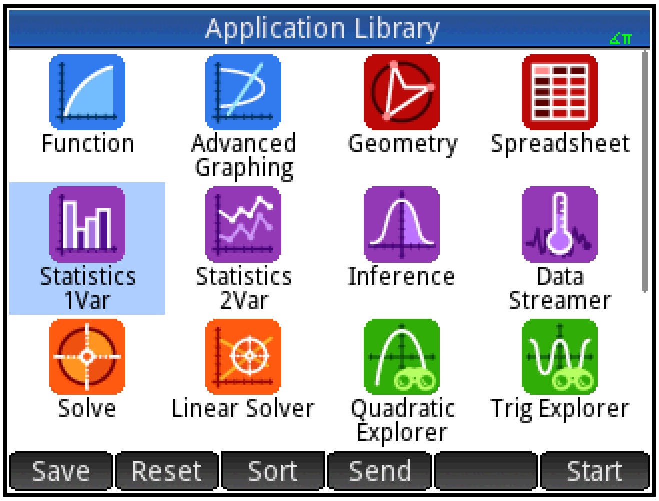
It is possible to link more than one app via programs. For example, a program associated with the Function app could execute a command to start the Statistics 1Var app, and a program associated with the Statistics 1Var app could return to the Function app (or launch any other app).

Example

The following example illustrates the process of creating a custom app. The app is based on the built-in Statistics 1Var app. It simulates the rolling of a pair of dice, each with a number of sides specified by the user. The results are tabulated, and can be viewed either in a table or graphically.

1. In the Application Library, select the Statistics 1Var app but don't open it.

   Apps Info Select **Statistics 1Var**.



2. Tap Save.

3. Enter a name for the new app (such as `DiceSimulation`.)

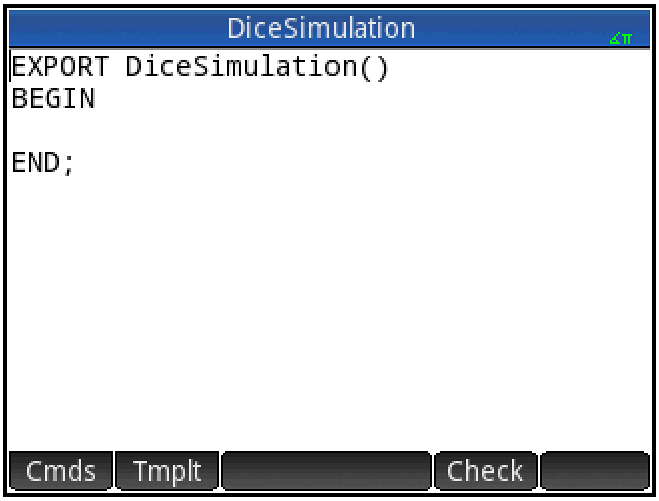
4. Tap OK twice. The new app appears in the Application Library.

5. Open the Program Catalog.

   Shift 1 Program

6. Tap the program to open it.

   Each customized app has one program associated with it. Initially, this program is empty. You customize the app by entering functions into that program.



```
EXPORT DiceSimulation()
BEGIN

END;
```

At this point you decide how you want the user to interact with the app. In this example, we will want the user to be able to:

- start and initialize the app, and display a short note
- specify the number of sides (that is, faces) on each die
- specify the number of times to roll the dice

- graphically display the results of the simulation
- numerically display the results of the simulation.

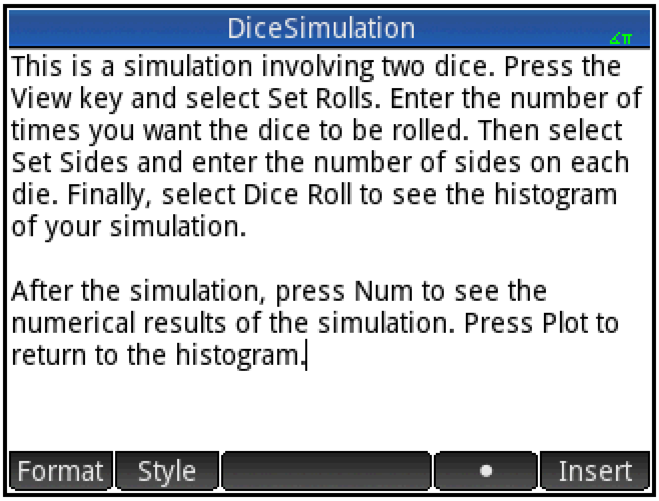
With that in mind, we will create the following views:

START, ROLL DICE, SET SIDES, and SET ROLLS.

The START option will initialize the app and display a note that gives the user instructions. The user will also interact with the app through the Numeric view and the Plot view.

These views will be activated by pressing Num and Plot, but the function `Plot()` in our app program will actually launch the latter view after doing some configuration.

Before entering the following program, press Shift Apps to open the Info editor and enter the text shown in the figure. This note will be attached to the app and will be displayed when the user selects the Start option from the View menu (or presses Shift Apps).



This is a simulation involving two dice. Press the View key and select Set Rolls. Enter the number of times you want the dice to be rolled. Then select Set Sides and enter the number of sides on each die. Finally, select Dice Roll to see the histogram of your simulation.

After the simulation, press Num to see the numerical results of the simulation. Press Plot to return to the histogram.

The program discussed earlier in this chapter to get the number of sides for a dice is expanded here, so that the possible sums of two such die are stored in dataset D1. Enter the following sub-routines into the program for the DiceSimulation app.

The DiceSimulation program

```
DICESIMVARS();

ROLLDIE();

 EXPORT SIDES,ROLLS;

EXPORT DiceSimulation()

BEGIN

END;

VIEW "Start",START()

BEGIN

 D1:={};
 D2:={};
 SetSample(H1,D1);
 SetFreq(H1,D2);
 H1Type:=1;
 STARTVIEW(6,1);
END;
VIEW "Roll Dice",ROLLMANY()
BEGIN
 LOCAL k,roll;
 D1:= MAKELIST(X+1,X,1,2*SIDES-1,1);
 D2:= MAKELIST(0,X,1,2*SIDES-1,1);
 FOR k FROM 1 TO ROLLS DO
 roll:=ROLLDIE(SIDES)+ROLLDIE
 (SIDES);
 D2(roll-1):= D2(roll-1)+1;
 END;
 Xmin:= -0.1;
 Xmax:= MAX(D1)+1;
 Ymin:= −0.1;
 Ymax:= MAX(D2)+1;
 STARTVIEW(1,1);
END;
VIEW "Set Sides",SETSIDES()
BEGIN
 REPEAT
 INPUT(SIDES,"Die Sides","N=","Enter# of sides",2);
 SIDES:= FLOOR(SIDES);
 IF SIDES<2 THEN MSGBOX("# of sides must be >= 4");
 END;
 UNTIL SIDES >=4;
 STARTVIEW(7,1);
END;

VIEW "Set Rolls",SETROLLS()
BEGIN
```

```
 REPEAT
 INPUT(ROLLS,"Num of rolls","N=","Enter# of rolls",25);
 ROLLS:= FLOOR(ROLLS);
 IF ROLLS<1 THEN MSGBOX("You must enter a num >=1");
 END;
 UNTIL ROLLS>=1;
 STARTVIEW(7,1);
END;
PLOT()
BEGIN
 Xmin:=-0.1;
 Xmax:= MAX(D1)+1;
 Ymin:= −0.1;
 Ymax:= MAX(D2)+1;
 STARTVIEW(1,1);
END;
Symb()
BEGIN
 SetSample(H1,D1);
 SetFreq(H1,D2);
 H1Type:=1;
 STARTVIEW(0,1);
END;
```

The `ROLLMANY()` routine is an adaptation of the program presented earlier in this chapter. Since you cannot pass parameters to a program called through a selection from a custom View menu, the exported variables `SIDES` and `ROLLS` are used in place of the parameters that were used in the previous versions.

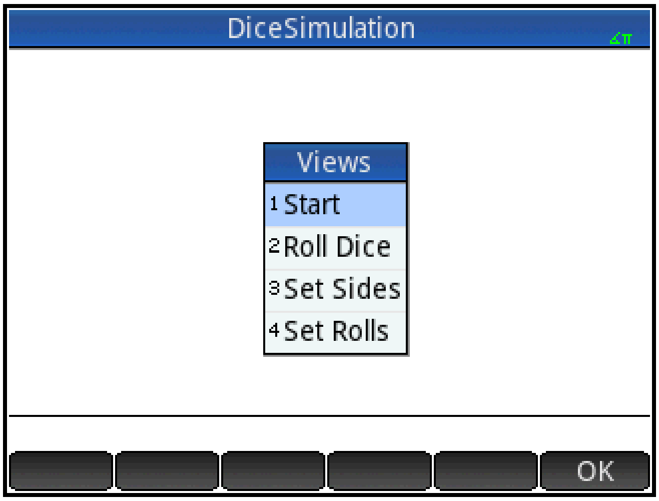
The program above calls two other user programs: ROLLDIE() and DICESIMVARS(). ROLLDIE() appears earlier in this chapter. Here is DICESIMVARS. Create a program with that name and enter the following code.

The program DICESIMVARS

```
EXPORT ROLLS,SIDES;
EXPORT DICESIMVARS()
BEGIN
10 ► ROLLS;
6 ► SIDES;
END;
```

1. Press Apps Info, and open DiceSimulation. The note will appear explaining how the app works.

2. Press View Copy to see the custom app menu. Here you can reset the app (Start), set the number of sides of the dice, the number of rolls, and execute a simulation.



DiceSimulation

Views

1 Start

2 Roll Dice

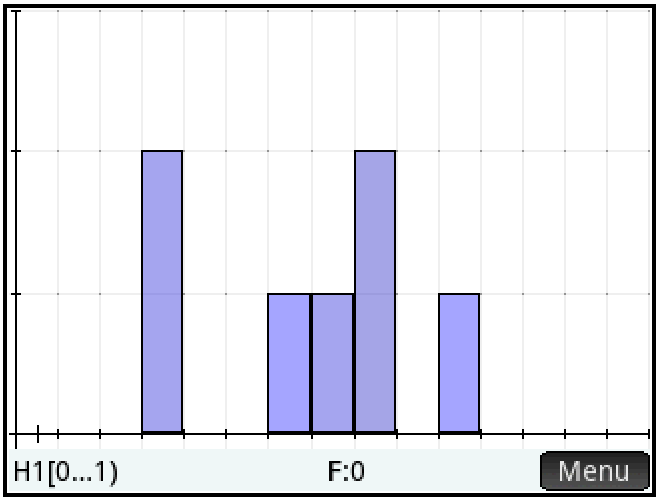
3 Set Sides

4 Set Rolls

3. Select **Set Rolls** and enter 100.

4. Select **Set Sides** and enter 6.

5. Select **Roll Dice**. You will see a histogram similar to the own shown in the figure.



H1[0...1)   F:0

6. Press Num Setup to see the data and Plot Setup to return to the histogram.

7. To run another simulation, press View Copy and select **Roll Dice**.

Program commands

This section describes each program command. The commands under the Tmplt menu are described first. The commands under the Cmds menu are described in Commands under the Cmds menu on page 589.

Commands under the Tmplt menu

Block

The block commands determine the beginning and end of a sub-routine or function. There is also a Return command to recall results from sub-routines or functions.

BEGIN END

Syntax: `BEGIN command1; command2;…; commandN; END;`

Defines a command or set of commands to be executed together. In the simple program:

```
EXPORT SQM1(X)

BEGIN

RETURN X^2-1;

END;
```

the block is the single RETURN command.

If you entered `SQM1(8)` in Home view, the result returned would be 63.

RETURN

Syntax: `RETURN` *expression*;

Returns the current value of *expression*.

KILL

Syntax: `KILL`;

Stops the step-by-step execution of the current program (with debug).

Branch

In what follows, the plural word *commands* refers to both a single command or a set of commands.

IF THEN

Syntax: `IF` *test* `THEN` *commands* `END`;

Evaluate *test*. If *test* is true (not 0), executes *commands*. Otherwise, nothing happens.

IF THEN ELSE

Syntax: `IF` *test* `THEN` *commands 1* `ELSE` *commands 2* `END`;

Evaluate *test*. If *test* is true (not 0), it executes *commands 1*, otherwise, it executes *commands 2*.

If *test* returns a list, *commands 1* and *commands 2* must return a single object or both must return a list that is the same size as the list returned by *test*.

If *commands 1* or *commands 2* both return a list, each list is the same size and each element is picked from either *commands 1* or *commands 2*, depending on the outcome of *test* on the elements of the test list.

CASE

Syntax:

```
CASE
 IF test1 THEN commands1 END;
 IF test2 THEN commands2 END;
...
[ DEFAULT commands]
END;
```

Evaluates *test1*. If true, executes *commands1* and ends the CASE. Otherwise, evaluates *test1*. If true, executes *commands2* and ends the CASE. Continues evaluating tests until a true is found. If no true test is found, execute default commands, if provided. The CASE command is limited to 127 branches.

Example:

```
CASE
IF A<0 THEN RETURN "negative"; END;
IF 0≤A≤1 THEN RETURN "small"; END;
DEFAULT RETURN "large";
END;
```

IFERR

`IFERR` *commands1* `THEN` *commands2* `END`;

Executes sequence of *commands1*. If an error occurs during execution of *commands1*, executes sequence of *commands2*.

NOTE: The error number is stored in the variable *Ans*. You can use this variable in the *commands2* syntax of the THEN clause of the IFERR command.

IFERR ELSE

`IFERR` *commands1* `THEN` *commands2* `ELSE` *commands3* `END`;

Executes sequence of *commands1*. If an error occurs during execution of *commands1*, executes sequence of *commands2*. Otherwise, execute sequence of *commands3*.

Loop

FOR

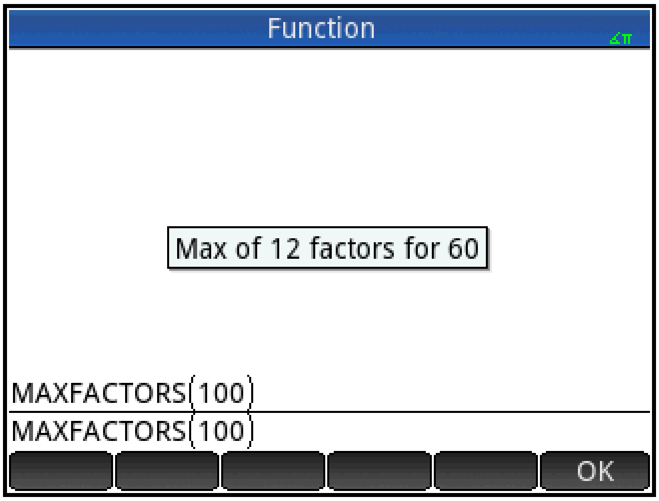
Syntax: `FOR` *var* `FROM` *start* `TO` *finish* `DO` commands END;

Sets variable *var* to *start*, and for as long as this variable is less than or equal to *finish*, executes the sequence of *commands*, and then adds 1 (*increment*) to *var*.

