Help Topics Tree 13217	Help Text
Main	Main Help This is the online help system for your HP Prime. Throughout the help system, the word "press" always refers to a physical key on the Prime keyboard. The word "tap" always refers to an object displayed on the touchscreen, frequently an item in the software menu (a menu "button") at the bottom of the display. Press the top or bottom of the rocker wheel, drag or swipe the touchscreen, or tap either side of ▲ Page ▼ to scroll through any help page (such as this one). To exit a help page, tap OK or Esc. To see all the help topics, tap Tree. • Navigate using the rocker wheel, drag, or swipe. • Tap + or - or press right or left rocker wheel to expand/contract a group of topics. • Press Enter or OK to view a topic. Tap Keys then press any key on the keyboard to see the help related to that key. Tap Search, enter search text (hold Alpha key to enter alphabetic characters) and press Enter or tap OK. If 'Find in content' is not checked, only the help titles will be searched. As a general rule, pressing Help will display the help page most relevant to the current screen or selection. For example, pressing Help while the Application Library is open will display an overview of whatever app is currently selected. Once an app is open, there is a separate help page for each app view (Symbolic, Plot, Numeric, etc.) which will explain the purpose of the view and the menu items at the bottom of the screen.
About HP Prime	HP Prime Graphing Calculator Software Version: 2017 12 11 (13217) BETA Hardware Version: Emu CAS Version: 1.4.9 Serial Number: 4CY3420785 Operating System: © 2017 HP Development Company, L.P.
Thanks	Jeri, Teddy, Mark, Ruth, Dave Bernard, Conrad, Cyrille, Gerald, GT, Jean-Yves, Jeff, Louis, Matt, Rian, Tim Angie, Bill, Chris, Craig, Glenn, Jason, Jason, Julia, Mike, Michelle, Nick, Shan-shan, Soolee, Syed, Geoff, Charlotte, George, Jack, James, Katy, Marcia, Mike, Tyler, Anusha, Archana, Dean Eddie, Edwin, Erik, Fabien, Felix, Gilles, Han, Joe, Johnny, Katie, Marv, Namir, Natasha, Patrice, Stefan, Wes
Home View	The base screen of the calculator is the Home view. Most calculations are done here. Enter an expression in the same left-to-right order in which you would write it. You can also do calculations using Reverse Polish Notation (RPN) if you have selected RPN as your preferred entry option in Home Settings. An expression can contain numbers, function calls, variables, lists, and matrices. To enter an expression, press the appropriate keys or select items from one of the menus (such as the Toolbox menu). You can also enter a function name by using the Alpha keys to spell out its name. Once you have finished entering the expression, press Enter to evaluate it. You can save an answer by assigning it to a variable, and then use that variable in later calculations. After evaluation, both the expression and its result are displayed in the history section of Home view. You can scroll through the history using the rocker wheel, swipe, or drag. You can re-use an expression or result by tapping twice on it. It is copied to the entry line ready for you to edit. (You can also scroll to an item in history and tap Copy.) The Home view touch-button menu items are: • Sto ►: store data in a variable • Copy: copy the selected entry or result to the entry line at the current cursor location • Show: display the selected item in full-screen mode (with horizontal and vertical scrolling enabled) As well as the Copy button, which only works in Home view and CAS view, you can use the clipboard to copy and paste expressions. Your last few entries are automatically copied to the clipboard. Press Shift Menu (Paste) to open the clipboard. Use the rocker wheel to select an expression from the clipboard and then tap OK or press Esc. To manually add an item to the clipboard, select it and press Shift View (Copy). Press the Backspace key to delete the character to the left of the cursor on the entry line. To clear the entire entry line, press Esc. To clear the entire history, press Shift Esc (Clear).
Home Settings	Home Settings Menu Home Settings Page 1 contains the following options: • Angle Measure: Select Degrees, Radians, or Gradians • Number Format: Select Standard, Fixed, Scientific, Engineering, Floating, or Rounded • Choose Digit Grouping • Entry: Select Algebraic, RPN, or Textbook • Integers: Select the default base for integer arithmetic: Binary, Octal, Decimal, or Hex. You can also choose the word size (or number of bits) and whether integers are signed or unsigned. • Complex: Select the format for representing complex numbers and allow complex output from real input

Help Topics Tree **13217**	**Help Text**
	• Language: Select a language If 'See Symbolic Setup' appears instead of the input field for a setting, that setting is overridden by the current app. To change the setting, do so in Symbolic Setup. If you want to clear the app-specific override and return to the system-wide value of the setting, go to Symbolic Setup and change the value of the setting to 'System'.
Page 1	Home Settings Page 1 Home Settings Page 1 contains the following options: • Angle Measure: Select Degrees, Radians, or Gradians • Number Format: Select Standard, Fixed, Scientific, Engineering, Floating or Rounded • Choose Digit Grouping • Entry: Select Algebraic, RPN, or Textbook • Integers: Select the default base for integer arithmetic: Binary, Octal, Decimal, or Hex. You can also choose the word size (or number of bits) and whether integers are signed or unsigned. • Complex: Select the format for representing complex numbers and allow complex output from real input • Language: Select a language
Entry Methods	The HP Prime provides you with three ways of entering expressions in Home view. • Textbook An expression is entered in the same way as if you were writing it on paper. For example, a division will be represented by a division bar with the dividend above and the divisor below. • Algebraic An expression is entered on a single line of text. • 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. Example: Suppose you wanted to calculate 2 + 3 In Textbook and Algebraic modes, you would enter 2 + 3 Enter In RPN mode, you would enter 2 Enter 3 + If you subsequently entered 12 +, 12 is automatically added to the last answer in RPN mode.
Page 2	Home Settings Page 2 Home Settings Page 2 contains the following options: • Font Size: Select font size Small, Medium, or Large for most displayed text. • Calculator Name: Give your HP Prime a unique name. This aids recognition in wireless communication with the Connectivity Kit • Textbook Display: Toggle Textbook display on or off. • Menu Display: If selected, mathematical functions and commands are represented in menus using a descriptive name; if unselected, they are represented by their command name. • Time: Set the time and the time format. Toggle clock in title bar on or off. • Date: Set the date and the date format. • Color Theme: Choose Light or Dark, and also a highlight color.
Page 3	Home Settings Page 3 The options on Home Settings Page 3 all relate to Exam Mode. There is a separate help topic for Exam Mode.
Page 4	Home Settings Page 4 If your HP Prime supports the wireless module for wireless connectivity, the options on Page 4 enable you to join an HP Wireless Classroom Network via the wireless module and the Connectivity Kit. You need to have the wireless adapter plugged in for these options to appear.
Computer Algebra System (CAS)	The Computer Algebra System (CAS) enables you to perform symbolic calculations. By default, CAS works in exact mode, giving you symbolic or exact arithmetic results. 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. For example, 1/3 + 2/7 yields the approximate answer 0.619047619047 in Home view (with Standard numerical format), but yields the exact answer 13/21 in the CAS. CAS offers hundreds of functions, covering algebra, calculus, equation solving, polynomials, and more. You select a function from the CAS menu, one of the Toolbox menus. You can also select a CAS function from the Catlg menu (another of the Toolbox menus).
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. The label "CAS" appears 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 • Copy: copies a selected entry in history to the entry line • Show: displays the selected item in full-screen mode (with horizontal and vertical scrolling enabled)

Help Topics Tree 13217	Help Text
CAS Settings	Various settings allow you to configure how the CAS works. To display the settings, press Shift CAS. The options are spread across two pages.
Page 1	CAS Settings Page 1 The options on CAS Settings Page 1 are: • Angle Measure: Select the units for angle measurements: Radians, Degrees, or Gradians. • Number Format (first drop-down list): Select the number format for displayed solutions: Standard, 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), Hexadecimal (base 16), or 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 None, Minimum, or Maximum automatic simplification. • Exact: If checked, the CAS is in exact mode and solutions will be symbolic. If not checked, the CAS is in approximate mode and solutions will be approximate. For example, 26/5 returns 26/5 in exact mode and 5.2 in approximate mode. • Use √: If checked, polynomials are factorized using square roots. • 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	CAS Settings Page 2 The options on CAS Settings Page 2 are: • Recursive Evaluation Specify the maximum number of embedded variables allowed in an interactive evaluation. See also Recursive Replacement below. • Recursive Replacement Specify the maximum number of embedded variables allowed in a single evaluation in a program. See also Recursive Evaluation above. • Recursive Function Specify the maximum number of embedded function calls allowed. • Epsilon Any number smaller than the value specified for epsilon will be treated as zero in some numerical algorithms. • 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 Newton's method to find the roots of a polynomial.
HP apps	Much of the functionality of the HP Prime is provided in packages called HP apps. The HP Prime 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 the HP StreamSmart 410. You launch an app by first pressing the Apps key and tapping on the icon of the app you want. You can make any number of copies of an existing app either as a backup of settings and data or to customize them (through a program for example). Such apps are opened from the application library in the same way that you open a built-in app. 1. Press Apps to open the Application Library. 2. Highlight (but do not open) the app that you want to base the new app on. You can use the rocker wheel to navigate to an app without opening it. 3. Tap Save. 4. Enter a name for the new app. 5. Tap OK twice. The new app appears in the Application Library. You can now open it as you would open any app. Customizing the Advanced Graphing App You can save a sample graph from the Plot Gallery. See "Advanced Graphing app" for instructions.
Main 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, Plot, and Num 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, Shift Plot and Shift Num respectively. 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 only a Numeric view and the Quadratic Explorer has only a Plot view.

Help Topics Tree	13217	Help Text
Symbolic View		The Symbolic view varies from app to app, but its purpose is to store symbolic definitions, whether they are functions, open sentences in x and y, or definitions of statistical analyses. Use the Symbolic view to define functions and open sentences, create geometric objects, set up a hypothesis test, and define statistical analyses. Note that the Symbolic view is not used in the Spreadsheet, Explorer, Triangle Solver or Linear Solver apps.
Symbolic Setup		The only view that is common to all apps is the Symbolic Setup view. Its primary purpose is to allow you to override three of the system-wide settings specified on the Home Settings window. Tap once on a setting to display a menu of options and then choose the option you want. You can also use the rocker wheel to highlight a field and tap Choose to display the menu of options.
Plot View		The Plot view varies from app to app but its function is to show you the graphical representations of objects defined in Symbolic view. In Plot view you can see the graphs of the functions and open sentences defined in Symbolic view. In the Geometry app you can create geometric objects. In various other apps, you can explore linear, quadratic, and sinusoidal functions as well as amortization graphs. In most of the Plot views, the following gestures and features are available: • tap to jump the tracer to an x-value • flick to initiate kinetic scrolling in the desired direction • drag to scroll the window • open/closed diagonal pinch (put two fingers on the screen and move them apart or together) to zoom square in or out • open/closed horizontal pinch to zoom in or out on the x-axis • open/closed vertical pinch to zoom in or out on the y-axis • press + to zoom in or - to zoom out on the cursor location You can press Shift View (Copy) to copy the current Plot view to the clipboard as an image. Note that the Plot view is not used in the Spreadsheet, Triangle Solver, or Linear Solver apps.
Plot Setup		This view is used primarily to modify the appearance of graphs and the plotting environment. It is not used in apps that have no Plot view. In apps with this view, Page 1 of the view has fields for the X-range, Y-range and tick spacing on the axes. Page 2 is devoted to various options such as grid lines, grid dots, axes, axis labels, and cursor type. Some HP Prime apps support the use of an image as a background in Plot view. For those apps, the third page of Plot Setup is devoted to selecting the image and configuring its appearance in Plot view. Press Shift-Plot, then tap the Page-Down menu key twice to access the Image-As-Background page. The first field is a drop-down list with the following options: • No Background: No image will be used as a background (default) • Centered: the selected image will be centered, vertically and horizontally, in Plot view • Stretched: the selected image will be stretched, both horizontally and vertically (if necessary), to fit the entire display in Plot view • Best fit: the selected image will be stretched, either horizontally or vertically (but not both), to fit one of the two dimensions in Plot view • XY Range: the user will enter the x- and y-range to place the image in Plot view Next to the first field is the Opacity field. Enter an integer from 0 to 100 to indicate the level of opacity of the image, where 0 is transparent and 100 is totally opaque. Below these two fields is a swipe chooser. The swipe chooser displays all of the images associated with the current app first, followed by all the built-in images. Swipe to view the library of available images, then tap to select the one you want. Once you have selected a display option, an opacity level, and an image, the selected image will be visible in Plot view as a background, with the options and opacity level you chose. Dragging the axes or pinching to zoom in or out in Plot view only affects the view of the image if you have chosen the XY Range option. For all other options, the image is a true background and does not respond to changes in Plot view. In XY Range, the image is maintained in it x- and y-ranges as you zoom or pan, allowing you to zoom in or out on the image or pan to a particular feature. There are two special menu keys on Page 3 of Plot Setup: • More: tap to pull in an image you want from another HP Prime app • Calc: with the XY Range option, use it to calculate the fourth coordinate, given the other three More Tap the More menu key to see an input form with a drop-down list and a swipe chooser. Use the drop-down list to select an HP Prime app that currently has images associated with it. The use the swipe chooser to swipe through the images associated with an app and tap to select one. Tap the OK menu key to pull that image into your current app. Calc If you have chosen the XY RANGE option, fill in 3 of the 4 x-range and y-range values, then select the fourth and tap the Calc menu key. The value of the fourth field will be computed for you, keeping the aspect ratio of the original image intact. For information on how to associate an image with an HP Prime app, see the HP Connectivity Kit User Guide.
Numeric View		The Numeric view varies from app to app, but its purpose is to present sets of numerical values, whether function values or numerical data.