Example 1: This program determines which integer from 2 to N has the greatest number of factors.

```
EXPORT MAXFACTORS(N)

BEGIN

LOCAL cur,max,k,result;

1 ▶ max;1 ▶ result;

FOR k FROM 2 TO N DO

 SIZE(CAS.idivis(k)) ▶ cur;

 IF cur(1) > max THEN

 cur(1) ▶ max;

 k ▶ result;

 END;

END;

MSGBOX("Max of "+ max +" factors for "+result);

END;
```

In Home, enter `MAXFACTORS(100).`



Max of 12 factors for 60

MAXFACTORS(100)

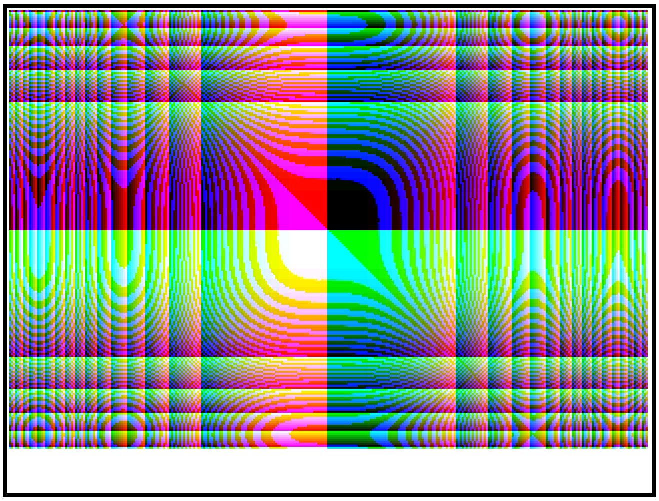
MAXFACTORS(100)

FOR STEP

Syntax: `FOR` *var* `FROM` *start* `TO` *finish* [*STEP increment*] `DO` commands END;

Sets variable *var* to *start*, and for as long as this variable is less than or equal to *finish*, executes the sequence of commands, and then adds *increment* to *var*.

Example 2: This program draws an interesting pattern on the screen.



```
EXPORT
DRAWPATTERN()
BEGIN
LOCAL
xincr,yincr,color;
STARTAPP("Function");
RECT();
xincr := (Xmax - Xmin)/318;
yincr := (Ymax - Ymin)/218;
FOR X FROM Xmin TO Xmax STEP xincr DO
FOR Y FROM Ymin TO Ymax STEP yincr DO
color := RGB(X^3 MOD 255,Y^3 MOD 255, TAN(0.1*(X^3+Y^3)) MOD 255);
PIXON(X,Y,color);
END;
END;
WAIT;
END;
```

FOR DOWN

Syntax: `FOR` *var* `FROM` ***start*** `DOWNTO` ***finish*** `DO` *commands END*;

Sets variable *var* to *start*, and for as long as this variable is more than or equal to *finish*, executes the sequence of commands, and then subtracts 1 (decrement) from *var*.

FOR STEP DOWN

Syntax: `FOR` *var* `FROM` ***start*** `DOWNTO` ***finish*** [*STEP increment*] `DO` *commands END*;

Sets variable *var* to *start*, and for as long as this variable is more than or equal to *finish*, executes the sequence of commands, and then subtracts *increment* from *var*.

WHILE

Syntax: `WHILE` ***test*** `DO` ***commands*** `END`;

Evaluates `test`. If result is true (not 0), executes the *commands*, and repeats.

Example: A perfect number is one that is equal to the sum of all its proper divisors. For example, 6 is a perfect number because 6 = 1+2+3. The example below returns true when its argument is a perfect number.

```
EXPORT ISPERFECT(n)
BEGIN
 LOCAL d, sum;
 2 ▶ d;
 1 ▶ sum;
WHILE sum <= n AND d < n DO
 IF irem(n,d)==0 THEN sum+d ▶ sum;
 END;
 d+1 ▶ d;
 END;
 RETURN sum==n;
END;
```

The following program displays all the perfect numbers up to 1000:

```
EXPORT PERFECTNUMS()
BEGIN
LOCAL k;
FOR k FROM 2 TO 1000 DO
 IF ISPERFECT(k) THEN
 MSGBOX(k+" is perfect, press OK");
 END;
END;
END;
```

REPEAT

Syntax: `REPEAT` commands `UNTIL` test;

Repeats the sequence of commands until test is true (not 0).

The example below prompts for a positive value for SIDES, modifying an earlier program in this chapter:

```
EXPORT SIDES;
EXPORT GETSIDES()
BEGIN
 REPEAT
 INPUT(SIDES,"Die Sides","N = ","Enter num sides",2);
 UNTIL SIDES>0;
END;
```

BREAK

Syntax: `BREAK(n)`

Exits from loops by breaking out of n loop levels. Execution picks up with the first statement after the loop. With no argument, exits from a single loop.

CONTINUE

Syntax: `CONTINUE`

Transfers execution to the start of the next iteration of a loop

Variable

These commands enable you to control the visibility of a user-defined variable.

LOCAL

Syntax: `LOCAL var1,var2,…varn;`

Makes the variables var1, var2, etc. local to the program in which they are found.

EXPORT

Syntax: `EXPORT var1, [var2, …, varn];`

– or –

```
EXPORT var1:=val1, [var2:=val2, … varn:=valn];
```

Exports the variables *var1*, *var2*, etc. so they are globally available and appear on the **User** menu when you press Vars and select User.

Example:

```
EXPORT ratio:=0.15;
```

Function

These commands enable you to control the visibility of a user-defined function.

EXPORT

Syntax: `EXPORT FunctionName(Parameters)`

– or –

```
EXPORT FunctionName(Parameters)
```

```
BEGIN

FunctionDefinition

END;
```

In a program, declares the functions or variables to export globally. The exported functions appear in the Toolbox User menu and the exported variables appear in the Vars CAS, App, and User menus.

Examples:

```
EXPORT X2M1(X);

Export X2M1(X)

BEGIN

RETURN X^2-1;

END;
```

VIEW

Syntax: `VIEW “text”, functionname();`

Replaces the **View** menu of the current app and adds an entry with “text”. If “text” is selected and the user presses OK or Enter ≈, then `functionname()` is called.

KEY

A prefix to a key name when creating a user keyboard. See The User Keyboard: Customizing key presses on page 571.

Commands under the Cmds menu

Strings

A string is a sequence of characters enclosed in double quotes (""). To put a double quote in a string, use two consecutive double quotes. The \ character starts an escape sequence, and the character(s) immediately following are interpreted specially. \n inserts a new line and two backslashes insert a single backslash. To put a new line into the string, press Enter ≈ to wrap the text at that point.

ASC

Syntax: `ASC` (string)

Returns a list containing the ASCII codes of string.

Example: `ASC` ("AB") returns [65,66]

LOWER

Converts uppercase characters in a string to lowercase.

Examples:

`LOWER("ABC")` returns "abc"

`LOWER("ΑΒΓ")` returns "αβγ"

UPPER

Converts lowercase characters in a string to uppercase.

Examples:

`UPPER("abc")` returns "ABC"

`UPPER("αβγ")` returns "ABΓ"

CHAR

Syntax: `CHAR(vector)` or `CHAR(integer)`

Returns the string corresponding to the character codes in `vector`, or the single code of `integer`.

Examples: `CHAR(65)` returns "A"

`CHAR([82,77,72])` returns "RMH"

DIM

Syntax: `DIM`(string)

Returns the number of characters in string.

Example: `DIM("12345")` returns 5, `DIM("""")` and `DIM("\n")` return 1. (Notice the use of the two double quotes and the escape sequence.)

STRING

Syntax: `STRING(Expression, [Mode], [Precision], [Separator]` or `{Separator, ["[DecimalPoint[Exponent[NegativeSign]]]"], [DotZero]}], [SizeLimit]` or `{SizeLimit, [FontSize], [Bold], [Italic], [Monospaced]}]`

Evaluates Expression and returns the result as a string.

The extra parameters specify how numbers are displayed.

If Mode is specified, it must be:

0: Use current setting

1: Standard

2: Fixed

3: Scientific

4: Engineering

5: Floating

6: Rounded

Add 7 to this value to specify proper fraction mode and 14 for mixed fraction mode.

Precision is either -1 for current settings or 0 to 12.

Separator is a string containing a set of digits and separators. The last digit is assumed to be the one just before the decimal point. Separator can also be a number. -1 means use default, 0 to 10 specify the use of one of the 11 built-in separators available in home settings.

“[DecimalPoint[Exponent[NegativeSign]]]” is a string of 0 to 3 characters. The first one will be used for the decimal point, the second for the exponent and the last one for the negative sign.

If DotZero is non-zero, then numbers are displayed in the form .1 instead of 0.1

If SizeLimit is specified, the command will attempt to generate a representation of the number that fits in the given number of pixels. You can also specify the font size (10 to 22) and properties (bold, italic and mono-spaced being Boolean values with 0 being false). There is no guarantee that the result will fit, but the command will attempt to make it fit.

Examples:

String	Result
string(F1), when F1(X) = COS(X)	"COS(X)"
STRING(2/3)	0.666666666667
string(L1) when L1 = {1,2,3}	"{1,2,3}"
string(M1) when M1 = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	"[[1,2,3],[4,5,6]]"

INSTRING

Syntax: INSTRING (str1,str2)

Returns the index of the first occurrence of *str2* in *str1*. Returns 0 if *str2* is not present in *str1*. Note that the first character in a string is position 1.

Examples:

INSTRING ("vanilla","van") returns 1

INSTRING ("banana","na") returns 3

INSTRING ("ab","abc") returns 0

LEFT

Syntax: LEFT (str,n)

Return the first *n* characters of string *str*. If *n* ≥ DIM(str) or *n* < 0, returns str. If *n* == 0 returns the string.

Example: LEFT("MOMOGUMBO",3) returns "MOM"

RIGHT

Syntax: RIGHT(str,n)

Returns the last *n* characters of string *str*. If *n* <= 0, returns empty string. If *n* > DIM(str), returns str

Example: RIGHT("MOMOGUMBO",5) returns "GUMBO"

MID

Syntax: MID(str,pos, [n])

Extracts *n* characters from string *str* starting at index pos. *n* is optional, if not specified, extracts all the remainder of the string.

Example: `MID("MOMOGUMBO",3,5)` returns "MOGUM", `MID("PUDGE",4)` returns "GE"

ROTATE

Syntax: `ROTATE(str,n)`

Permutation of characters in string *str*. If 0 <=*n* < DIM(str), shifts *n* places to left. If –DIM(str) < *n* <= –1, shifts *n* spaces to right. If *n* > DIM(str) or *n* < –DIM(str), returns *str*.

Examples:

`ROTATE("12345",2)` returns "34512"

`ROTATE("12345",-1)` returns "51234"

`ROTATE("12345",6)` returns "12345"

STRINGFROMID

Syntax: `STRINGFROMID`(integer)

Returns, in the current language, the built-in string associated in the internal string table with the specified *integer*.

Examples:

STRINGFROMID(56) returns “Complex”

STRINGFROMID(202) returns “Real”

REPLACE

Syntax: `REPLACE(object1, start, object2)`

Replaces part of object1 with object2 beginning at start. The objects can be matrices, vectors, or stings.

Example:

`REPLACE("12345",”3”,”99”)` returns "12995"

Drawing

There are 10 built-in graphics variables in the HP Prime, called G0–G9. G0 is always the current screen graphic.

G1 to G9 can be used to store temporary graphic objects (called GROBs for short) when programming applications that use graphics. They are temporary and thus cleared when the calculator turns off.

Twenty-six functions can be used to modify graphics variables. Thirteen of them work with Cartesian coordinates using the Cartesian plane defined in the current app by the variables Xmin, Xmax, Ymin, and Ymax.

The remaining thirteen work with pixel coordinates where the pixel 0,0 is the top left pixel of the GROB, and 320, 240 is the bottom right. Functions in this second set have a _P suffix to the function name.

C→PX

Converts from Cartesian coordinates to screen coordinates.

Syntax: `C→PX(x,y)` or `C→PX({x,y})`

DRAWMENU

Syntax: `DRAWMENU({string1, string2, …, string6})`

Draws a six-button menu at the bottom of the display, with labels string1, string2, …, string6.

Example:

`DRAWMENU("ABC", "", "DEF")` creates a menu with the first and third buttons labelled ABC and DEF, respectively. The other four menu keys are blank.

FREEZE

Syntax: `FREEZE`

Pauses program execution until a key is pressed. This prevents the screen from being redrawn after the end of the program execution, leaving the modified display on the screen for the user to see.

PX→C

Converts from screen coordinates to Cartesian coordinates.

RGB

Syntax: `RGB(R, G, B, [A])`

Returns an integer number that can be used as the color parameter for a drawing function, based on Red-, Green-, and Blue-component values (each 0 to 255).

If Alpha is greater than 128, returns the color flagged as transparent. There is no alpha channel blending on Prime.

Examples:

`RGB(255,0,128)` returns 16711808.

`RECT(RGB(0,0,255))` makes a blue screen

`LINE(0,0,8,8,RGB(0,255,0))` draws a green line

Pixels and Cartesian

ARC_P, ARC

Syntax; `ARC(G, x, y, r [ , a1, a2, c])`

Syntax; `ARC_P(G, x, y, r [ , a1, a2, c])`

Draws an arc or circle on G, centered on point x,y, with radius r and color c starting at angle a1 and ending on angle a2.

G can be any of the graphics variables and is optional. The default is G0

r is given in pixels.

c is optional and if not specified black is used. It should be specified in this way: #RRGGBB (in the same way as a color is specified in HTML).

a1 and a2 follow the current angle mode and are optional. The default is a full circle.

Example:

`ARC(0,0,60,0,π,RGB(255,0,0))` draws a red semicircle with center at (0,0)—using the current Plot Setup window—and with a radius of 60 pixels. The semicircle is drawn counterclockwise from 0 to π.

BLIT_P, BLIT

Syntax: `BLIT([trgtGRB, dx1, dy1, dx2, dy2], [srcGRB, sx1, sy1, sx2, sy2, c, alpha])`

Syntax: `BLIT_P ([trgtGRB, dx1, dy1, dx2, dy2], [srcGRB, sx1, sy1, sx2, sy2, c, alpha])`

Copies the region of srcGRB from (and including) point (sx1, sy1) to (but not including) point (sx2, sy2), into the region of trgtGRB between points (dx1, dy1) and (dx2, dy2). In practice, 1 is added to each of sx1 and sx2 to obtain the correct region. Does not copy pixels from srcGRB that are color c.

The number alpha must be between 0 (transparent) and 255 (opaque). It represents the transparency, or alpha channel, of the source bitmap.

trgtGRB can be any of the graphics variables and is optional. The default is G0.

srcGRB can be any of the graphics variables.

dx2, dy2 are optional and, if not specified, are calculated so that the destination area is the same size as the source area.

sx2, sy2 are optional and, if not specified, are the bottom-right corner of the srcGRB.

sx1, sy1 are optional and, if not specified, are the top-left corner of the srcGRB.

dx1, dy1 are optional and, if not specified, are the top-left corner of trgtGRB.

c can be any color specified as #RRGGBB. If it is not specified, all pixels from srcGRB will be copied.

alpha is optional and, if not specified, is 255 (opaque) by default.

NOTE: Using the same variable for trgtGRB and srcGRB can be unpredictable when the source and destination overlap.

If you use both c and alpha, HP recommends that you also specify the source x- and y- coordinates to be sure that the system can distinguish the purpose of each parameter.

DIMGROB_P, DIMGROB

Syntax: `DIMGROB_P(G, w, h, [color])` or `DIMGROB_P(G, list)`

Syntax: `DIMGROB(G, w, h, [color])` or `DIMGROB(G, list)`

Sets the dimensions of GROB G to w × h. Initializes the graphic G with color or with the graphic data provided by the list variable. If the graphic is initialized using graphic data, then list is a list of integers. Each integer, as seen in base 16, describes one color every 16 bits.