Help Topics Tree	13217	Help Text
		In Numeric view, you can explore tables of values generated by functions, make geometric measurements, do spreadsheet calculations, and enter data for statistical analyses. In most of the Numeric views, the following gestures and features are supported: • tap to select an x- or y-value. You can copy the value to the clipboard and then paste it anywhere. If you select a value of the independent variable, you can type in a real number and the table will re-configure around that value. • tap and hold, then drag to select a rectangular array of numbers. You can then copy the array and paste it into the Spreadsheet, Statistics 1Var or Statistics 2Var apps, or into the List or Matrix Editors. • flick to initiate kinetic scrolling in the desired direction • drag to scroll the window • open pinch vertically to zoom in on the currently selected row of the table • close pinch vertically to zoom out on the currently selected row of the table • press + to zoom in or - to zoom out on the currently selected row of the table, using the zoom factor set in Numeric Setup Note that the Numeric view is not used in the Explorer apps.
Numeric Setup View		The Numeric Setup view is used to determine the appearance of the Numeric view and to set the zoom factor.
Info		Add a Note to an App App Note Editor: Shift + Apps (Info) Use this editor to add a note to the current app (or modify the existing note). The note created here stays with its app when it is transferred to a PC or another HP Prime. This editor has the same functionality as the Notes Editor. The menu items are: • Edit: tap to add a new note or edit the current note • Format: displays a menu of formatting options • Style: displays a menu of style options • ▲ Page ▼: moves from page to page in a multi-page note • •: cycles through bullet styles • Insert: tap to display a menu of items that can be inserted. Press ALPHA twice to lock the alpha shift. Press it again to release the alpha shift. You can copy and paste text using Shift View (Copy) and Shift Menu (Paste) respectively.
View		View Key Press the View key to access options specific to the graphing apps (such as split-screen options and commonly used scaling options). • Split Screen: Plot Detail Splits the Plot view into two panels, with zooming only affecting the plot shown in the panel at the right. • Split Screen: Plot Table Splits the view into two panels, one showing the Plot view and the other the Numeric view. • Common zoom options • Autoscale: adjusts the range (YRange) so that at least one of the selected definitions in Symbolic view is clearly visible • Decimal: rescales both axes so that each pixel represents 0.1 units. • 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. You can also modify the View menu options to call programs you have written and attached to apps. See the User Guide for more details.
Background Image		This dialog box is available as the last page of the plot setup view of most of the apps that have a plot view. The top choose box lets you choose the image which will be displayed in the background of the app's Plot view. The list contains all the images built into the calculator for this app, plus any other images you have saved in the app file for this app (for example through the connectivity kit). The checkbox on the right of the image selection lets you choose if you want the image to be a full screen background (unchecked) or if you want to specify the Cartesian coordinates where the image will be drawn. You can control the image opacity (100 means a solid image, the smaller the number, the more the image will fade out in the background). If an image stored in an app file is selected, you can delete it by tapping the Delete menu (select the image name for the menu to appear) You can select any one of the 4 image coordinates and tap on the Calculate menu to recalculate this coordinate so that the image aspect ratio is preserved. You can import images from the other apps by tapping the Import menu.
Image import		This dialog box lets you select any image from any of the other apps (built-in or user-created) and import it as a user file in the current app. Select the image that you want to import using the choose box. You can change the target file name for the image. You can change the image size. If the Proportion check box is checked, the system will keep the image width/height ratio during resizing. When you press OK, the resized image will be copied to the current app using the specified name.

Help Topics Tree 13217	Help Text
Common App Variables	App variables store current settings and results in their respective apps. For example, the X Tick setting is stored in a variable, as is the angle measure setting and the graphing method setting. You can change a setting by storing a value to an app variable or check a setting by evaluating an app variable and looking at it value. The variables common to most apps are listed in this section. For variables specific to a particular app, see the section devoted to that app.
Common Plot View Variables	This section lists the variables common to many of the HP apps that have a Plot view. Plot variables include, among other variables specific to the app, Xmin, Xmax, Xtick, Ymin, Ymax, and Ytick. Not all apps will have Plot view variables.
Xmin	Xmin App Variable Xmin sets the minimum horizontal value of the Plot view. Xmin := n, where n is a real number
Xmax	Xmax App Variable Xmax sets the maximum horizontal value in the Plot view. Xmax := n, where n is a real number such that n>Xmin
Ymin	Ymin App Variable Ymin sets the minimum vertical value of the Plot view. Ymin := n, where n is a real number
Ymax	Ymax App Variable Ymax sets the maximum vertical value in the Plot view. Ymax := n, where n is a real number such that n>Ymin
Xtick	Xtick App Variable Xtick sets the distance between tick marks for the horizontal axis. Xtick := n, where n is a real number such that n>0
Ytick	Ytick App Variable Ytick sets the distance between tick marks for the vertical axis. Ytick := n, where n is a real number such that n>0

Axes	Axes App Variable
	Turns the display of X and Y axes in Plot View on or off.
	Axes := 0 axes on. (default)
	Axes := 1 axes off.

Labels	Labels App Variable
	Labels enables or disables drawing labels in Plot view showing X and Y ranges.
	Labels := 0 labels off (default)
	Labels := 1 labels on

GridDots	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.

GridLines	GridLines App Variable
	GridLines turns the background grid lines in Plot View on or off.
	GridLines := 0 grid lines on. (default)
	GridLines := 1 grid lines off.

Cursor	Cursor App Variable
	Cursor sets the cursor type in Plot view.
	Cursor := 0 normal (default)
	Cursor := 1 inverted
	Cursor := 2 blinking

ImageName	ImageName App Variable ImageName controls which image is set as a background in plot views. ImageName := name, where name is a file name string (such as "photo1").
ImageDisplay	ImageDisplay App Variable ImageDisplay controls how a background image is displayed 0 for no background 1 for centered 2 for stretched 3 for best fit 4 for XY range ImageName controls which image is used
ImageOpacity	ImageOpacity App Variable ImageOpacity controls the opacity of the background image, if present. Use 100 for an unmodified image and smaller values for less opacity (more transparency) in the image. ImageOpacity := n, where n is a real number between 0 and 100.
ImageXmin	ImageXmin App Variable ImageXmin sets the minimum horizontal value occupied by the background image, if present. ImageXmin := n, where n is a real number.
ImageXmax	ImageXmax App Variable

Help Topics Tree 13217	Help Text
	ImageXmax sets the maximum horizontal value occupied by the background image, if present. ImageXmax := n, where n is a real number.
ImageYmin	ImageYmin App Variable ImageYmin sets the minimum vertical value occupied by the background image, if present. ImageYmin := n, where n is a real number.
ImageYmax	ImageYmax App Variable ImageYmax sets the maximum vertical value occupied by the background image, if present. ImageYmax := n, where n is a real number.
PlotMethod	PlotMethod App Variable PlotMethod sets the graphing method: PlotMethod := 0, Adaptive (default): gives the most accurate results but takes longer to produce the graph PlotMethod := 1, Fixed-step segments: this method samples x-values, computes their corresponding y-values, and then plots and connects the points. PlotMethod := 2, Fixed-step dots: this works like the fixed-step segments method but does not connect the points.
Recenter	Recenter App Variable Recenter specifies if the plot recenters on the cursor during Zoom operations in plot view. Recenter := 0 — recenter on cursor (default) Recenter := 1 — do not recenter on cursor
Xzoom	Xzoom App Variable Xzoom sets the horizontal zoom factor. Xzoom := n, where n is a real number such that n>0 (default is 2)
Yzoom	Yzoom App Variable Yzoom sets the vertical zoom factor. Yzoom := n, where n is a real number such that n>0 (default is 2)
Tmin	Tmin App Variable Tmin contains the starting value for T in parametric Plot view. Tmin := n, where n is a real number
Tmax	Tmax App Variable Tmax contains the final value for T in parametric Plot view. Tmax := n , where n is a real number such that n>Tmin
Tstep	Tstep App Variable Tstep contains the step value (increment) of T in parametric Plot view. Tstep := n, where n is a real number such that n>0
Nmin	Nmin App Variable Nmin contains the starting value for N in Sequence App Plot view. Nmin := n, where n is a counting number
Nmax	Nmax App Variable Nmax contains the final value for N in Sequence app Plot view. Nmax := n, where n is a counting number such that n>Nmin
θmin	θmin App Variable θmin contains the starting value for θ in Polar app Plot view. θmin := n, where n is a real number
θmax	θmax App Variable θmax contains the final value for θ in Polar app Plot view. θmax := n, where n is a real number such that n>θmin
θstep	θstep App Variable θstep contains the stepping value (increment) of θ in Polar app Plot view. θstep := n, where n is a real number such that n>0
Common Numeric View Variables	This section lists the variables common to many of the HP apps that have a Numeric view.
NumStart	NumStart App Variable NumStart sets the starting value for the independent variable in Numeric view when Automatic is the Num Type. NumStart := n, where n is a real number
NumStep	NumStep App Variable NumStep sets the step size (increment) for the independent variable in Numeric view when Automatic is the Num Type. NumStep := n, where n is a real number such that n>0
NumType	NumType App Variable Determines how the independent variable values in Numeric view are generated: • Automatic: uses the NumStart and NumStep values to create the independent variable values. • Build Your Own: you enter the independent variable values one by one. NumType := 0 for Automatic (default) NumType := 1 for BuildYourOwn
NumIndep	NumIndep App Variable

Help Topics Tree 13217	Help Text
	NumIndep contains the list of values you have entered in Numeric view when you have chosen BuildYourOwn mode. In the case of the Advanced Graphing app, the list is of pairs of values.
NumZoom	NumZoom App Variable NumZoom sets the Numeric view factor. NumZoom := n, where n>0 (default is 4)
AFiles	App Files Each HP Prime app can have any number of files associated with it. These files are sent with the app. AFiles returns the list of all these files. AFiles("name") returns the content of the file with the given name. AFiles("name"):= object stores the object in the file with the given name.
AFilesB	Binary App Files 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.
DelAFiles	Delete App Files DelAFiles("name") Deletes the specified file associated with an HP app.
ANote	App Note 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	App Program AProgram returns the program associated with an HP Prime app. AProgram:="string" sets the program associated with the app to contain the string.
AVars	App Variables 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, creates a new one. Note that once an app variable is created through AVars("name"):= value, you can use the variable by simply typing the variable name.
DelAVars	Delete App Variables DelAVars(n) or DelAVars("name") erases the specified app variable.
Common App Mode Variables	This section lists the Mode variables used by the HP apps. They are found in the Symbolic Setup view of each app. If the value is set to 0, the settings of the Home view will be used. Else, the setting in the app Symbolic Setup view overrides the home view setting.

AAngle	AAngle App Variable
	AAngle sets the angle mode to Degrees, Radians, or System, for the app.
	AAngle := 0 use Home view setting
	AAngle := 1 for Radians
	AAngle := 2 for Degrees
	AAngle := 3 for Gradians

ADigits	ADigits App Variable ADigits sets the number of decimal places to use for the number format. ADigits := n, where n is an integer such that 0≤n≤11

AFormat	AFormat App Variable
	AFormat defines the format of the number display for the app.
	AFormat := 0 use Home view setting
	AFormat := 1 for Standard
	AFormat := 2 for Fixed
	AFormat := 3 for Scientific
	AFormat := 4 for Engineering
	AFormat := 5 for Floating
	AFormat := 6 for Rounded

AComplex	AComplex App Variable
	AComplex sets the complex number mode for the app.
	AComplex := 0 use Home view setting
	AComplex := 1 for ON
	AComplex := 2 for OFF

Function app	The Function app enables you to explore up to 10 real-valued functions of the form Y=f(X) (for example, y=2x+3). Once you have defined a function you can: • View its graphical representation in the Cartesian plane • Find roots, points of intersection, slope, signed area, and extrema

Help Topics Tree 13217	Help Text
	• Create tables of function values • Zoom in or out in the representation To launch the Function app, go to the Application Library and tap the Function app icon. You can also use the rocker wheel to select the Function app icon, then tap Start or press Enter to launch the app.
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 • ✓: toggles the selected function on or off for plotting and table-building • X: a typing aid for entering functions of x • Show: displays the selected function in full-screen mode with horizontal and vertical scrolling enabled. • Eval: resolves references to other functions, such as F2(X)=F1(X-1)+2 • Choose: select a color for the graph Examples: SIN(6*X)*e^X SIN(2*X)*√(64-X²)
Function Plot View	The Plot View is used to display and explore the graphs of the functions defined in Symbolic View. Tap Menu to toggle the menu on and off. The menu buttons are: • Zoom: opens the Zoom menu, with options to zoom in or out, etc. • Trace: toggles the tracing cursor on and off. • Go To: lets you move the cursor to any point on the curve by entering its x-coordinate. • Fcn: opens the Function menu, with the following commands: • Sketch: sketch a function with your finger and HP Prime will match it with a function graph • Transform: drag to translate or pinch to die the current function • Defn: displays the definition of the function being traced, with options to edit the expression or transform the graph • Root: find the root of the current function that is closest to the tracer • Intersection: find the intersection of the current function with one other function, closest to the tracer • Slope: find the slope of the current function at the current tracer location • Signed Area: find signed area under a curve or between two curves • Extremum: find an extremum for the current function, closest to the tracer location • Tangent: draw the tangent to the current function through the current trace point • Menu: toggles the menu on and off Sketch, Transform, and Definition Select Sketch to put Plot view in Sketch mode. Sketch a simple function (linear, quadratic, etc.) with your finger. When you lift your finger from the screen, your sketch will be replaced with the closest graph that HP Prime can manage; tap OK and its expression will be added to Symbolic view. Select Transform to translate and dilate the current function graph. Drag to translate and pinch vertically or horizontally to dilate the graph. The expression will respond accordingly. Tap Simplify to simplify the current expression. Tap Form to select an alternate form for your function equation. Select Definition to view the expression for the current graph. In the Definition Box, you can tap Edit to edit the expression; when you are done editing, tap OK to see the change in the graph. You can also tap Transform to manipulate the current graph. Tap OK when you are done to return to the Definition Box. Tap the down-arrow menu key again to close the box. Use the left- and right-cursor keys or tap to trace along a function. Use the up- and down-cursor keys to switch from one function to another. Press + to zoom in on the current cursor location and press - to zoom out. Set the zoom factor under the Zoom menu. You can also use all the gestures common to the Plot views. See Plot View for more details.
Function Plot Setup	The Function Plot Setup enables you to control the appearance of the graph window, including the appearance of the cursor, whether or not the axes are drawn, etc. This setup has two pages. On the first page, the fields are: • X Rng: the horizontal graphing range • Y Rng: the vertical graphing range • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are: • Edit: opens an edit line to edit the value of the selected field • Page 1/2 ▼: displays the second page of the setup On the second page, the fields are: • Axes: toggles axes on and off • Labels: toggles axis labels on and off • Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off

Help Topics Tree 13217	Help Text
	• Cursor: choose between standard, inverting, and blinking cursors • Method: choose between Adaptive, Fixed-Step Segments, and Fixed-Step Dots The menu buttons on the second page are: • ✓: toggles the current setting on or off • Choose: make a choice from a choose box • ▲ Page 2/2: returns to the first page of the setup The Method field requires an explanation. By default, the HP Prime uses the Adaptive method, an advanced method that gives very accurate results. You can choose the more traditional method, called Fixed-Step Segments, which samples x-values, computes their corresponding y-values, and then plots and connects the points. Or you can choose Fixed-Step Dots, which works like Fixed-Step Segments but does not connect the points.
Function Numeric View	The Function Numeric View is designed to create and explore a table of X/Y values, based on the function(s) defined in the Symbolic View. Tap any row of the x-column, enter any real value, and tap OK. The table will reconfigure. You can also zoom in or out on any row in the table. Press + to zoom in on a row of the table and - to zoom out. The menu buttons are: • Zoom: zooms in or out on a highlighted row of the table. Note that in Numeric view, zooming changes the increment between consecutive x-values. Zooming in decreases the increment; zooming out increases the increment. The values in the row you zoom in or out on remain the same. • More: opens a menu with options for selecting multiple cells; you can then copy and pasted them elsewhere. • Go To: jumps to a specified value of the independent variable • Defn: displays the definition of the selected column The More menu The More menu contains the following options: • Select • Row: selects the row that contains the currently selected cell; the row can then be copied to paste elsewhere • Swap Ends: this option is available once a multi-cell selection has been made. Swaps the beginning and ending cells of the current selection. • Include Headers: the same as Select Row, except that the row headers are selected as well • Selection: toggles selection mode on and off • Font size: select from a small, medium, or large font size You can also use any of the gestures common to the Numeric views. See Numeric View for more details.
Function Numeric Setup	The Function Numeric Setup enables you to control the appearance of the table in the Numeric View, including which x-value is at the top of the table, the step between x-values in the table, and what the zoom factor is used for zooming in and out on a row of the table. The fields are: • Num Start: the first value of x shown in the table • Num Step: the increment between consecutive x-values • Num Zoom: the zoom factor for zooming • Num Type: choose between table types • Automatic: provides x- and function-values • BuildYourOwn: you supply x-values; the App provides the corresponding function-values The menu buttons are: • Edit: opens an edit line to edit the current value in a field • Choose: make a choice from a choose box • Plot→: sets Num Start and Num Step so that the Numeric view table independent variable values match the independent variable values while tracing in Plot view
Function App Variables	To display the variables relating to the Function app, press Vars, tap App and select Function. The Function app variables are grouped in 5 categories: • Results • Symbolic • Plot • Numeric • Modes The Plot, Numeric, and Modes app variables are discussed under Common App Variables. The Results and Symbolic function app variables are discussed in the following sections.
Symbolic Variables	Function App Symbolic Variables The Function app symbolic variables are F1 through F9 and F0. These variables contain algebraic expressions in X. Fn := f(X), where n is an integer between 0 and 9 inclusive and f(X) is an algebraic expression in X. Example: F1:='X+3' → 'X+3'
Results Variables	Function App Results Variables The Function app results variables store the results of calculations made from the Fcn menu in Plot view.