Colors are in A1R5G5B5 format (that is,1 bit for the alpha channel, and 5 bits for R, G, and B).

FILLPOLY_P, FILLPOLY

Syntax: `FILLPOLY_P([G],{(x1, y1), (x2, y2),…(xn, yn)}, Color, [Alpha])`

Syntax: `FILLPOLY([G],{(x1, y1), (x2, y2),…(xn, yn)}, Color, [Alpha])`

For the polygon defined by the list of points, fills the polygon with the color defined by the RGB number Color. If Alpha is provided as an integer between 0 and 255 inclusive, the polygon is drawn with the corresponding transparency level. You can use a vector of points instead of a list; in this case, the points can be expressed as complex numbers.

Example:

`FILLPOLY_P({(20,20), (100, 20), (100, 100), (20, 100)}, #FF, 128)` draws a square, 80 pixels on a side, near the upper left of the display, using the color purple and transparency level 128.

GETPIX_P, GETPIX

Syntax: `GETPIX([G], x, y)`

Syntax: `GETPIX_P([G], x, y)`

Returns the color of the pixel G with coordinates x,y.

G can be any of the graphics variables and is optional. The default is G0, the current graphic.

GROBH_P, GROBH

Syntax: `GROBH(G)`

Syntax: `GROBH_P(G)`

Returns the height of G.

G can be any of the graphics variables and is optional. The default is G0.

GROBW_P, GROB

Syntax: `GROBW(G)`

Syntax: `GROBW_P(G)`

Returns the width of G.

G can be any of the graphics variables and is optional. The default is G0.

INVERT_P, INVERT

Syntax: `INVERT([G, x1, y1, x2, y2])`

Syntax: `INVERT_P([G, x1, y1, x2, y2])`

Executes a reverse video of the selected region. G can be any of the graphics variables and is optional. The default is G0.

x2, y2 are optional and if not specified will be the bottom right of the graphic.

x1, y1 are optional and if not specified will be the top left of the graphic. If only one x,y pair is specified, it refers to the top left.

LINE_P, LINE

Syntax: `LINE_P([G], x1, y1, x2, y2, [color])`

Syntax: `LINE_P([G],points_definition, lines_definitions, otation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{3Dxmin, 3Dxmax, 3Dymin, 3Dymax, 3Dzmin, 3Dzmax}]}, [zstring])`

Syntax: `LINE_P([G],pre_rotated_points, line_definitions, [zstring])` Syntax: `LINE([G], x1, y1, x2, y2, [color])`

Syntax: `LINE([G],points_definition, lines_definitions, otation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{3Dxmin, 3Dxmax, 3Dymin, 3Dymax, 3Dzmin, 3Dzmax}]}, [zstring])`

Syntax: `LINE([G],pre_rotated_points, line_definitions, [zstring])`

The basic form of LINE_P draws one line between specified pixel coordinates in the graphic using the specified color.

The advanced form of LINE_P allows the multiple lines to be rendered at the same time with a potential 3D transformation of the triangle's vertices.

This is mostly used if you have a set of vertices and lines and want to display them all at once (faster).

`points_definition` is either a list or a matrix of point definitions. Each point is defined by two to four numbers: x, y, z, and color. A valid point definition can have multiple forms. Here are some examples: [x, y, z, c], {x, y, z, c}, {x, y, #c}, {(x, y), c}, (x,y). You can use a vector of points instead of a list; in this case, the points can be expressed as complex numbers.

`line_ definitions` is either a list or a matrix of line definitions. Each line is defined by two to four numbers: p1, p2, color and alpha. p1 and p2 are the index in the `points_definition` of the two points that define the line. Color is used to override the per point color definition. If you need to provide an Alpha, but not a color, use -1 for the color.

Note that `{Color, [Alpha], line_1, ..., line_n}` is also a valid form to avoid respecifying the same color for each line.

`rotation_matrix` is a matrix between the sizes 2*2 to 3*4 that specifies the rotation and translation of the point using usual 3D or 4D geometry.

`{eye_x, eye_y, eye_z}` defines the eye position (projection).

`{3Dxmin, 3Dxmax, 3Dymin, 3Dymax, 3Dzmin, 3Dzmax}` is used to perform 3D clipping on the pretransformed objects.

Each point is rotated and translated through a multiplication by the `rotation_matrix`. It is then projected on the view plan using the eye position calculated by the following equation: x=eye_z/z*x-eye_x and y=eye_z/ z*y-eye_y.

Each line is clipped in 3D, if 3D clipping data is provided.

If "N" is specified, the Z coordinates are normalized between 0 and 255 after rotation, providing easier zClipping.

If zstring is provided, per pixel z clipping happens using the z value string (see the following).

`LINE_P` returns a string that contains all the transformed points. If you plan to call TRIANGLE or LINE multiple times in a row using the same points and transformation, you can do so by replacing the points_definition with this string and omitting the transformation definition in subsequent calls to TRIANGLE and LINE.

About zstring:

`TRIANGLE_P([G])` returns a string adapted for z clipping.

To use z clipping, call `TRIANGLE_P` to create a z clipping string (initialized at 255 for each pixels). You can then call LINE_P with appropriate z (0-255) values for each of the triangle vertices and `LINE_P` will not draw pixels farther than the already drawn pixels. ZString is automatically updated as appropriate.

PIXOFF_P, PIXOFF

Syntax: `PIXOFF([G], x, y)`

Syntax: `PIXOFF_P([G], x, y)`

Sets the color of the pixel G with coordinates x,y to white. G can be any of the graphics variables and is optional. The default is G0, the current graphic.

PIXON_P, PIXON

Syntax: `PIXON([G], x, y [ ,color])`

Syntax: `PIXON_P([G], x, y [ ,color])`

Sets the color of the pixel in graphic variable G with coordinates (x,y) to the entered color. G can be any of the graphics variables and is optional. The default is G0, the current graphic.

The optional color can be any hexadecimal integer entered in the form aaRRGGBB. This is an RGB color with the alpha channel in the high order byte. The alpha channel numbers can be any integer between 0 (opaque) and 255 (transparent). If no color is specified, the default black is used.

RECT_P, RECT

Syntax: `RECT([G, x1, y1, x2, y2, edgecolor, fillcolor])`

Syntax: `RECT_P([G, x1, y1, x2, y2, edgecolor, fillcolor])`

Draws a rectangle on G between points x1,y1 and x2,y2 using edgecolor for the perimeter and fillcolor for the inside.

G can be any of the graphics variables and is optional. The default is G0, the current graphic.

x1, y1 are optional. The default values represent the top left of the graphic.

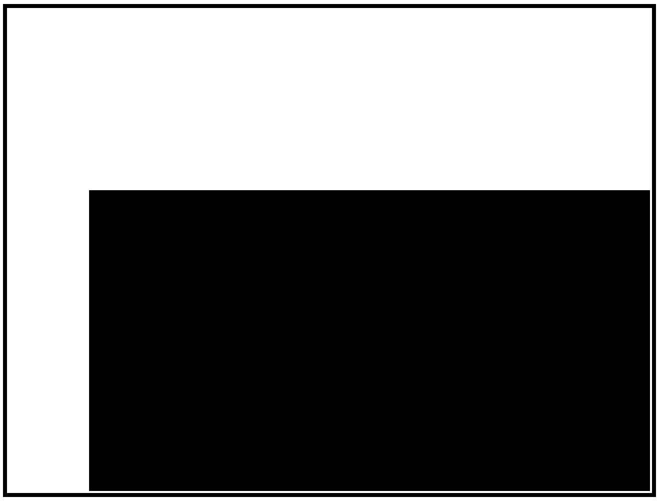
x2, y2 are optional. The default values represent the bottom right of the graphic.

edgecolor and fillcolor can be any color specified as #RRGGBB. Both are optional, and fillcolor defaults to edgecolor if not specified.

To erase a `GROB`, execute `RECT(G)`. To clear the screen execute `RECT()`.

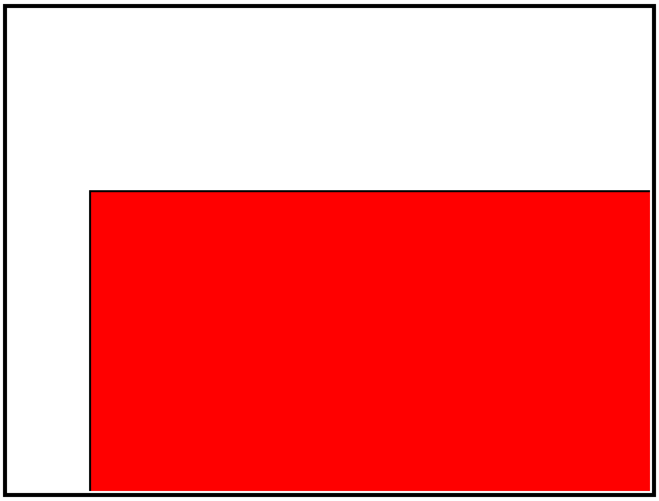
When optional arguments are provided in a command with multiple optional parameters (like `RECT`), the arguments provided correspond to the leftmost parameters first. For example, in the program below, the arguments `40` and `90` in the `RECT_P` command correspond to x1 and y1. The argument `#000000` corresponds to edgecolor, since there is only the one additional argument. If there had been two additional arguments, they would have referred to x2 and y2 rather than edgecolor and fillcolor. The program produces a rectangle with a black edge and black fill.

```
EXPORT BOX()

BEGIN

RECT();

RECT_P(40,90,#0 00000);

WAIT;

END;
```



The program below also uses the RECT_P command. In this case, the pair of arguments 320 and 240 correspond to x2 and y2. The program produces are rectangle with a black edge and a red fill.

```
EXPORT BOX()

BEGIN

RECT();

RECT_P(40,90,32 0,240,#000000,# FF0000);

WAIT;

END;
```



SUBGROB_P, SUBGROB

Syntax: `SUBGROB(srcGRB [ ,x1, y1, x2, y2], trgtGRB)`

Syntax: `SUBGROB_P(srcGRB [ ,x1, y1, x2, y2], trgtGRB)`

Sets trgtGRB to be a copy of the area of srcGRB between points x1,y1 and x2,y2.

srcGRB can be any of the graphics variables and is optional. The default is G0.

trgtGRB can be any of the graphics variables except G0.

x2, y2 are optional and if not specified will be the bottom right of srcGRB.

x1, y1 are optional and if not specified will be the top left of srcGRB.

Example: `SUBGROB(G1, G4)` will copy G1 in G4.

TEXTOUT_P, TEXTOUT

Syntax: `TEXTOUT(text [ ,G], x, y [ ,font, c1, width, c2])`

Syntax: `TEXTOUT_P(text [ ,G], x, y [ ,font, c1, width, c2])`

Draws text using color c1 on graphic G at position x, y using font. Do not draw text more than width pixels wide and erase the background before drawing the text using color c2.

G can be any of the graphics variables and is optional. The default is G0. This command returns the x-coordinate of the pixel at the end of the text output.

Font can be:

0: current font selected on the Homes Settings screen, 1: small font 2: large font. Font is optional and if not specified is the current font selected on the Homes Settings screen.

c1 can be any color specified as #RRGGBB. The default is black (#000000).

width is optional and if not specified, no clipping is performed.

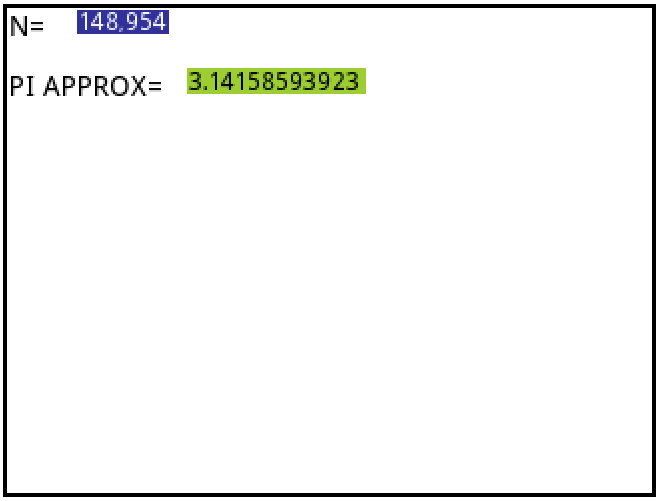
c2 can be any color specified as #RRGGBB. c2 is optional. If not specified the background is not erased.

Example:

The following program displays the successive approximations for π using the series for the arctangent(1). Note that a color for the text, and for background, has been specified (with the width of the text being limited to 100 pixels).

```
EXPORT PISERIES()

BEGIN

LOCAL sign;

K:=2;

A:=4;

sign:=-1;

RECT();

TEXTOUT_P("N=",0,0);

TEXTOUT_P("PI APPROX=",0,30);

REPEAT

A+sign*4/(2*K-1)►A;

TEXTOUT_P(K ,35,0,2,#FFFFFF,100,#333399);

TEXTOUT_P(A ,90,30,2,#000000,100,#99CC33);

sign*-1►sign;

K+1►K;

UNTIL 0;
```

```
END;
```



```
N=   148,954

PI APPROX=   3.14158593923
```

The program executes until the user presses On Off to terminate.

TRIANGLE_P, TRIANGLE

Syntax: `TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3])`

Syntax: `TRIANGLE_P([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"])`

Syntax: `TRIANGLE_P([G],points_definition, triangle_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{3Dxmin, 3Dxmax, 3Dymin, 3Dymax, 3Dzmin, 3Dzmax}]}, [zstring])`

Syntax: `TRIANGLE_P([G],pre_rotated_points, triangle_definitions, [zstring])`

Syntax: `TRIANGLE_P([G])`

The basic form of `TRIANGLE` draws one triangle between the specified pixel coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If three colors are specified, it blends the colors in between the vertices.

The advanced form of `TRIANGLE_P` allows multiple triangles to be rendered at the same time with a potential 3D transformation of the triangles' vertices.

This is mostly used if you have a set of vertices and triangles and want to display them all at once (faster).

`points_definition` is either a list or a matrix of point definition. Each point is defined by two to four numbers: x, y, z, and color. A valid point definition can have multiple forms. Here are a couple of example: [x, y, z, c], {x, y, z, c}, {x, y, #c}, {(x, y), c}, (x,y)... You can use a vector of points instead of a list; in this case, the points can be expressed as complex numbers.

`triangle_ definitions` is either a list or a matrix of triangle definitions. Each triangle is defined by three to five numbers: p1, p2, p3, color and alpha. p1, p2 and p3 are the index in the `points_definition` of the 3 points that define the triangle. Color is used to override the per point color definition. If you need to provide an Alpha, but not a color, use -1 for the color.

Note that `{Color, [Alpha], triangle_1, ..., triangle_n}` is also a valid form to avoid respecifying the same color for each triangle.

rotation_matrix is a matrix between sizes 2*2 to 3*4 that specifies the rotation and translation of the point using usual 3D and 4D geometry.

{eye_x, eye_y, eye_z} defines the eye position (projection).

{3Dxmin, 3Dxmax, 3Dymin, 3Dymax, 3Dzmin, 3Dzmax} is used to perform 3D clipping on the pretransformed objects.

Each point is rotated and translated through a multiplication by the rotation_matrix. It is then projected on the view plane using the eye position calculated by the following equation: x=eye_z/z*x-eye_x and y=eye_z/ z*y-eye_y.

Each triangle is clipped in 3D, if 3D clipping data is provided.

If "N" is specified, the Z coordinates are normalized between 0 and 255 after rotation, providing easier zClipping.

If zstring is provided, per pixel z clipping happens using the z value string (see the following).

TRIANGLE_P returns a string which contains all the transformed points. If you plan to call TRIANGLE or LINE multiple times in a row using the same points and transformation, you can do so by replacing the points_definition with this string and omitting the transformation definition in subsequent calls to TRIANGLE and LINE.

About zstring:

TRIANGLE_P([G]) returns a string adapted for z clipping.

To use z clipping, call TRIANGLE_P([G]) to create a z clipping string (initialized at 255 for each pixels). You can then call TRIANGLE_P with appropriate z (0-255) values for each of the triangle vertices and TRIANGLE_P([G]) will not draw pixels farther than the already drawn pixels. ZString is automatically updated as appropriate.

Matrix

Some matrix commands take as their argument the matrix variable name on which the command is applied. Valid names are the global variables M0–M9 or a local variable that contains a matrix. You can also enter a matrix directly as an argument to the command.

ADDCOL

Syntax: ADDCOL(matrixname, vector, column_number)

Inserts the values in vector into a new column inserted before column_number in the specified matrix. The number of values in the vector must be the same as the number of rows in the matrix.

ADDROW

Syntax: ADDROW(matrixname, vector, row_number)

Inserts the values in vector into a new row inserted before row_number in the specified matrix. The number of values in the vector must be the same as the number of columns in the matrix.

DELCOL

Syntax: DELCOL(name ,column_number)

Deletes *column column_number* from matrix name.

DELROW

Syntax: `DELROW(name ,row_number)`

Deletes *row row_number* from matrix name.

EDITMAT

Syntax: `EDITMAT(matrix variable, [title], [read only])` or `EDITMAT(matrix, [title], [read only])`

Allows you to view or edit the specified matrix.

If a matrix variable (M0–M9) is used, the variable is updated when you tap OK.

The optional title can be either "title" or {"title", ["row names"], ["column names"]}. If entered, "title" is displayed at the top of the matrix editor. If "row names" and "column names" are entered, they are used as the row and column headers in the editor.

If read only is not 0, the user can only view the matrix. That is, the user cannot edit the matrix.

EDITMAT returns the matrix as soon as the command is completed. If used in a program, it returns to the program when you tap OK.

REDIM

Syntax: `REDIM(name, size)`

Redimensions the specified matrix (name) or vector to size. For a matrix, size is a list of two integers (n1,n2). For a vector, size is a list containing one integer (n). Existing values in the matrix are preserved. Fill values will be 0.

REPLACE

Syntax: `REPLACE(name, start, object)`

Replaces portion of a matrix or vector stored in name with an object starting at position start. Start for a matrix is a list containing two numbers; for a vector, it is a single number. `REPLACE` also works with lists, graphics, and strings. For example, REPLACE("123456", 2, "GRM") -> "1GRM56"

SCALE

Syntax: `SCALE(name, value, rownumber)`

Multiplies the specified `row_number` of the specified matrix by `value`.

SCALEADD

Syntax: `SCALEADD(name, value, row1, row2)`

Multiplies the specified `row1` of the matrix (`name`) by `value`, then adds this result to the second specified `row2` of the matrix (`name`) and replaces `row1` with the result.

SUB

Syntax: `SUB(name, start, end)`

Extracts a sub-object—a portion of a list, matrix, or graphic—and stores it in name. Start and end are each specified using a list with two numbers for a matrix, a number for vector or lists, or an ordered pair, (X,Y), for graphics: SUB(M1{1,2},{2,2})

SWAPCOL

Syntax: `SWAPCOL(name, column1, column2)`

Swaps column1 and column2 of the specified matrix (name).

SWAPROW

Syntax: `SWAPROW(name, row1, row2)`

Swaps row1 and row2 in the specified matrix (name).

App Functions

These commands allow you to launch any HP app, bring up any view of the current app, and change the options in the View menu.

STARTAPP

Syntax: `STARTAPP("name")`

Starts the app with name. This will cause the app program's `START` function to be run, if it is present. The app's default view will be started. Note that the `START` function is always executed when the user taps Start in the Application Library. This also works for user-defined apps.

Example: `STARTAPP("Function")` launches the Function app.

STARTVIEW

Syntax: `STARTVIEW( [,draw?])`

Starts the nth view of the current app. If *draw?* is true (that is, not 0), it will force an immediate redrawing of the screen for that view.

The view numbers (n) are as follows:

`Symbolic:0`

`Plot:1`

`Numeric:2`

`Symbolic Setup:3`

`Plot Setup:4`

`Numeric Setup:5`

`App Info: 6`

`View Menu:7`

`First special view (Split Screen Plot Detail):8`

`Second special view (Split Screen Plot Table):9`

`Third special view (Autoscale):10`

`Fourth special view (Decimal):11`

`Fifth special view (Integer):12`

`Sixth special view (Trig):13`

The special views in parentheses refer to the Function app, and may differ in other apps. The number of a special view corresponds to its position in the View menu for that app. The first special view is launched by `STARTVIEW(8)`, the second with `STARTVIEW(9)`, and so on.

You can also launch views that are not specific to an app by specifying a value for n that is less than 0:

```
Home Screen:-1

Home Settings:-2

Memory Manager:-3

Applications Library:-4

Matrix Catalog:-5

List Catalog:-6

Program Catalog:-7

Notes Catalog:-8
```

VIEW

Syntax: `VIEW ("string"[,program_name])`

```
BEGIN

Commands;

END;
```

Adds a custom option to the **View** menu. When **string** is selected, runs `program_name`. See *The DiceSimulation program* in the Example on page 577 section.

Integer

BITAND

Syntax: `BITAND(int1, int2, … intn)`

Returns the bitwise logical AND of the specified integers.

Example: `BITAND(20,13)` returns 4.

BITNOT

Syntax: `BITNOT(int)`

Returns the bitwise logical NOT of the specified integer.

Example: `BITNOT(47)` returns 549755813840.

BITOR

Syntax: `BITOR(int1, int2, … intn)`

Returns the bitwise logical OR of the specified integers.

Example: `BITOR(9,26)` returns 27.

BITSL

Syntax: `BITSL(int1 [,int2])`

Bitwise Shift Left. Takes one or two integers as input and returns the result of shifting the bits in the first integer to the left by the number places indicated by the second integer. If there is no second integer, the bits are shifted to the left by one place.

Examples:

`BITSL(28,2)` returns 112

`BITSL(5)` returns 10.

BITSR

Syntax: `BITRL(int1 [,int2])`

Bitwise Shift Right. Takes one or two integers as input and returns the result of shifting the bits in the first integer to the right by the number places indicated by the second integer. If there is no second integer, the bits are shifted to the right by one place.

Examples:

`BITSR(112,2)` returns 28

`BITSR(10)` returns 5.

BITXOR

Syntax: `BITXOR(int1, int2, … intn)`

Returns the bitwise logical exclusive OR of the specified integers.

Example: `BITXOR(9,26)` returns 19.

B→R

Syntax: `B→R(#integerm)`

Converts an integer in base m to a decimal integer (base 10). The base marker m can be b (for binary), o (for octal), or h (for hexadecimal).

Example: `B→R(#1101b)` returns 13

GETBASE

Syntax: `GETBASE(#integer[m])`

Returns the base for the specified integer (in whatever is the current default base): 0 = default, 1 = binary, 2 = octal, 3 = hexadecimal.

Examples: `GETBASE(#1101b)` returns #1h (if the default base is hexadecimal) while `GETBASE (#1101)` returns #0h.

GETBITS

Syntax: `GETBITS(#integer)`

Returns the number of bits used to encode an integer.

If the integer is not specified, the current value of the Integers box in Page 1 of Home Settings is used.

Examples:

`GETBITS(#22122)` returns 32.

`GETBITS(#1:45h)` returns 45.

R→B

Syntax: `R→B(integer)`

Converts a decimal integer (base 10) to an integer in the default base.

Example: `R→B(13)` returns #1101b (if the default base is binary) or #Dh (if the default base is hexadecimal).

SETBITS

Syntax: `SETBITS(#integer[m] [,bits])`

Sets the number of bits to represent integer. Valid values are in the range –63 to 64. If m or bits is omitted, the default value is used.

Example: `SETBITS(#1111b, 15)` returns #1111:15b

SETBASE

Syntax: `SETBASE(#integer[m][c])`

Displays integer expressed in base m in whatever base is indicated by c, where c can be 1 (for binary), 2 (for octal), or 3 (for hexadecimal). Parameter m can be b (for binary), d (for decimal), o (for octal), or h (for hexadecimal). If m is omitted, the input is assumed to be in the default base. Likewise, if c is omitted, the output is displayed in the default base.

Examples: `SETBASE (#34o,1)` returns #11100b while `SETBASE (#1101)` returns #0h ((if the default base is hexadecimal).

I/O

I/O commands are used for inputting data into a program, and for outputting data from a program. They allow users to interact with programs.

CHOOSE

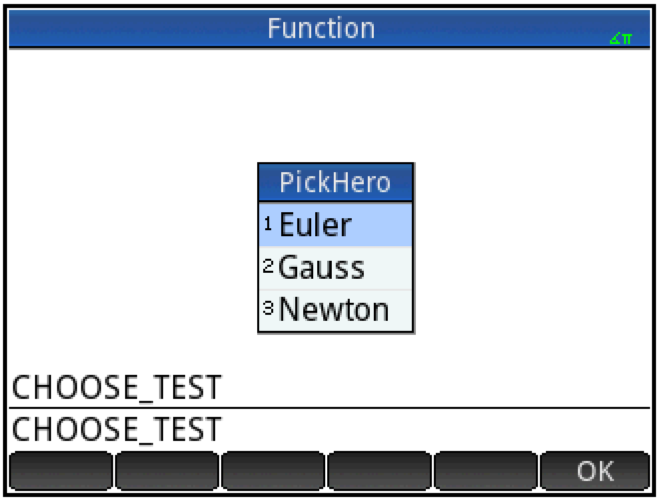
Syntax: `CHOOSE(var, "title", "item1", "item2",…,"itemn")`

Displays a choose box with the title and containing the choose items. If the user selects an object, the variable whose name is provided will be updated to contain the number of the selected object (an integer, 1, 2, 3, …) or 0 if the user taps Cancel.

Returns true (not zero) if the user selects an object, otherwise return false (0).

Example:

```
CHOOSE

(N,"PickHero","Euler","Gauss","Newton");

IF N==1 THEN PRINT("You picked Euler"); ELSE IF N==2 THEN PRINT("You
picked Gauss");ELSE PRINT("You picked Newton");

END;

END;
```



```
CHOOSE_TEST
CHOOSE_TEST
```

After execution of `CHOOSE`, the value of N will be updated to contain 0, 1, 2, or 3. The `IF THEN ELSE` command causes the name of the selected person to be printed to the terminal.

EDITLIST

Syntax: `EDITLIST(listvar)`

Starts the List Editor loading listvar and displays the specified list. If used in programming, returns to the program when user taps OK.

Example: `EDITLIST(L1)` edits list L1.

EDITMAT

Syntax: `EDITMAT(matrixvar)`

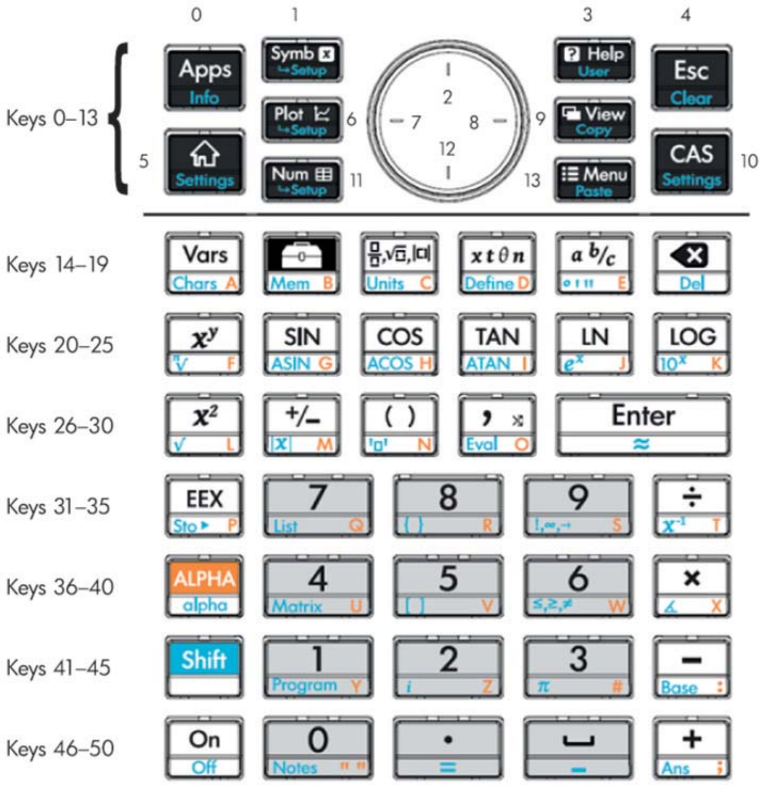
Starts the Matrix Editor and displays the specified matrix. If used in programming, returns to the program when user taps OK.

Example: `EDITMAT(M1)` edits matrix M1.

GETKEY

Syntax: `GETKEY`

Returns the ID of the first key in the keyboard buffer, or –1 if no key was pressed since the last call to GETKEY. Key IDs are integers from 0 to 50, numbered from top left (key 0) to bottom right (key 50) as shown in figure 27-1.



INPUT

Syntax: `INPUT(var,["title"], ["label"], ["help"], [reset_value], [initial_value])`

Syntax: `INPUT({vars},["title"], [{"labels"}], [{"help"}], [{reset_values}], [{initial_values}])`

The simpler form of this command opens a dialog box with the given title and one field named label, displaying help at the bottom. The dialog box includes the CANCEL and OK menu keys. The user can enter a value in the labeled field. If the user presses the OK menu key, the variable var is updated with the entered value and 1 is returned. If the user presses the CANCEL menu key, the variable is not updated and 0 is returned.

In the more complex form of the command, lists are used to create a multi-field dialog box. If var is a list, each element can be either a variable name or a list using the following syntax.

- {var_name, real, [{pos}]} to create a check box control. If real is >1, this check box gets pooled with the next n -1 check boxes in a radio group (that is, only one of the n check boxes can be checked at any time)
- {var_name, [allowed_types_matrix] , [{pos}]} to create an edit field. [allowed_types_matrix] lists all the allowed types ([-1] stands for all types allowed). If the only allowed type is a string, the edition hides the double quotes.
- {var_name, {Choose items}, [{pos}]} to create a choose field.

If pos is specified, it is a list of the form {field start in screen %, field width in screen%, line(starts at 0)}. This allows you to control the precise position and size of your fields. Note that you have to specify pos for either none or all fields in the dialog box.

There is a maximum of seven lines of controls per page. Controls with more than seven lines are placed in subsequent pages. If more than one page is created, ["title"] can be a list of titles.

ISKEYDOWN

Syntax: `ISKEYDOWN(key_id);`

Returns true (non-zero) if the key whose key_id is provided is currently pressed, and false (0) if it is not.