Help Topics Tree				13217	Help Text
				Extremum	Extremum App Variable Extremum contains the last value found by the Extremum function in the Plot view Fcn menu.
				Isect	Isect App Variable Isect contains the last value found by the Intersection function in the Plot view Fcn menu.
				Root	Root App Variable Root contains the last value found by the Root function in the Plot view Fcn menu.
				SignedArea	SignedArea App Variable SignedArea contains the last value found by the Signed area function in the Plot view Fcn menu.
				Slope	Slope App Variable Slope contains the last value found by the Slope function in the Plot view Fcn menu.
		Function App Functions			The Function app functions provide the same functionality found in the Function app's Plot view under the Fcn menu. All of 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 App Function Syntax: AREA(Fn, [Fm], Lower, Upper) Signed area under a curve or between curves. Approximates 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. Example: AREA(-X,X^2-2,-2,1) → 4.5
			EXTREMUM		EXTREMUM App Function Syntax: EXTREMUM(Fn, [Guess]) Extremum of a function. Finds the extremum (if one exists) of the function Fn that is closest to the X-value Guess. Examples: EXTREMUM(X^2-X-2,0) → 0.5 F1:='X^2-4X+3'; EXTREMUM(F1,1) → 2
			ISECT		ISECT App Function Syntax: ISECT(Fn, Fm, [Guess]) Intersection of two functions. Finds the intersection (if one exists) of the two functions Fn and Fm that is closest to the X-value Guess. Example: ISECT(X,3-X,2) → 1.5
			ROOT		ROOT App Function Syntax: ROOT(Fn, [Guess]) Root of a function. Finds the root of the function Fn (if one exists) that is closest to the X-value Guess. Example: ROOT(3-X^2,2) → 1.73205080757
			SLOPE		SLOPE App Function Syntax: SLOPE(Fn,Value) Slope of a function. Returns the slope of the function Fn at the X-value Value (if it exists). Example: SLOPE(3-X^2,2) → -4
	Advanced Graphing App				The Advanced Graphing app enables you to define and explore the graphs of symbolic open sentences depending on x or y, both or neither. You can plot conic sections, polynomials in standard or general form, inequalities, and functions. Once you have defined an open sentence, you can plot it and view a table of values that shows when the open sentence is satisfied. To launch the Advanced Graphing app, go to the Application Library and tap the Advanced Graphing app icon. You can also use the rocker wheel to select the Advanced Graphing app icon, then tap Start or press Enter to launch the app. PLOT GALLERY To get an idea of the sorts of graphs you can plot with the Advanced Graphing app, open the app, go to Plot view, press the Menu key, and choose Visit Plot Gallery from the menu. A gallery of plots opens, with the definitions that generated a plot shown at the bottom of the screen. Press the rocker wheel left/right to move from plot to plot in the gallery. To save a plot for your own exploration, tap Save and enter a name for a new customized app. You can then explore that new app as you would explore the built-in Advanced Graphing app.
		Advanced Graphing Symbolic View			The Symbolic View of the Advanced Graphing app enables you to define up to ten open sentences. Highlight one of the ten definition fields (labeled V0 to V9) and begin entering or editing an open sentence. Menu Buttons: PRIMARY VIEW • Edit: copies the selected definition to the entry line and activates the cursor

Help Topics Tree 13217	Help Text
	• ✓: selects or deselects the highlighted definition. (Only selected definitions are plotted.) • X: a typing aid for entering an X • Y: a typing aid for entering a Y • Show: displays the selected definition in full-screen mode with horizontal and vertical scrolling enabled. • Eval: resolves references when one definition is defined in terms of another • Choose: select a color for the graph Menu Buttons: EDIT VIEW • =: inserts the equal sign • ≤,≥,≠: opens a palette of relational operators • X: a typing aid for entering an X • Y: a typing aid for entering a Y • Cancel: closes the edit line without making changes • OK: saves the changes and closes the edit line Example: SIN((1+SIN(X+Y))*ASIN(.5*(SIN(X)+SIN(Y)))+(1+SIN(X-Y))*ASIN(.5*(COS(X)+COS(Y))))=0
Advanced Graphing Plot View	The Plot View is used to display and explore the graphs of the open sentences defined in the Symbolic View. Note that only those definitions selected (checked) in Symbolic view are plotted. Tap Menu to show or hide the other menu buttons. Menu Buttons: • Zoom: opens the Zoom menu, with options to zoom in or out, etc. • Trace: provides various tracing options • Go To: moves the cursor to the location you specify • Defn: displays the definition of the selected open sentence (which you can then edit without needing to go back to Symbolic view) • Menu: shows or hides the other menu buttons The tracer can be customized to trace along the edge of the graph, or to just trace to points of interest such as x-intercepts, y-intercepts, extrema, or inflection points. The gestures common to the Plot views are supported here as well. See Plot View for more details.
Advanced Graphing Plot Tracing	The tracer in the Plot view of the Advanced Graphing App can be set to trace any of the following configurations: • Edge values • Points of Interest, such as: • X-intercepts • Y-intercepts • Horizontal extrema • Vertical extrema • Inflection points In Plot view, tap the Trace menu key and make a selection. For example, if you are tracing on the graph of Y=sin(X) and choose to trace X-intercepts, the tracer will jump from one X-intercept to the next. If the tracer is on the point (0, 0) and you press rocker wheel right, the tracer will move to (π, 0). The same trace options are available in Numeric view so that you can set up the table to show the same values that you traced in Plot view.
Advanced Graphing Plot Defn	You can edit the open sentence shown when you tap Defn (that is, edit it without having to go to Symbolic view). Tap Edit and a cursor appears at the end of the definition. Make your changes and tap OK (or press Enter). The graph will change to reflect the new definition. (The original definition in Symbolic view will also be changed.) To close the definition pane, tap the down-arrow menu button.
Advanced Graphing Plot Gallery	The Plot Gallery is a gallery of plots each of which you can save as a new instance of the Advanced Graphing app. While you are in Plot view, press the Menu key and select Visit Plot Gallery. Press the rocker wheel left/right to move from plot to plot. Note that the definitions that generated the current plot are shown at the bottom of the screen. To save a plot for your own exploration, tap Save and enter a name for a new customized app. You can then explore that new app as you would explore the built-in Advanced Graphing app.
Advanced Graphing Plot Setup	The Plot Setup view of the Advanced Graphing app enables you to control the appearance of Plot view, including the appearance of the cursor, whether or not the axes are drawn, etc. This view has two pages. Page 1 • X Rng: the horizontal graphing range • Y Rng: the vertical graphing range • X Tick: horizontal tick-mark spacing • Y Tick: vertical tick-mark spacing Page 2 • Axes: toggles axes on and off • Labels: toggles axis labels on and off • Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors

Help Topics Tree 13217	Help Text
Advanced Graphing Numeric View	Unlike many of the other HP apps, the Numeric view in the Advanced Graphing app does not give a table of values of the dependent variable as generated by the definitions selected in Symbolic view. Instead, both x and y are seen as independent and Numeric view shows whether or not each selected open sentence is satisfied for a set of (x, y) ordered pairs. If it is satisfied, True appears beside that combination; otherwise False appears. You can enter your own values in the X and Y columns. The app will automatically re-assess whether those values are satisfied by each open sentence that is represented in Numeric view. You can customize the table using the Trace menu key. Instead of showing x- and y-values and whether or not they satisfy the sentence, you can choose to fill the table with the coordinates of points of interest that appear in Plot view. For example, you can select to create a table of the current set of x-intercepts, inflection points, or intersection points visible in Plot view. Menu Buttons: • Zoom: opens the Zoom menu, with options to zoom in or out on the selected row of the table • More: opens a menu with editing options • Trace: provides various options for what to show in the table: • Edge values • Points of Interest (POI) • Defn: displays the definition of the selected open sentence The More menu The More menu contains the following options: • Select • Row: selects the row that contains the currently selected cell; the row can then be copied to paste elsewhere • Swap Ends: this option is available once a multi-cell selection has been made. Swaps the beginning and ending cells of the current selection. • Include Headers: the same as Select Row, except that the row headers are selected as well • Selection: toggles selection mode on and off
Advanced Graphing: Numeric View: Trace	Advanced Graphing: Numeric View: Trace Options The Trace options give you a way of seeing the numerical values of various of the graphs plotted in Plot view. You can choose to see the: • Edge values • various Points of Interest, such as: • X-intercepts • Y-intercepts • Horizontal extrema • Vertical extrema • Inflection points Note that the values shown in Numeric view while tracing is on represent features that fall within the plotting domain (that is, between Xmin and Xmax, and Ymin and Ymax). Thus if you have chosen to display the Y-intercepts of open sentence S3 and there are no Y-intercepts within the plotting domain, no values will appear in the S3 column. To return Numeric view to its standard presentation, tap Trace > Off.
Advanced Graphing Numeric Setup	The Numeric Setup view of the Advanced Graphing app enables you to configure what is displayed in Numeric view. Fields: • Num X Start: the starting value for the displayed X range • Num Y Start: the starting value for the displayed Y range • Num X Step: the increment between consecutive X-values • Num Y Step: the increment between consecutive Y-values • Num Type: Automatic = displayed values generated by Numeric Setup values; BuildYourOwn = you will specify the values for X and Y in Numeric view • Num X Zoom: the horizontal zoom factor for zooming operations • Num Y Zoom: the vertical zoom factor for zooming operations Menu Buttons: • Edit: modify the selected value • Choose: make a choice from a menu • Plot ->: Set the Numeric Setup values to match the current Plot view
Advanced Graphing App Variables	To display the variables relating to the Advanced Graphic app, press Vars, tap App and select Advanced Graphing. The Advanced Graphing app has the following variables: • Symbolic • Plot • Numeric • Modes
Advanced Graphing Symbolic Variables	The Advanced Graphing app symbolic variables are V1 through V9 and V0. These variables contain open sentences (in X, Y, both, or neither). Example: V1:='(X^2/3) –(Y^2/5)=1' → '(X^2/3) –(Y^2/5)=1'
Numeric Variables	Advanced Graphing Numeric Variables

Help Topics Tree	13217	Help Text
		This section lists the variables used in the Numeric view of the Advanced Graphing app.
NumXStart		NumXStart App Variable NumXStart sets the starting value for the X variable in Numeric view when Automatic is the Num Type. NumXStart := n, where n is a real number
NumYStart		NumYStart App Variable NumYStart sets the starting value for the Y variable in Numeric view when Automatic is the Num Type. NumYStart := n, where n is a real number
NumXStep		NumXStep App Variable NumXStep sets the step size (that is, increment value) for the X variable in Numeric view when Automatic is the Num Type. NumXStep := n, where n>0
NumYStep		NumYStep App Variable NumYStep sets the step size (that is, increment value) for the Y variable in Numeric view when Automatic is the Num Type. NumYStep := n, where n>0
NumType		NumType App Variable NumType determines how the independent variable values in Numeric view are generated: • Automatic: uses the NumStart and NumStep values to create the independent variable values. • Build Your Own: you enter the independent variable values one by one. NumType := 0 for Automatic (default) NumType := 1 for Build Your Own
NumXZoom		NumXZoom App Variable NumXZoom sets the default zoom factor for zooming in on the X values. NumXZoom := n, where n is a real number greater than 1
NumYZoom		NumYZoom App Variable NumYZoom sets the default zoom factor for zooming in on the Y values. NumYZoom := n, where n is a real number greater than 1
NumIndep		NumIndep App Variable NumIndep contains a list of values you have entered in Numeric view when you have chosen BuildYourOwn mode.
Graph 3D		Graph 3D app The Graph 3D app allows you to explore the graphical representations of functions that define Z in terms of X and Y. In Symbolic View, you can define up to ten functions, FZ1(X,Y) through FZ9(X,Y) and FZ0(X,Y). Highlight one of the function fields and begin entering an expression dependent on X and Y, or tap the Edit menu key to edit an existing expression. Once you have defined a function, you can view a table of its values or plot its graph. To launch the Graph 3D app, go to the Application Library and tap the Graph 3D app icon. You can also use the rocker wheel to select the Graph 3D app icon, then tap Start or press Enter to launch the app.
Graph 3D Symbolic View		The Graph 3D Symbolic view contains fields to define up to ten functions, each one defining Z in terms of X and Y. Press the Symb key to return to this view at any time. The menu buttons are: • Edit/Choose: opens an input line to edit the selected function definition or opens a choose box to make a selection • ✓: toggles the selected definition on or off for plotting and table-building • X: a typing aid for entering definitions in X • Y: a typing aid for entering definitions in Y • Show: displays the selected definition in full-screen mode with horizontal and vertical scrolling enabled • Eval: resolves references to other functions, such as FZ2(X,Y)=FZ1(X,Y)+1 Highlight one of the definition fields and begin entering an expression in X and Y, or tap Edit to open an edit line to edit an existing expression.
Graph 3D Plot View		Press the Plot key to enter the Graph 3D Plot view. This view displays the graphs of functions defined in Symbolic view. Tap the Menu soft key to open the Plot View menu. The menu buttons are: • Zoom: opens the Zoom menu, with options to zoom in or out • Trace: toggles the tracing cursor off and on • Go To: takes the cursor to the point with given X- and Y-values • FCN: opens the FCN menu with the following options: • Defn: displays the symbolic definition of the current graph; you can edit the expression directly in Plot view • Menu: toggles the Plot view menu off and on Tap to move the tracer to a location. Press + to zoom in on the current cursor location and press - to zoom out. The Zoom menu has many of the options found in the Zoom menu of the Function app. The options unique to the Graph 3D app are described here. All the zoom operations use the current zoom factors. These factors are set using Zoom Factors... in the Zoom menu. All the zooms described below use the current cursor location to center the zoom. Option Result In Zooms in on all three dimensions