MOUSE

Syntax: `MOUSE[(index)]`

Returns two lists describing the current location of each potential pointer (or empty lists if the pointers are not used). The output is {x , y, original x, original y, type} where type is 0 (for new), 1 (for completed), 2 (for drag), 3 (for stretch), 4 (for rotate), and 5 (for long click).

The optional parameter index is the nth element that would have been returned—x, y, original x, etc.—had the parameter been omitted (or –1 if no pointer activity had occurred).

MSGBOX

Syntax: `MSGBOX(expression or string [ ,ok_cancel?]);`

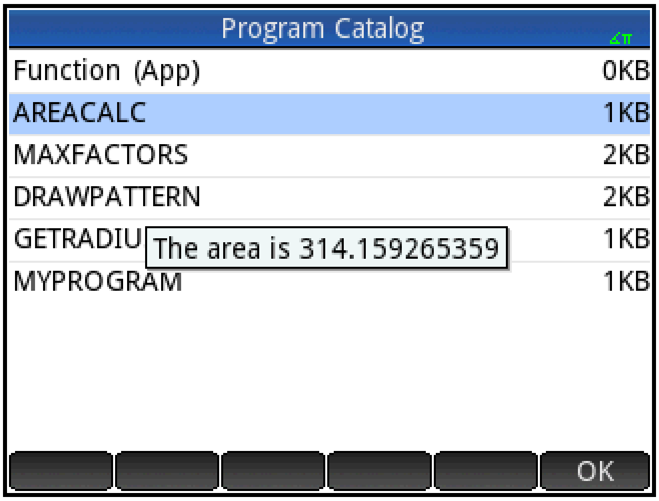
Displays a message box with the value of the given expression or string.

If `ok_cancel?` is true, displays the OK and Cancel buttons, otherwise only displays the OK button. Default value for `ok_cancel` is false.

Returns true (non-zero) if the user taps OK, false (0) if the user presses Cancel.

```
EXPORT AREACALC()
BEGIN
LOCAL radius;
INPUT(radius, "Radius of Circle","r = ","Enter radius",1);
MSGBOX("The area is " +π*radius^2);
END;
```

If the user enters 10 for the radius, the message box shows this:



Program Catalog

Function (App)	0KB
AREACALC	1KB
MAXFACTORS	2KB
DRAWPATTERN	2KB
GETRADIU	1KB
MYPROGRAM	1KB

The area is 314.159265359

PRINT

Syntax: `PRINT(expression or string);`

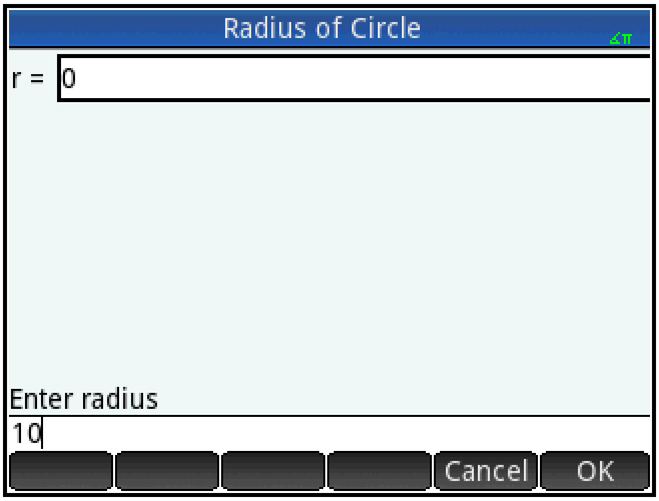
Prints the result of expression or string to the terminal.

The terminal is a program text output viewing mechanism which is displayed only when `PRINT` commands are executed. When visible, you can press ▼ or ▲ to view the text, ⌫ Del to erase the text and any other key to hide the terminal. Pressing On Off stops the interaction with the terminal. PRINT with no argument clears the terminal.

There are also commands for outputting data in the Graphics section. In particular, the commands `TEXTOUT` and `TEXTOUT_P` can be used for text output.

This example prompts the user to enter a value for the radius of a circle, and prints the area of the circle on the terminal.

```
EXPORT AREACALC()

BEGIN

LOCAL radius;

INPUT(radius, "Radius of Circle","r = ","Enter radius",1);

PRINT("The area is " +π*radius^2);

END;
```

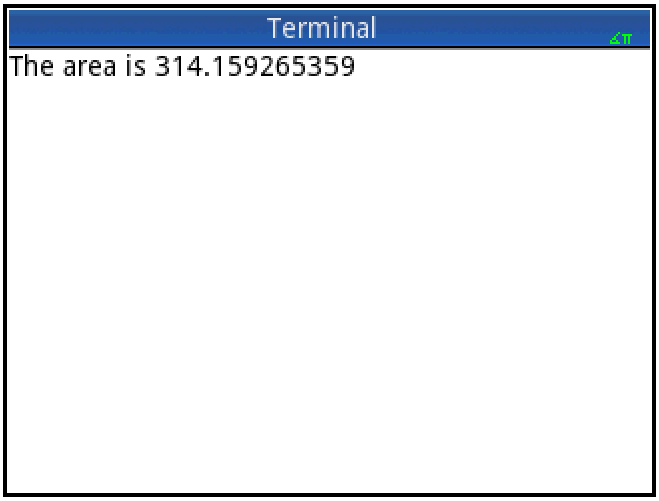


Radius of Circle

r = 0

Enter radius

10



```
The area is 314.159265359
```

Notice the use of the `LOCAL` variable for the radius, and the naming convention that uses lower case letters for the local variable. Adhering to such a convention will improve the readability of your programs.

WAIT

Syntax: `WAIT(n);`

Pauses program execution for n seconds. With no argument or with n = 0, pauses program execution for one minute.

More

%CHANGE

Syntax: `%CHANGE(x,y)`

The percentage change in going from x to y.

Example: `%CHANGE(20,50)` returns 150.

%TOTAL

Syntax: `%TOTAL(x,y)`

The percentage of x that is y.

Example: `%TOTAL(20,50)` returns 250.

CAS

Syntax: `CAS.function()` or `CAS.variable`

Executes the function or returns the variable using the CAS.

EVALLIST

Syntax: `EVALLIST({list})`

Evaluates the content of each element in a list and returns an evaluated list.

EXECON

Syntax: `EXECON (&expr, List1, [List2,…])`

Creates a new list based on the elements in one or more lists by iteratively modifying each element according to an expression that contains the ampersand character (&).

Examples:

`EXECON("&1+1",{1,2,3})` returns {2,3,4}

Where the & is followed directly by a number, the position in the list is indicated. For example:

`EXECON("&2-&1",{1, 4, 3, 5}"` returns {3, –1, 2}

In the example above, &2 indicates the second element and &1 the first element in each pair of elements. The minus operator between them subtracts the first from the second in each pair until there are no more pairs. In this case (with just a single list), the numbers appended to & can only be from 1 to 9 inclusive.

EXECON can also operate on more than one list. For example:

`EXECON("&1+&2",{1,2,3},{4,5,6})` returns {5,7,9}

In the example above, &1 indicates an element in the first list and &2 indicates the corresponding element in the second list. The plus operator between them adds the two elements until there are no more pairs. With two lists, the numbers appended to & can have two digits; in this case, the first digit refers to the list number (in order from left to right) and the second digit can still only be from 1 to 9 inclusive.

EXECON can also begin operating on a specified element in a specified list. For example:

`EXECON("&23+&1",{1,5,16},{4,5,6,7})` returns {7,12}

In the example above, &23 indicates that operations are to begin on the second list and with the third element. To that element is added the first element in the first list. The process continues until there are no more pairs.

→HMS

Syntax: `→HMS(value)`

Converts a decimal value to hexagesimal format; that is, in units subdivided into groups of 60. This includes degrees, minutes, and seconds as well as hours, minutes, and seconds.

Example: `→HMS(54.8763)` returns 54°52′34.68″

HMS→

Syntax: `HMS→(value)`

Converts a value expressed in hexagesimal format to decimal format.

Example: `HMS→(54°52′34.68″)` returns 54.8763

ITERATE

Syntax: `ITERATE(expr, var, ivalue, #times)`

For `#times`, recursively evaluates `expr` in terms of `var` beginning with `var = ivalue`.

Example: `ITERATE(X^2, X, 2, 3)` returns 256

TICKS

Syntax: `TICKS`

Returns the internal clock value in milliseconds.

TEVAL

Syntax: `TEVAL(parameter)`

Returns the time in seconds that it takes to evaluate the parameter.

TYPE

Syntax: `TYPE(object)`

Returns the type of the object:

0: Real

1: Integer

2: String

3: Complex

4: Matrix

5: Error

6: List

8: Function

9: Unit

14.?: cas object. The fractional part is the cas type.

Variables and programs

The HP Prime has four types of variables: Home variables, App variables, CAS variables, and User variables. You can retrieve these variables from the Variable menu (Vars).

The names of Home variables are reserved; that is, they cannot be deleted from the system and cannot be used to store objects of any other type than that for which they were designed. For example, A–Z and θ are reserved to store real numbers, Z0–Z9 are reserved to store complex numbers, and L0–L9 are reserved to store lists, etc. As a result, you cannot store a matrix in L8 or a list in Z.

Home variables keep the same value in Home and in apps; that is, they are global variables common to the system. They can be used in programs with that understanding.

App variable names are also reserved, though a number of apps may share the same app variable name. In any of these cases, the name of the app variable must be qualified if that variable is not from the current app. For example, if the current app is the Function app, `Xmin` will return the minimum x-value in the Plot view of the Function app. If you want the minimum value in the Plot view of the Polar app, then you must enter `Polar.Xmin`. App variables represent the definitions and settings you make when working with apps interactively. As you work through an app, the app functions may store results in app variables as well. In a program, app variables are used to edit an app's data to customize it and to retrieve results from the app's operation.

CAS variables are similar to the Home real variables A–Z, except that they are lowercase and designed to be used in CAS view and not Home view. Another difference is that Home and App variables always contain values, while CAS variables can be simply symbolic and not contain any particular value. The CAS variables are not typed like the Home and App variables. For example, the CAS variable t may contain a real number, a list, or a vector, etc. If a CAS variable has a value stored in it, calling it from Home view will return its contents.

User variables are variables created by the user, either directly or exported from a user program. They provide one of several mechanisms to allow programs to communicate with the rest of the calculator and with other programs. User variables created in a program may be either local to that program or global. Once a variable has been exported from a program, it will appear among the user variables in the **Variables** menu, next to the program that exported it. User variables may be multicharacter, but must follow certain rules; see Variables and visibility on page 567 for details.

User variables, like CAS variables, are not typed and thus may contain objects of different types.

The following sections deal with using app variables in programs, providing descriptions of each app variable by name and its possible contents. For a list of all the Home and app variables, see the “Variables“ chapter. For user variables in programs, see The HP Prime programming language on page 567.

App variables

Not all app variables are used in every app. S1Fit, for example, is only used in the Statistics 2Var app. However, many of the variables are common to the Function, Advanced Graphing, Parametric, Polar, Sequence, Solve, Statistics 1Var, and Statistics 2Var apps. If a variable is not available in all of these apps, or is available only in some of these apps (or some other app), then a list of the apps where the variable can be used appears under the variable name.

The following sections list the app variables by the view in which they are used. To see the variables listed by the categories in which they appear on the Variables menu see the “App variables” section of the “Variables” chapter.

Current app variables

These variables give the user access to data and files associated with the currently active app.

AFiles

Each HP Prime app can have any number of files associated with it. These files are sent with the app. For example, if there is a file named icon.png associated with the app, then this file is used for the app icon in the Application Library.

`AFiles` returns the list of all these files.

`AFiles("name")` returns the contents of the file with the given name.

`AFiles("name"):= object` stores the object in the file with the given name.

AFilesB

Each HP Prime app can have any number of files associated with it. These files are sent with the app. AFilesB is the binary equivalent of the AFiles variable.

`AFilesB` returns the list of all files associated with an app.

`AFilesB("name")` returns the size of the file with the given name.

`AFilesB("name", position, [nb])` returns nb bytes read from the file with the given name, starting from position in the file (position starts at 0).

`AFilesB("name", position):= value` or `{values...}` stores n bytes, starting at position, in the file with the given name.

ANote

ANote returns the note associated with an HP app. This is the note displayed when the user presses Shift Apps Info.

`ANote:="string"` sets the note associated with the app to contain the string.

AProgram

AProgram returns the program associated with an HP Prime app.

`AProgram:="string"` sets the program associated with the app to contain the string.

AVars

AVars returns the list of the names of all the variables associated with an HP Prime app.

`AVars(n)` returns the content of the nth variable associated with the app.

`AVars("name")` returns the content of the specified variable associated with the app.

`AVars(n or "name"):= value` sets the specified app variable to contain the given value. If "name" is not an existing variable, a new one is created.

After an app variable is created through AVars("name"):= value, you can use the variable by typing the variable name.

DelAVars

`DelAVars(n, or "name")` deletes the specified app variable.

DelAFiles

`DelAFiles("name")` deletes the specified file associated with an HP app.

Plot view variables

Axes

Turns axes on or off.

In Plot Setup view, check (or uncheck) `AXES`.

In a program, type:

`0 ▶ Axes`—to turn axes on.

`1 ▶ Axes`—to turn axes off.

Cursor

Sets the type of cursor. (Inverted or blinking is useful if the background is solid).

In Plot Setup view, choose **Cursor**.

In a program, type:

`0 ▶ Cursor`—for solid crosshairs (default).

`1 ▶ Cursor`—to invert the crosshairs.

`2 ▶ Cursor`—for blinking crosshairs.

GridDots

Turns the background dot grid in Plot view on or off. In Plot Setup view, check (or uncheck) `GRID DOTS`. In a program, type:

`0 ▶ GridDots`—to turn the grid dots on (default).

`1 ▶ GridDots`—to turn the grid dots off.

GridLines

Turns the background line grid in Plot View on or off.

In Plot Setup view, check (or uncheck) GRID LINES.

In a program, type:

`0 ▶ GridLines`—to turn the grid lines on (default).

`1 ▶ GridLines`—to turn the grid lines off.

Hmin/Hmax

Statistics 1Var

Defines the minimum and maximum values for histogram bars.

In Plot Setup view for one-variable statistics, set values for `HRNG`.

In a program, type:

n_1 ▶ `Hmin`

n_2 ▶ `Hmax`

where $n_1 < n_2$

Hwidth

Statistics 1Var

Sets the width of histogram bars.

In Plot Setup view for one-variable statistics, set a value for `Hwidth`.

In a program, type:

`n ▶ Hwidth` where n > 0

Labels

Draws labels in Plot View showing X and Y ranges.

In Plot Setup View, check (or uncheck) `Labels`.

In a program, type:

`1 ▶ Labels`—to turn labels on (default).

`2 ▶ Labels`—to turn labels off.

Method

Function, Solve, Parametric, Polar, Statistics 2Var

Defines the graphing method: adaptive, fixed-step segments, or fixed-step dots.

In a program, type:

`0 ▶ Method`—select adaptive.

`1 ▶ Method`—select fixed-step segments.

`2 ▶ Method`—select fixed-step dots.

Nmin/Nmax

Sequence

Defines the minimum and maximum values for the independent variable.

Appears as the **N RNG** fields in the Plot Setup view. In Plot Setup view, enter values for `N Rng`.

In a program, type:

n_1 `▶ Nmin`

n_2 `▶ Nmax`

where $n_1 < n_2$

PixSize

Geometry

Sets the dimensions of each square pixel in the Geometry app. In Plot view, enter a positive value in `Pixel Size`.

Or enter `PixSize:=n`, where n>0.

Recenter

Recenters at the cursor when zooming.

From Plot-Zoom-Set Factors, check (or uncheck) **Recenter**.

In a program, type:

`0 ▶ Recenter`—to turn recenter on (default).

`1 ▶ Recenter`—to turn recenter off.

S1mark-S5mark

Statistics 2Var

Sets the mark to use for scatter plots.

In Plot Setup view for two-variable statistics, select one of `S1 Mark-S Mark`.

ScrollText

Geometry

Determines whether the current command in Plot view scrolls automatically or manually. In Plot view, select or clear Scroll Text.

You can also enter `ScrollText:=0` to scroll manually or `ScrollText:=1` to scroll automatically.

SeqPlot

Sequence

Enables you to choose between a Stairstep or a Cobweb plot.

In Plot Setup view, select `SeqPlot`, then choose `Stairstep` or `Cobweb`.

In a program, type:

`0 ▶ SeqPlot`—for Stairstep.

`1 ▶ SeqPlot`—for Cobweb.

θmin/θmax

Polar

Sets the minimum and maximum independent values.

In Plot Setup view enter values for θ `Rng`.

In a program, type:

n_1 ▶ `θmin`

n_2 ▶ `θmax`

where $n_1 < n_2$

θstep

Polar

Sets the step size for the independent variable.

In Plot Setup view, enter a value for θ Step.

In a program, type:

`n ▶ θstep`

where n > 0

Tmin/Tmax

Parametric

Sets the minimum and maximum independent variable values.

In Plot Setup view, enter values for `T Rng`.

In a program, type:

n_1 ▶ `Tmin`

n_2 ▶ `Tmax`

where $n_1 < n_2$

Tstep

Parametric

Sets the step size for the independent variable.

In Plot Setup view, enter a value for `T Step`.

In a program, type:

n ▶ Tstep

where n > 0

Xtick

Sets the distance between tick marks for the horizontal axis.

In Plot Setup view, enter a value for `X Tick`.

In a program, type:

```
n ▶ Xtick
```

where n > 0

Ytick

Sets the distance between tick marks on the vertical axis.

In Plot Setup view, enter a value for `Y Tick`.

In a program, type:

```
n ▶ Ytick
```

where n > 0

Xmin/Xmax

Sets the minimum and maximum horizontal values of the plot screen.

In Plot Setup view, enter values for X Rng.

In a program, type:

n_1 ▶ Xmin

n_2 ▶ Xmax

where $n_1 < n_2$

Ymin/Ymax

Sets the minimum and maximum vertical values of the plot screen.

In Plot Setup view, enter the values for `Y Rng`.

In a program, type:

n_1 ▶ Ymin

n_2 ▶ Ymax

where $n_1 < n_2$

Xzoom

Sets the horizontal zoom factor.

In Plot View, press Menu Paste then Zoom. Scroll to **Set Factors**, select it and tap OK. Enter the value for `X Zoom` and tap OK.

In a program, type:

```
n ▶ Xzoom
```

where n > 0

The default value is 4.

Yzoom

In Plot View, press Menu Paste then Zoom. Scroll to **Set Factors**, select it and tap OK. Enter the value for `Y Zoom` and tap OK.

In a program, type:

```
n ▶ Yzoom
```

where n > 0

The default value is 4.

Symbolic view variables

AltHyp

Inference

Determines the alternative hypothesis used for hypothesis testing.

In Symbolic View, select an option for `Alt Hypoth`.

In a program, type:

```
0 ▶ AltHyp—μ < μ0
1 ▶ AltHyp—μ > μ0
2 ▶ AltHyp—μ ≠ μ0
```

E0...E9

Solve

Contains an equation or expression. In Symbolic view, select one of `E0` through `E9` and enter an expression or equation. The independent variable is selected by highlighting it in Numeric view.

In a program, type (for example):

```
X+Y*X-2=Y ▶ E1
```

F0...F9

Function

Contains an expression in X. In Symbolic View, select one of F0 through F9 and enter an expression.

In a program, type (for example):

```
SIN(X) ▶ F1
```

H1...H5

Statistics 1Var

The Statistics 1Var symbolic variables are H1 to H5. These variables contain the data values for a one-variable statistical analysis. For example, H1(n) returns the nth value in the data set for the H1 analysis. With no argument, H1 returns a list of the objects that define H1. These objects are as follows, in the order given:

- An expression (in single quotes) that defines the data list (or empty double quotes)
- An expression (in single quotes) that optionally defines the frequencies for each of the values in the data list (or empty double quotes)
- The plot type number
- The option number
- The color for the plot

The plot type number is an integer from 1 to 9 that controls which statistical plot type is used with each of the variables H1 to H5. The correspondence is as follows:

- **1**—Histogram (default)
- **2**—Box and whisker
- **3**—Normal probability
- **4**—Line
- **5**—Bar
- **6**—Pareto
- **7**—Control
- **8**—Dot
- **9**—Stem and leaf

The option number is an integer from 0 to 2 that controls any option available for the plot type. The correspondence is as follows:

- **0**—No option
- **1**—Do not show outliers for the box-and-whisker plot
- **2**—Show outliers for the box-and-whisker plot

Example:

`H3:={"D1", "", 2, 1, #FF:24h}` defines H3 to use D1 for its data list, use no frequencies, and draw a box-and-whisker plot without outliers in blue.

Method

Inference

Determines whether the Inference app is set to calculate hypothesis test results or confidence intervals. In Symbolic view, make a selection for Method.

In a program, type:

`0 ▶ Method`—for Hypothesis Test

`2 ▶ Method`—for Confidence Interval

`3 ▶ Method`—for Chi-Square

`4 ▶ Method`—for Regression

R0...R9

Polar

Contains an expression in θ. In Symbolic view, select one of `R0` through `R9` and enter an expression.

In a program, type:

```
SIN(θ) ▶ R1
```

S1...S5

Statistics 2Var

The Statistics 2Var app variables are S1 to S5. These variables contain the data that define a two-variable statistical analysis. S1 returns a list of the objects that define S1. Each list contains the following items, in order:

- An expression (in single quotes) that defines the independent variable data list (or empty double quotes)
- An expression (in single quotes) that defines the dependent variable data list (or empty double quotes)
- A string or expression that optionally defines the frequencies for the dependent data list
- The fit type number
- The fit expression
- The scatter plot color
- The mark type number for the scatter-plot point
- The fit plot color

The fit type number is an integer from 1 to 13 that controls which statistical plot type is used with each of the variables S1 to S5. The correspondence is as follows:

- **1**—Linear
- **2**—Logarithmic
- **3**—Exponential
- **4**—Power
- **5**—Exponent
- **6**—Inverse
- **7**—Logistic
- **8**—Quadratic
- **9**—Cubic
- **10**—Quartic
- **11**—Trigonometric
- **12**—Median-median line
- **13**—User defined

The mark type number for the scatter-plot point is an integer from 1 to 9 that controls which graphic is used to represent each point in a scatter plot. The correspondence is as follows:

- **1**—Small hollow dot
- **2**—Small hollow square
- **3**—Thin x
- **4**—Hollow cross
- **5**—Small hollow diamond
- **6**—Thick x
- **7**—Small solid dot
- **8**—Thin diamond
- **9**—Large hollow dot

Example:

`S1:={"C1", "C2", "", 1, "", #FF:24h, 1, #FF:24h}` sets C1 as the independent data, C2 as the dependent data, no frequencies for the dependent data, a linear fit, no specific equation for that linear fit, a blue scatter plot with mark type 1, and a blue fit plot.

InfType

Inference

Determines the type of hypothesis test or confidence interval. Depends upon the value of the variable `Method`. From Symbolic View, make a selection for `Type`.

Or, in a program, store the constant number from the list below into the variable Type. With `Method=0`, the constant values and their meanings are as follows:

0 Z-Test: 1 μ

1 Z-Test: μ_1 - μ_2

2 Z-Test:1 π

3 Z-Test: π_1 - π_2

4 T-Test: 1 μ

5 T-Test: μ_1 - μ_2

With `Method=1`, the constants and their meanings are as follows:

0 Z-Int: 1 μ

1 Z-Int: μ_1 - μ_2

2 Z-Int:1 π

3 Z-Int: π_1 - π_2

4 T-Int: 1 μ

5 T-Int: μ_1 - μ_2

With `Method=2`, the constants and their meanings are as follows:

0 Chi-square goodness of fit test

1 Chi-square two-way test

With `Method=3`, the constants and their meanings are as follows:

0 Linear t-test

1 Interval: Slope

2 Interval: Intercept

3 Interval: Mean Response

4 Prediction Interval

X0, Y0...X9,Y9

Parametric

Contains two expressions in `T: X(T)` and `Y(T)`. In Symbolic view, select any of `X0-Y0` through `X9-Y9` and enter expressions in T.

In a program, store expressions in T in Xn and Yn, where n is an integer from 0 to 9.

Example:

```
SIN(4*T)▶ Y1;2*SIN(6*T)▶ X1
```

U0...U9

Sequence

Contains an expression in N. In Symbolic view, select any of `U0` through `U9` and enter an expression in `N`, `Un(N-1)`, or `Un(N-2)`.

In a program, use the `RECURSE` command to store the expression in U*n*, where *n* is an integer from 0 to 9.

Example:

```
RECURSE (U,U(N-1)*N,1,2) ▶ U1
```

Numeric view variables

C0...C9

Statistics 2Var

Contain lists of numerical data. In Numeric view, enter numerical data in `C0` through `C9`.

In a program, type:

```
 LIST ▶ Cn
```

where n = 0, 1, 2, 3 ... 9 and `LIST` is either a list or the name of a list.

D0...D9

Statistics 1Var

Contain lists of numerical data. In Numeric view, enter numerical data in `D0` through `D9`.

In a program, type:

```
 LIST ▶ Dn
```

where n = 0, 1, 2, 3 ... 9 and `LIST` is either a list or the name of a list.

NumIndep

Function Parametric Polar Sequence Advanced Graphing

Specifies the list of independent values (or two-value sets of independent values) to be used by Build Your Own Table. Enter your values one-by-one in the Numeric view.

In a program, type:

```
LIST ▶ NumIndep
```

`List` can be either a list itself or the name of a list. In the case of the Advanced Graphing app, the list will be a list of pairs (a list of 2-element vectors) rather than a list of numbers.

NumStart

Function Parametric Polar Sequence

Sets the starting value for a table in Numeric view.

From Numeric Setup view, enter a value for `NUMSTART`.

In a program, type:

```
n ▶ NumStart
```

NumXStart

Advanced Graphing

Sets the starting number for the X-values in a table in Numeric view.

From Numeric Setup view, enter a value for `NUMXSTART`.

In a program, type:

```
n ▶ NumXStart
```

NumYStart

Advanced Graphing

Sets the starting value for the Y-values in a table in Numeric view.

From Numeric Setup view, enter a value for `NUMYSTART`.

In a program, type:

```
n ▶ NumYStart
```

NumStep

Function Parametric Polar Sequence

Sets the step size (increment value) for the independent variable in Numeric view.

From Numeric Setup view, enter a value for `NUMSTEP`.

In a program, type:

```
n ▶ NumStep
```

where n > 0

NumXStep

Advanced Graphing

Sets the step size (increment value) for the independent X variable in Numeric view.

From Numeric Setup view, enter a value for `NUMXSTEP`.

In a program, type:

```
n ▶ NumXStep
```

where n > 0

NumYStep

Advanced Graphing

Sets the step size (increment value) for the independent Y variable in Numeric view.

From Numeric Setup view, enter a value for `NUMYSTEP`.

In a program, type:

```
n ▶ NumYStep
```

where n > 0

NumType

Function Parametric Polar Sequence Advanced Graphing

Sets the table format.

In Numeric Setup view, make a selection for `NumType`.

In a program, type:

`0 ▶ NumType`—for Automatic (default).

`1 ▶ NumType`—for BuildYourOwn.

NumZoom

Function Parametric Polar Sequence

Sets the zoom factor in the Numeric view.

From Numeric Setup view, type in a value for `NUMZOOM`.

In a program, type:

```
n ▶ NumZoom
```

where n > 0

NumXZoom

Advanced Graphing

From Numeric Setup view, type in a value for `NUMXZOOM`.

In a program, type:

```
n ▶ NumXZoom
```

where n > 0

NumYZoom

Advanced Graphing

Sets the zoom factor for the values in the Y column in the Numeric view.

From Numeric Setup view, type in a value for `NUMYZOOM`.

In a program, type:

```
n ▶ NumYZoom
```

where n > 0

Inference app variables

The following variables are used by the Inference app. They correspond to fields in the Inference app Numeric view. The set of variables shown in this view depends on the hypothesis test or the confidence interval selected in the Symbolic view.

Alpha

Sets the alpha level for the hypothesis test. From the Numeric view, set the value of `Alpha`.

In a program, type:

```
n ▶ Alpha
```

where 0 < n < 1

Conf

Sets the confidence level for the confidence interval. From Numeric view, set the value of `C`.

In a program, type:

```
n ▶ Conf
```

where 0 < n < 1

ExpList

Contains the expected counts by category for the chi-square goodness of fit test. In the Symbolic view field Expected, select `Count`. Then, in Numeric view, enter the data in `ExpList`.

Mean$_1$

Sets the value of the mean of a sample for a 1-mean hypothesis test or confidence interval. For a 2-mean test or interval, sets the value of the mean of the first sample. From Numeric view, set the value of $\bar{x}$ or $\bar{x}_1$.

In a program, type:

n ▶ Mean_1

Mean$_2$

For a 2-mean test or interval, sets the value of the mean of the second sample. From Numeric view, set the value of $\bar{x}_2$.

In a program, type:

n ▶ Mean_2

μ_0

Sets the assumed value of the population mean for a hypothesis test. From the Numeric view, set the value of μ_0.

In a program, type:

n ▶ μ^0

where $0 < \mu_0 < 1$

n_1

Sets the size of the sample for a hypothesis test or confidence interval. For a test or interval involving the difference of two means or two proportions, sets the size of the first sample. From the Numeric view, set the value of n_1.

In a program, type:

n ▶ n^1

n_2

For a test or interval involving the difference of two means or two proportions, sets the size of the second sample. From the Numeric view, set the value of n_2.

In a program, type:

n ▶ n_2

ObsList

Contains the observed count data for the chi-square goodness of fit test. In Numeric view, enter your data in `ObsList`.

ObsMat

Contains the observed counts by category for the chi-square two-way test. In Numeric view, enter your data in `ObsMat`.

π_0

Sets the assumed proportion of successes for the One-proportion Z-test. From the Numeric view, set the value of π_0.

In a program, type:

n ▶ π_0

where $0 < \pi_0 < 1$

Pooled

Determine whether or not the samples are pooled for tests or intervals using the Student's T-distribution involving two means. From the Numeric view, set the value of `Pooled`.

In a program, type:

```
0 ▶ Pooled
```
—for not pooled (default).

```
1 ▶ Pooled
```
—for pooled.

ProbList

Contains the expected probabilities by category for the chi-square goodness of fit test. In the Symbolic view, in the Expected box, select `Probability`. Then, in Numeric view, enter the data in `ProbList`.

s_1

Sets the sample standard deviation for a hypothesis test or confidence interval. For a test or interval involving the difference of two means or two proportions, sets the sample standard deviation of the first sample. From the Numeric view, set the value of s_1.