Help Topics Tree 13217	Help Text
	Out Zooms out on all three dimensions Z In Zooms in on the Z-dimension Z Out Zooms out on the Z-dimension Square XY Makes the Y-range the same as the X-range Square Makes the Y-range and the Z-range the same as the X-range Decimal Makes the steps between values of both independent variables 0.1
Graph 3D Plot Setup	Press Shift Plot to enter the Graph 3D Plot setup. This view enables you to control the appearance of the graph and Plot view. The Setup has five pages. On the first page, the fields are: Field Description X Rng The minimum and maximum visible values of X in Plot view Y Rng The minimum and maximum visible values of Y in Plot view Z Rng The minimum and maximum visible values of Z in Plot view X Tick The spacing between tick marks on the X-axis Y Tick The spacing between tick marks on the Y-axis Z Tick The spacing between tick marks on the Z-axis The menu buttons on the first page are: • Edit: opens an edit line to edit the value of the selected field • Page 1/5 ▼: displays the second page of Plot Setup (tap on the right side of the menu key) On the second page, the fields are: Field Option Description Grid The number of steps used in computing X- and Y-values for each plot Surface The Surface field controls the coloring schema for the plots. The options are: Top/Bottom Uses one color for the top (positive Z, looking down) and another color for the bottom (negative Z, looking up) Checkerboard Uses a checkerboard pattern to color both the top and bottom of each plot. You can set the checker size as well. Elevation Color changes depending on the Z-value of each point plotted Slope Color changes depending on the gradient at each point plotted Key Axes Determines whether or not the orientation of the 3 axes is shown in the top left of Plot view. If checked, you can set the colors for the axes as well. The menu keys are: • Edit/Choose: opens an input line to edit the selected field or opens a choose box to make a selection • ✓: toggles the selection on or off On the third page of Plot Setup, the fields are: Field Option Description Box Sides None Do not color any of the box frame faces Rear Color each of the X-Y, Y-Z, and X-Z faces of the box frame that sit behind the plots Zmin Color the X-Y face at Zmin Box Frame None Do not draw the box frame around the plots Rear Draw only the 9 segments of the box frame that are behind the plots Front and Rear Draw the entire box frame Box Axes None Do not draw the axes Rear Draw the axes behind the plots Front and Rear Draw axes in front of and behind the plots Box Lines None Do not draw tick mark grid lines Rear Draw the tick mark grid lines that sit behind the plots Front and Rear Draw tick mark grid lines both behind and in front of the plots Box Dots None Do not draw tick mark grid dots Rear Draw the tick mark grid dots that sit behind the plots Front and Rear Draw tick mark grid dots both behind and in front of the plots Labels Check to label the axes The menu keys here are the same as on Page 2. The Plot Setup Page 4 options are: Field Option Description Cursor Standard Steady white cursor Inverting Blinking Blinking white cursor Box Scale 0.5-2 The scale factor for the box frame. The default is 1. Pose X Axis A real number The X-coordinate of the endpoint of the rotation vector Pose Y Axis A real number The Y-coordinate of the endpoint of the rotation vector Pose Z Axis A real number The Z-coordinate of the endpoint of the rotation vector Pose Turn A real number The angle of rotation (in radians) of the rotation vector The menu keys are the same as Pages 2 and 3. The last page of Plot Setup is the same as Page 3 of Plot Setup for the Function, Polar, and Parametric apps. Here you select an image to be used as a background in Plot view.

Help Topics Tree 13217	Help Text
Graph 3D Numeric View	Press the Num key to enter Numeric view. The Graph 3D Numeric View is designed to create and explore a table of X/Y/Z values, based on the function(s) defined in Symbolic View. Tap any row of the X or Y-columns, enter any real value, and tap OK. The table will reconfigure. You can also scroll the table by dragging or flicking with a finger. The menu buttons are: • Zoom: opens the Zoom menu. Note that in Numeric view, zooming changes the increment between consecutive X- and Y-values. Zooming in decreases the increment; zooming out increases the increment. The values in the row you zoom in or out on remain the same. • More: opens the More menu, identical to the one in the Function or Advanced Graphing app • Go To: jumps to specified values of the independent variables X and Y • Defn: displays the definition of the selected column The Zoom menu in Numeric view has many of the same options in the Zoom menu in Numeric view of the other graphing apps. Here are the zoom operations unique to the Graph 3D app. All the zoom operations use the current zoom factors. These factors are set in Numeric Setup. Option Result In Zooms in on the current row of the table in both the X- and Y-columns Out Zooms out on the current row in both the X- and Y-columns Y In Zooms in on the current row in just the Y-column Y Out Zooms out on the current row in just the Y-column Decimal Makes the steps between values of both independent variables 0.1 Integer Makes the steps between values of both independent variables 1 Trig Makes the step between values of both independent variables π/24 (radians) Un-Zoom Undoes the last zoom
Graph 3D Numeric Setup	The Graph 3D Numeric Setup extends the Function Numeric Setup to cover both independent variables X and Y instead of just X. The Graph 3D Numeric Setup lets you set start- and step-values for both X and Y, as well as set the zoom factors for both X and Y.
Graph 3D App Variables	To display the variables relating to the 3D Graphing app, press Vars, tap App and select 3D Graphing.
Symbolic Variables	Symbolic View Variables The symbolic variables in the Graph 3D app are FZ1(X, Y) through FZ9(X, Y) and FZ0(X, Y). Each of the variables contains an expression that defines Z in terms of X and Y.
Plot Variables	Plot View Variables The Graph 3D Plot variables include many of the Plot variables from the other graphing apps such as Function and Advanced Graphing. This section includes only those Plot variables unique to the Graph 3D app.
Page 1 Variables	Plot Setup Page 1 Variables The unique Plot view app variables on Page 1 of Plot Setup are: • Zmin: the minimum Z-value • Zmax: the maximum Z-value • Ztick: the tick mark spacing on the Z-axis
Page 2 Variables	Plot Setup Page 2 Variables The unique Plot view app variables on Page 2 of Plot Setup are: • GridX: the number of steps used from Xmin to Xmax for plotting points • GridY: the number of steps used from Ymin to Ymax for plotting points • Surface: contains a list that defines the coloring schema • {0}: Top/Bottom • {1, c, c}: Checkerboard; each square is c by c • {2}: Elevation • {3}: Slope • KeyAxes: contains 0 if Key Axes is unchecked and 1 if it is checked
Page 3 Variables	Plot Setup Page 3 Variables The unique Plot view app variables on Page 3 of Plot Setup are: • BoxSides: Which of the faces of the box frame are colored • 0: None - do not color any of the box frame faces • 1: Rear - color the three faces of the box frame that lie behind the plot(s) • 2: Zmin - color the face that lies at Zmin • BoxFrame: How the box frame is drawn • 0: None - do not display the box frame • 1: Rear - display the three faces of the box frame that lie behind the plot(s) • 2: Front and Rear - display all 6 faces of the box frame • BoxAxes: How the three axes are drawn • 0: None - do not display the axes • 1: Rear - display the three axes that lie behind the plot(s) • 2: Front and Rear - display axes in front of and behind the plot(s) • BoxLines: How grid lines are drawn on the box frame • 0: None - do not draw grid lines on the box frame • 1: Rear - draw grid lines on the three faces of the box frame that lie behind the plot(s) • 2: Front and Rear - draw grid lines on all of the box frame faces