In a program, type:

```
n ▶ s1
```

s_2

For a test or interval involving the difference of two means or two proportions, sets the sample standard deviation of the second sample. From the Numeric view, set the value of s_2.

In a program, type:

```
n ▶ s2
```

σ_1

Sets the population standard deviation for a hypothesis test or confidence interval. For a test or interval involving the difference of two means or two proportions, sets the population standard deviation of the first sample. From the Numeric view, set the value of σ_1.

In a program, type:

```
n ▶ σ1
```

σ_2

For a test or interval involving the difference of two means or two proportions, sets the population standard deviation of the second sample. From the Numeric view, set the value of σ_2.

In a program, type:

```
n ▶ σ2
```

x_1

Sets the number of successes for a one-proportion hypothesis test or confidence interval. For a test or interval involving the difference of two proportions, sets the number of successes of the first sample. From the Numeric view, set the value of x_1.

In a program, type:

```
n ▶ x1
```

x_2

For a test or interval involving the difference of two proportions, sets the number of successes of the second sample. From the Numeric view, set the value of x_2.

In a program, type:

```
n ▶ x₂
```

Xlist

Contains the list of explanatory data (X) for the regression tests and intervals. In Numeric view, enter your data in `Xlist`.

Xval

For the confidence interval for the mean response and prediction interval for a future response, contains the value of the explanatory variable (X) under scrutiny. Enter a value when prompted by the wizard.

Ylist

Contains the list of response data (Y) for the regression tests and intervals. In Numeric view, enter your data in `Ylist`.

Finance app variables

The following variables are used by the Finance app. They correspond to the fields in the Finance app Numeric view.

CPYR

Compounding periods per year. Sets the number of compounding periods per year for a cash flow calculation. From the Numeric view of the Finance app, enter a value for `C/YR`.

In a program, type:

```
n ▶ CPYR
```

where n > 0

BEG

Determines whether interest is compounded at the beginning or end of the compounding period. From the Numeric view of the Finance app, select or clear `End`.

In a program, type:

`1 ▶ BEG`—for compounding at the end of the period (default).

`0 ▶ BEG`—for compounding at the beginning of the period.

FV

Future value. Sets the future value of an investment. From the Numeric view of the Finance app, enter a value for `FV`.

In a program, type:

```
n ▶ FV
```

Positive values represent return on an investment or loan.

IPYR

Interest per year. Sets the annual interest rate for a cash flow. From the Numeric view of the Finance app, enter a value for I%YR.

In a program, type:

```
n ▶ IPYR
```

where n > 0

NbPmt

Number of payments. Sets the number of payments for a cash flow. From the Numeric view of the Finance app, enter a value for N.

In a program, type:

```
n ▶ NbPmt
```

where n > 0

PMT

Payment value. Sets the value of each payment in a cash flow. From the Numeric view of the Finance app, enter a value for PMT.

In a program, type:

```
n ▶ PMT
```

Note that payment values are negative if you are making the payment and positive if you are receiving the payment.

PPYR

Payments per year. Sets the number of payments made per year for a cash flow calculation. From the Numeric view of the Finance app, enter a value for P/YR.

In a program, type:

```
n ▶ PPYR
```

where n > 0

PV

Present value. Sets the present value of an investment. From the Numeric view of the Finance app, enter a value for PV.

In a program, type:

```
n ▶ PV
```

Note: negative values represent an investment or loan.

GSize

Group size. Sets the size of each group for the amortization table. From the Numeric view of the Finance app, enter a value for Group Size.

In a program, type:

```
n ▶ GSize
```

Linear Solver app variables

The following variables are used by the Linear Solver app. They correspond to the fields in the app's Numeric view.

LSystem

Contains a 2x3 or 3x4 matrix which represents a 2x2 or 3x3 linear system. From the Numeric view of the Linear Solver app, enter the coefficients and constants of the linear system.

In a program, type:

```
matrix▶LSystem
```

where matrix is either a matrix or the name of one of the matrix variables M0-M9.

Triangle Solver app variables

The following variables are used by the Triangle Solver app. They correspond to the fields in the app's Numeric view.

SideA

The length of Side a. Sets the length of the side opposite the angle A. From the Triangle Solver Numeric view, enter a positive value for a.

In a program, type:

```
n ▶ SideA
```

where n > 0

SideB

The length of Side b. Sets the length of the side opposite the angle B. From the Triangle Solver Numeric view, enter a positive value for b.

In a program, type:

```
n ▶ SideB
```

where n > 0

SideC

The length of Side c. Sets the length of the side opposite the angle C. From the Triangle Solver Numeric view, enter a positive value for c.

In a program, type:

```
n ▶ SideC
```

where n > 0

AngleA

The measure of angle A. Sets the measure of angle A. The value of this variable will be interpreted according to the angle mode setting (Degrees or Radians). From the Triangle Solver Numeric view, enter a positive value for angle A.

In a program, type:

```
n ▶ AngleA
```

where n > 0

AngleB

The measure of angle B. Sets the measure of angle B. The value of this variable will be interpreted according to the angle mode setting (Degrees or Radians). From the Triangle Solver Numeric view, enter a positive value for angle B.

In a program, type:

```
n ▶ AngleB
```

where n > 0

AngleC

The measure of angle C. Sets the measure of angle C. The value of this variable will be interpreted according to the angle mode setting (Degrees or Radians). From the Triangle Solver Numeric view, enter a positive value for angle C.

In a program, type:

```
n ▶ AngleC
```

where n > 0

TriType

Corresponds to the status of ⊿ in the Numeric view of the Triangle Solver app. Determines whether a general triangle solver or a right triangle solver is used. From the Triangle Solver view, tap ⊿.

In a program, type:

`0 ▶ TriType`—for the general Triangle Solver

`1 ▶ TriType`—for the right Triangle Solver

Home Settings variables

The following variables (except Ans) are found in Home Settings. The first four can all be over-written in an app's Symbolic Setup view.

Ans

Contains the last result calculated in the Home or CAS views. `Ans(n)` returns the nth result in the history of Home view. In CAS view, if Ans is a matrix, `Ans(m,n)` returns the element in row m and column n.

HAngle

Sets the angle format for the Home view. In Home Settings, choose `Degrees` or `Radians` for angle measure.

In a program, type:

`0 ▶ HAngle`—for Radians.

`1 ▶ HAngle`—for Degrees.

`2 ▶ HAngle`—for Gradians.

HDigits

Sets the number of digits for a number format other than Standard in the Home view. In Home Settings, enter a value in the second field of **Number Format**.

In a program, type:

`n ▶ HDigits`, where 0 < n < 11.

HFormat

Sets the number display format used in the Home view. In Home Settings, choose `Standard`, `Fixed`, `Scientific`, or `Engineering` in the **Number Format** field.

In a program, store one of the following the constant numbers (or its name) into the variable `HFormat`:

0 Standard

1 Fixed

2 Scientific

3 Engineering

HComplex

Enables a complex result from a real input. For example, if HComplex is set to `0`, ASIN(2) returns an error; if HComplex is set to `1`, ASIN(2) returns 1.57079632679–1.31695789692*i.

In Home Settings, select or clear the **Complex** field. Or, in a program, type:

`0 ▶ HComplex`—for OFF.

`1 ▶ HComplex`—for ON.

Date

Contains the system date. The format is YYYY.MMDD. This format is used irrespective of the format set on the Home Settings screen. On page 2 of Home Settings, enter values for `Date`.

In a program, type:

`YYYY.MMDD ▶ Date`, where YYYY are the four digits of the year, MM are the two digits of the month, and DD are the two digits of the day.

Time

Returns the current clock time in DMS format. This is similar to the TICKS variable, which contains the number of milliseconds since the computer was booted.

To set the clock time, enter `Time:= H°MM′SS′′`.

Language

Contains an integer indicating the system language. From Home Settings, choose a language for the **Language** field.

In a program, store one of the following constant numbers into the variable Language:

`1 ▶ Language` (English)

`2 ▶ Language` (Chinese)

3 ▶ `Language` (French)

4 ▶ `Language` (German)

5 ▶ `Language` (Spanish)

6 ▶ `Language` (Dutch)

7 ▶ `Language` (Portuguese)

Entry

Contains an integer that indicates the entry mode. In Home Settings, select an option for **Entry**.

In a program, enter:

0 ▶ `Entry`—for Textbook

1 ▶ `Entry`—for Algebraic

2 ▶ `Entry`—for RPN

INTEGER

Base

Returns or sets the integer base. In Home Settings, select an option for the first field next to **Integers**. In a program, enter:

0 ▶ `Base`—for Binary

1 ▶ `Base`—for Octal

2 ▶ `Base`—for Decimal

3 ▶ `Base`—for Hexadecimal

Bits

Returns or sets the number of bits for representing integers. In Home Settings, enter a value for the second field next to **Integers**. In a program, enter:

n ▶ `Bits`, where n is the number of bits.

Signed

Returns the status of, or sets a flag, indicating that the integer wordsize is signed or not. In Home Settings, check or uncheck the ± field to the right of **Integers**. In a program, enter:

0 ▶ `Signed`—for unsigned

1 ▶ `Signed`—for signed

Additional common Home variables

In addition to the Home variables that control the Home Settings, there are four additional Home variables that allow the user programmatic access to various types of Home objects.

DelHVars

`DelHVars(n)` or `DelHVars("name")` deletes the specified home user variable.

HVars

Gives access to user-defined home variables.

`HVars` returns a list of the names of all the defined home user variables.

`HVars(n)` returns the nth user-defined home variable.

`HVars("name")` returns the user-defined home variable with the given name.

`HVars(n or "name", 2)` returns the list of the parameters for that function if the variable is a user-defined function; otherwise returns 0.

`HVars(n):=value` stores value in the nth home user variable.

`HVars("name"):=value` stores value in the home user variable called "name". If no such variable exists, this creates it.

`HVars(n or "name", 2):= {"Param1Name", ..., "ParamNName"}` assumes that the specified user variable contains a function and specifies what the parameters of that function are.

Notes

The Notes variable gives access to the notes saved in the calculator.

`Notes` returns a list of the names of all the notes in the calculator.

`Notes(n)` returns the contents of the nth note in the calculator (1 to NbNotes).

`Notes("name")` returns the contents of the note titled "name".

This command can also be used to define, redefine, or clear a note.

`Notes(n):="string"` sets the value of note n. If the string is empty, the note is deleted.

`Notes("name"):="string"` sets the value of note "name". If string is empty, the note is deleted. If there is no note called "name", it is created with string as content.

Programs

The Programs variable gives access to the programs saved in the calculator.

`Programs` returns the list of the names of all the programs in the calculator.

`Programs(n)` returns the contents of the nth program in the calculator (1 to NbPrograms).

`Programs(n):="string"` sets the program source code for program n. If string is empty, the program is deleted.

`Programs("name")` returns the source of program "name".

`Programs("name"):="string"` sets program "name" source code to string. If string is empty, the program is deleted. If there is no program called "name", it is created.

TOff

TOff contains an integer that defines the number of milliseconds until the next calculator auto shutoff. The default is 5 minutes, or #493E0h (5*60*1000 milliseconds).

Valid ranges are from #1388h to #3FFFFFFFh.

Symbolic Setup variables

The following variables are found in the Symbolic setup of an app. They can be used to overwrite the value of the corresponding variable in Home Settings.

AAngle

Sets the angle mode.

From Symbolic setup, choose `System`, `Degrees`, or `Radians` for angle measure. `System` (default) will force the angle measure to agree with that in Home Settings.

In a program, type:

`0 ▶ AAngle`—for System (default)

`1 ▶ AAngle`—for Radians

`2 ▶ AAngle`—for Degrees

`3 ▶ AAngle`—for Gradians

AComplex

Sets the complex number mode.

From Symbolic setup, choose `System`, `ON`, or `OFF`. `System` (default) will force the complex number mode to agree with the corresponding setting in Home Settings.

In a program, type:

`0 ▶ AComplex`—for System (default)

`1 ▶ AComplex`—for ON

`2 ▶ AComplex`—for OFF

ADigits

Contains the number of decimal places to use for the Fixed, Scientific, or Engineering number formats in the app's Symbolic Setup.

From Symbolic setup, enter a value in the second field of `Number Format`.

In a program, type:

```
n ▶ ADigits
```

where 0 < n <11

AFormat

Defines the number display format used for number display in the Home view and to label axes in the Plot view.

From Symbolic setup, choose `Standard`, `Fixed`, `Scientific`, or `Engineering` in the **Number Format** field.

In a program, store the constant number into the variable `AFormat`.

0 System

1 Standard

2 Fixed

3 Scientific

4 Engineering

Example:

```
3 ▶ AFormat
```

Results variables

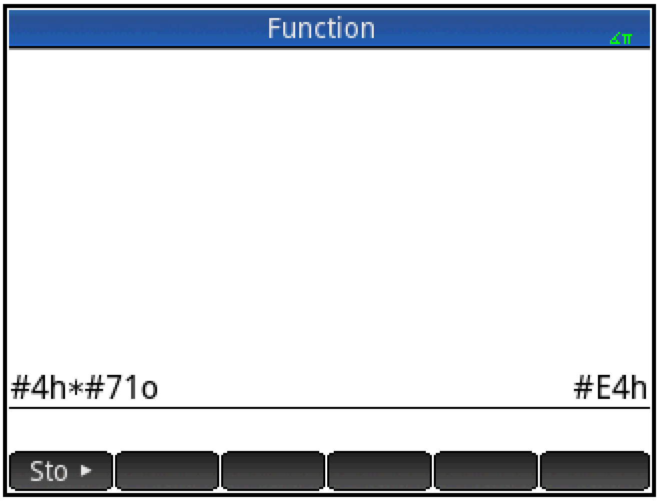
The Function, Statistics 1Var, Statistics 2Var, and Inference apps offer functions that generate results that can be reused outside those apps (such as in a program). For example, the Function app can find a root of a function, and that root is written to a variable called Root. That variable can then be used elsewhere.

The results variables are listed with the apps that generate them.

29 Basic integer arithmetic

The common number base used in contemporary mathematics is base 10. By default, all calculations performed by the HP Prime are carried out in base 10, and all results are displayed in base 10.

However, the HP Prime enables you to carry out integer arithmetic in four bases: decimal (base 10), binary, (base 2), octal (base 8), and hexadecimal (base 16). For example, you could multiply 4 in base 16 by 71 in base 8 and the answer is E4 in base 16. This is equivalent in base 10 to multiplying 4 by 57 to get 228.



```
Function
#4h*#71o                                   #E4h
```

You indicate that you are about to engage in integer arithmetic by preceding the number with the pound symbol (#, gotten by pressing ALPHA alpha 3 π #). You indicate what base to use for the number by appending the appropriate base marker:

Base marker	Base
[blank]	Adopt the default base (see The default base on page 640)
d	decimal
b	binary
o	octal
h	hexadecimal

Thus #11b represents 3_{10}. The base marker *b* indicates that the number is to interpreted as a binary number: 11_2. Likewise #E4h represents 228_{10}. In this case, the base marker *h* indicates that the number is to interpreted as a hexadecimal number: $E4_{16}$.

Note that with integer arithmetic, the result of any calculation that would return a remainder in floating-point arithmetic is truncated: only the integer portion is presented. Thus #100b/#10b gives the correct answer: #10b (since $4_{10}/2_{10}$ is 2_{10}). However, #100b/#11b gives just the integer component of the correct result: #1b.

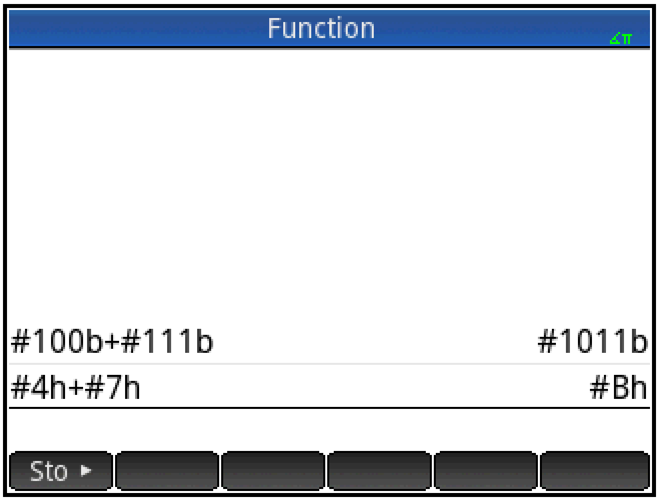
Note too that the accuracy of integer arithmetic can be limited by the integer wordsize. The wordsize is the maximum number of bits that can represent an integer. You can set this to any value between 1 and 64. The smaller the wordsize, the smaller the integer that can be accurately represented. The default wordsize is 32, which is adequate for representing integers up to approximately 2 × 10^9. However, integers larger than that would be truncated, that is, the most significant bits (that is, the leading bits) would be dropped. thus the result of any calculation involving such a number would not be accurate.

The default base

Setting a default base only affects the entry and display of numbers being used in integer arithmetic. If you set the default base to binary, 27 and 44 will still be represented that way in Home view, and the result of adding those numbers will still be represented as 71. However, if you entered #27b, you would get a syntax error, as 2 and 7 are not integers found in binary arithmetic. You would have to enter 27 as #11011b (since 27_{10}=11011_2).

Setting a default base means that you do not always have to specify a base marker for numbers when doing integer arithmetic. The exception is if you want to include a number from the non-default base: it will have to include the base marker. Thus if your default base is 2 and you want to enter 27 for an integer arithmetic operation, you could enter just #11011 without the *b* suffix. But if you wanted to enter $E4_{16}$, you need to enter it with the suffix: #E4h. (The HP Prime adds any omitted base markers when the calculation is displayed in history.)

Note that if you change the default base, any calculation in history that involves integer arithmetic *for which you did not explicitly add a base marker* will be redisplayed in the new base. In the following figure, the first calculation explicitly included base markers (*b* for each operand). The second calculation was a copy of the first but without the base markers. The default base was then changed to hex. The first calculation remained as it was, while the second—without base markers being explicitly added to the operands—was redisplayed in base 16.

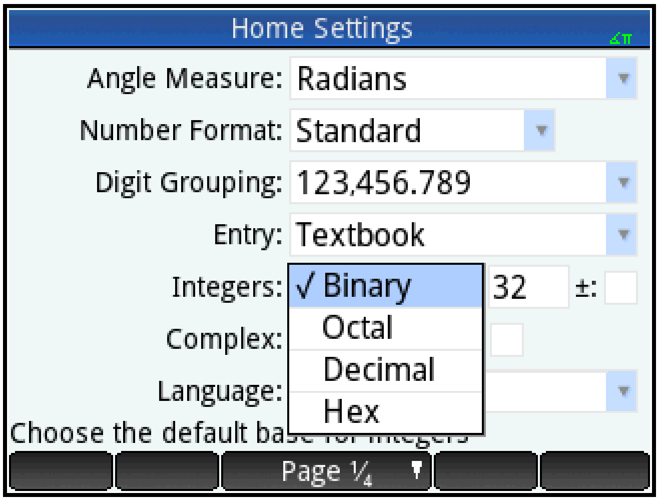


```
#100b+#111b                #1011b
#4h+#7h                       #Bh
```

Changing the default base

The calculator’s default base for integer arithmetic is 16 (hexadecimal). To change the default base:

1. Display the **Home Settings** screen: Shift Settings



Home Settings

Angle Measure: Radians

Number Format: Standard

Digit Grouping: 123,456.789

Entry: Textbook

Integers: √ Binary 32 ±: □

Complex: □

Language:

Choose the default ba

2. Choose the base you want from the **Integers** menu: **Binary**, **Octal**, **Decimal**, or **Hex**.

3. The field to the right of Integers is the wordsize field. This is the maximum number of bits that can represent an integer. The default value is 32, but you can change it to any value between 1 and 64.

4. If you want to allow for signed integers, select the **±** option to the right of the wordsize field. Choosing this option reduces the maximum size of an integer to one bit less than the wordsize.

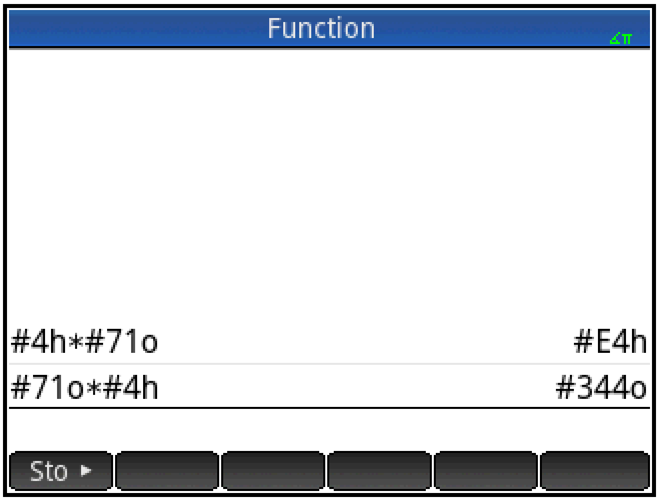
Examples of integer arithmetic

The operands in integer arithmetic can be of the same base or of mixed bases.

Integer calculation	**Decimal equivalent**
#10000b+#10100b =#100100b	16 + 20 = 36
#71o–#10100b = #45o	57 – 20 = 37
#4Dh * #11101b = #8B9h	77 × 29 = 2233
#32Ah/#5o = #A2h	810/5 = 162

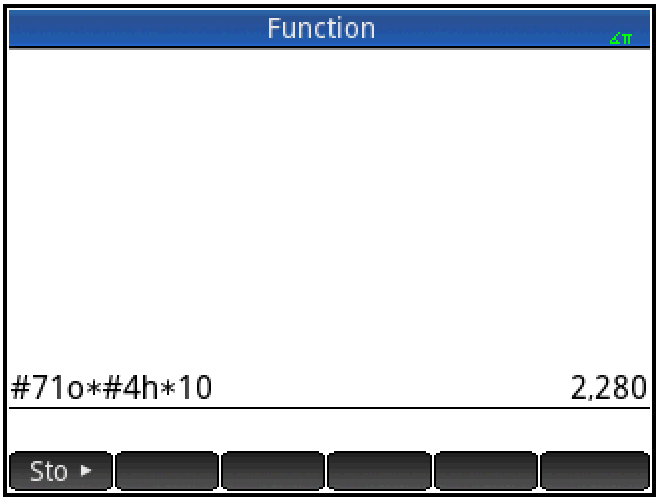
Mixed-base arithmetic

With one exception, where you have operands of different bases, the result of the calculation is presented in the base of the first operand. The following figure shows two equivalent calculations: the first multiplies 4_{10} by 57_{10} and the second multiplies 57_{10} by 4_{10}. Obviously the results too are mathematically equivalent. However, each is presented in the base of the operand entered first: 16 in the first case and 8 in the second.



```
Function
#4h*#71o                #E4h
#71o*#4h               #344o
Sto ▸
```

The exception is if an operand is not marked as an integer by preceding it with #. In these cases, the result is presented in base 10.



```
Function
#71o*#4h*10            2,280
Sto ▸
```

Integer manipulation

The result of integer arithmetic can be further analyzed, and manipulated, by viewing it in the **Edit Integer** dialog.

1. In Home view, use the cursor keys to select the result of interest.
2. Press Shift − Base (Base).

   The **Edit Integer** dialog appears. The **Was** field at the top shows the result you selected in Home view.

   The hex and decimal equivalents are shown under the **Out** field, followed by a bit-by-bit representation of the integer.

   Symbols beneath the bit representation show the keys you can press to edit the integer. (Note that this doesn’t change the result of the calculation in Home view.) The keys are:

- ◀ or ▶ (Shift): these keys shift the bits one space to the left (or right). With each press, the new integer represented appears in the **Out** field (and in the hex and decimal fields below it).

- ▲ or ▼ (Bits): these keys increase (or decrease) the wordsize. The new wordsize is appended to the value shown in the **Out** field.

- +/− (Neg): returns the two's complement (that is, each bit in the specified wordsize is inverted and one is added. The new integer represented appears in the **Out** field (and in the hex and decimal fields below it).

- + or − (Cycle base): displays the integer in the **Out** field in another base.

Menu buttons provide some additional options:

Reset : returns all changes to their original state

Base : cycles through the bases; same as pressing +

Signed : toggles the wordsize between signed and unsigned

NOT : returns the one's complement (that is, each bit in the specified wordsize is inverted: a 0 is replaced by 1 and a 1 by 0. The new integer represented appears in the Out field (and in the hex and decimal fields below it).

Edit : activates edit mode. A cursor appears and you can move about the dialog using the cursor keys. The hex and decimal fields can be modified, as can the bit representation. A change in one such field automatically modifies the other fields.

OK : closes the dialog and saves your changes. If you don't want to save your changes, press Esc instead.

3. Make whatever changes you want.

4. To save your changes, tap OK ; otherwise press Esc .

NOTE: If you save changes, the next time you select that same result in Home view and open the **Edit Integer** dialog, the value shown in the **Was** field will be the value you saved, not the value of the result.

Base functions

Numerous functions related to integer arithmetic can be invoked from Home view and within programs:

BITAND	BITNOT	BITOR
BITSL	BITSR	BITXOR
B→R	GETBASE	GETBITS
R→B	SETBASE	SETBITS

These are described in Integer on page 604.

30 Appendix A – Glossary

app

A small application, designed for the study of one or more related topics or to solve problems of a particular type. The built-in apps are Function, Advanced Graphing, Geometry, Spreadsheet, Statistics 1Var, Statistics 2Var, Inference, DataStreamer, Solve, Linear Solver, Triangle Solver, Finance, Parametric, Polar, Sequence, Linear Explorer, Quadratic Explorer, and Trig Explorer. An app can be filled with the data and solutions for a specific problem. It is reusable (like a program, but easier to use) and it records all your settings and definitions.

button

An option or menu shown at the bottom of the screen and activated by touch. Compare with *key*.

CAS

Computer Algebra System. Use the CAS to perform exact or symbolic calculations. Compare to calculations done in Home view, which often yield numerical approximations. You can share results and variables between the CAS and Home view (and vice versa).

catalog

A collection of items, such as matrices, lists, programs and the like. New items you create are saved to a catalog, and you choose a specific item from a catalog to work on it. A special catalog that lists the apps is called the Application Library.

command

An operation for use in programs. Commands can store results in variables, but do not display results.

expression

A number, variable, or algebraic expression (numbers plus functions) that produces a value.

function

An operation, possibly with arguments, that returns a result. It does not store results in variables. The arguments must be enclosed in parentheses and separated with commas.

Home view

The basic starting point of the calculator. Most calculations can be done in Home view. However, such calculations only return numeric approximations. For exact results, you can use the CAS. You can share results and variables between the CAS and Home view (and vice versa).

input form

A screen where you can set values or choose options. Another name for a dialog box.

key

A key on the keypad (as opposed to a button, which appears on the screen and needs to be tapped to be activated).

Library

A collection of items, more specifically, the apps. See also *catalog*.

list

A set of objects separated by commas and enclosed in curly braces. Lists are commonly used to contain statistical data and to evaluate a function with multiple values. Lists can be created and manipulated by the List Editor and stored in the List Catalog.

matrix

A two-dimensional array of real or complex numbers enclosed by square brackets. Matrices can be created and manipulated by the Matrix Editor and stored in the Matrix Catalog. Vectors are also handled by the Matrix Catalog and Editor.

menu

A choice of options given in the display. It can appear as a list or as a set of touch buttons across the bottom of the display.

note

Text that you write in the Note Editor. It can be a general, standalone note or a note specific to an app.

open sentence

An open sentence consists of two expressions (algebraic or arithmetic), separated by a relational operator such as =, <, etc. Examples of open sentences include $y^2<x^{-1}$ and $x^2–y^2=3+x$.

program

A reusable set of instructions that you record using the Program Editor.

variable

A name given to an object—such as a number, list, matrix, graphic and so on—to assist in later retrieving it. The Sto ▸ command assigns a variable, and the object can be retrieved by selecting the associated variable from the variables menu (Vars).

vector

A one-dimensional array of real or complex numbers enclosed in single square brackets. Vectors can be created and manipulated by the Matrix Editor and stored in the Matrix Catalog.

views

The primary environments of HP apps. Examples of app views include Plot, Plot Setup, Numeric, Numeric Setup, Symbolic, and Symbolic Setup.

31 Appendix B – Troubleshooting

Calculator not responding

If the calculator does not respond, you should first try to reset it. This is much like restarting a PC. It cancels certain operations, restores certain conditions, and clears temporary memory locations. However, it does not clear stored data (variables, apps, programs, etc.).

To reset

Turn the calculator over and insert a paper clip into the Reset hole just above the battery compartment cover. The calculator will reboot and return to Home view.

If the calculator does not turn on

If the HP Prime does not turn on, follow the steps below until the calculator turns on. You may find that the calculator turns on before you have completed the procedure. If the calculator still does not turn on, contact Customer Support for further information.

1. Charge the calculator for at least one hour.
2. After an hour of charging, turn the calculator on.
3. If it does not turn on, reset the calculator as per the preceding section.

Operating limits

Operating temperature: 0º to 45ºC (32º to 113ºF).

Storage temperature: –20º to 65ºC (–4º to 149ºF).

Operating and storage humidity: 90% relative humidity at 40ºC (104ºF) maximum. Avoid getting the calculator wet.

The battery operates at 3.7V with a capacity of 1500mAh (5.55Wh).

Status messages

The table below lists the most common general error messages and their meanings. Some apps and the CAS have more specific error messages that are self-explanatory.

Message	Meaning
Bad argument type	Incorrect input for this operation.
Insufficient memory	You must recover some memory to continue operation. Delete one or more customized apps, matrices, lists, notes, or programs.
Insufficient statistics data	Not enough data points for the calculation. For two-variable statistics there must be two columns of data, and each column must have at least four numbers.
Invalid dimension	Array argument had wrong dimensions.

Message	Meaning
Statistics data size not equal	Need two columns with equal numbers of data values.
Syntax error	The function or command you entered does not include the proper arguments or order of arguments. The delimiters (parentheses, commas, periods, and semi-colons) must also be correct. Look up the function name in the index to find its proper syntax.
No functions checked	You must enter and check an equation in the Symbolic view before entering the Plot view.
Receive error	Problem with data reception from another calculator. Re-send the data.
Undefined name	The global variable named does not exist.
Out of memory	You must recover a lot of memory to continue operation. Delete one or more customized apps, matrices, lists, notes, or programs.
Two decimal separators input	One of the numbers you have entered has two or more decimal points.
X/0	Division by zero error.
0/0	Undefined result in division.
LN(0)	LN(0) is undefined.
Inconsistent units	The calculation involves incompatible units (that is, adding length and mass).

Index