Help Topics Tree				13217	Help Text
					• BoxDots: How grid dots are drawn on the box frame • 0: None - do not draw grid dots on the box frame • 1: Rear - draw grid dots on the three faces of the box frame that lie behind the plot(s) • 2: Front and Rear - draw grid dots on all of the box frame faces • BoxScale: A real number from 0.5 to 2 that determines the scale factor used to draw the box frame
				Page 4 Variables	Plot Setup Page 4 Variables The unique Plot view app variables on Page 4 of Plot Setup are: • BoxScale: a real number between 0.5 and 2 that determines the scale for the box frame • PoseXaxis: the X-coordinate of the endpoint of the rotation vector • PoseYaxis: the Y-coordinate of the endpoint of the rotation vector • PoseZaxis: the Z-coordinate of the endpoint of the rotation vector • PoseTurn: the angle of rotation (in radians) of the pose axis
				Page 5 Variables	Plot Setup Page 5 Variables There are no Plot view app variables unique to the Graph 3D app on Page 5 of Plot Setup. Only the three common Plot view app variables ImageName, ImageDisplay, and ImageOpacity are used here.
			Numeric Variables		Numeric View App Variables There are three Numeric view app variables unique to the Graph 3D app: • NumYStart: the starting value for Y in Numeric view • NumYStep: the step-value for Y in Numeric view • NumYZoom: the zoom-value for Y in Numeric view
	Geometry app				The Geometry app enables you to create, edit, 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 as you manipulate the construction. Tap Start or press Enter to launch the app. The app opens in Plot view.
		Geometry Plot View			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 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. Selecting an object usually involves two steps: tap to select a location or object and then press Enter. Often there will be multiple objects near where you tap the screen; a pop-up menu will appear to allow you to select the object(s) you want. Check the ones you want to select and tap OK. Otherwise, just press Enter to confirm your intention to create the point or select the object. When creating a point, you can tap on the screen and then use the rocker wheel to accurately position the point before pressing Enter. The bottom of the screen always contains help to guide you. You may see any or all of the following: • If a tool is active, you will see help at the bottom left (above the Cmds menu key) • If a tool is active, you will see the current command as it will be recorded in Symbolic view to the right of the help text • the current pointer coordinates are displayed at the bottom left • the objects recognized as being at the pointer coordinates are displayed at the bottom right; if you press Enter, these objects will appear in the pop-up menu Menu Button: • Cmds: The Commands menu has the following categories: • Zoom: displays a menu of zoom options for you to zoom in or out in Plot view • Point: displays a menu for creating various types of points • Line: displays a menu for creating various types of straight objects (segments, lines, tangents, etc.) • Polygon: displays a menu for creating various types of polygons • Curve: displays a menu for drawing conic sections and so on • Plot: displays a menu for plotting functions and so on • Transform: displays a menu offering various types of transformations (reflection, rotation, etc.) • Cartesian: displays a menu for displaying coordinates of points, Cartesian equations and so forth • Measure: displays a menu of various measurements, such as distance, slope, and so forth • Tests: displays a menu of various tests you can perform, such as is_collinear (tests whether or not a set of points is collinear) PROCESS Choose a tool from one of the menus listed above and follow the on-screen prompts. Press Enter after following each prompt, and then press Esc to de-activate the drawing tool. Each geometric object you create in the Plot view has its own entry or entries in the Symbolic view. In fact, you can create objects directly in the Symbolic view and they will appear in the Plot view. For most geometric objects, you can create them in either view. The Cmds menu appears in both views.

Help Topics Tree 13217	Help Text
	Calculations (measurements, tests, and so on) created from the Cartesian, Measure, or Tests menus appear at the top left of the display and are docked there; that is, they remain in place even if you scroll or zoom the display. You can edit the default labels for these objects. Tap and hold and an edit line will appear for you to enter your own descriptive label. Tap OK when you are done. You can select and move these objects, effectively undocking them. Once undocked, they move as you scroll and zoom in Plot view. To re-dock one of these objects, tap and hold. The edit line will re-appear; note the Dock menu key on the left. Tap Dock and the object will return to its docked position. Each calculation you create has its own entry in Numeric view, just as each geometric object you create has its own entry in Symbolic view. You can also create calculations and their labels in Numeric view. Once you have created objects in Plot view, you can select and move them, either one at a time or as a group. To select a group of objects, tap and hold, then drag to create a selection rectangle. Everything within the rectangle is selected. Drag to move the selected objects and press Esc or Enter when you are done. OPTIONS MENU When an object is selected, a new Options menu key appears. Tap this key to select among the options for the object selected, including color and fill options. The options vary, depending on the type of object selected, but include the following: • Choose Color: opens the color picker to select a color for the object • Fill: toggles between filling the object with its current color and no fill • Trace: toggles between tracing and not tracing for a selected point • Animate: starts and stops an animation based on the selected object • Point Style: opens a choose box to select a style for the selected point You can pinch (put two fingers on the screen and move them together or apart) to zoom out or in. You can drag to scroll the Plot view. Of course, + and - work to zoom in and out, respectively, on the pointer.
Keyboard Shortcuts	Geometry: Keyboard Shortcuts The keyboard shortcuts operate in Plot view only. The keyboard shortcuts are arranged by alphabetical letter; however, you do not have to press the Alpha key first - just press the key on which the corresponding letter appears. The keyboard shortcuts are: • A: toggles axes off and on (press Vars) • C: create a new circle (press the Template key) • I: finds an intersection of two objects (press TAN) • L: creates a new line (press x^2) • P: creates a new point (press EEX) • S: creates a new segment (press 9) • T: creates a new triangle (press ÷)
Geometry Plot Setup	The Plot Setup of the Geometry app enables you to control the appearance of Plot view, including the range of values shown, whether or not the axes are drawn, etc. FIELDS: • X Rng: the horizontal range • Y Rng: the vertical range • Pixel Size: the size of each square pixel in Cartesian units • Axes: show or hide the axes • Labels: show or hide the labels for each axis • Grid Dots: shows or hide grid dots • Grid Lines: show or hide grid lines • Scroll Text: auto-scroll the current command text in the help area The Plot view of the Geometry app always maintains a square window; that is, one in which the horizontal and vertical units remain the same. For this reason, you cannot modify the X-range and Y-range as freely as you can in the other apps. You can set the minimum X-value and minimum Y-value and then set the size of a pixel. Setting the Pixel size to 0.1 will change the pointer coordinates by 0.1 for each press of the rocker wheel (up, down, left, or right). As you make choices in these three fields, the other two are calculated and displayed automatically.
ScrollText	ScrollText Variable ScrollText controls the Scroll Text property in the Geometry Plot Setup screen. ScrollText := 0 Scroll Text check box cleared. (Default) ScrollText := 1 Scroll Text check box set. When ScrollText=1, long text in the menu in the Geometry plot view will scroll.
Geometry Symbolic View	Every object—whether a point, segment, line, polygon, or curve—is given a name, and its definition is displayed in Symbolic view. 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. In Symbolic view, you can modify existing objects and create new objects. Each object definition appears in its own line and has the following parts: • Check Box: the check box determines whether the object defined is shown or hidden in Plot view; check to show and uncheck to hide the object • Color Box: tap the color box to open the color picker; tap a color to select a color and close the picker • Name Field: the name field determines a unique name for the object; usually, the name will be set automatically. • Definition Field: the definition field is where the command is entered to define the object symbolically. Open the Cmds menu and choose a command You can flick up or down to kinetically scroll the display. Menu Buttons: PRIMARY VIEW

Help Topics Tree 13217	Help Text
	• Edit: modify a selected definition • ✓: select or deselect an object. Only selected objects appear in Plot view. • New: tap to create a new object. The buttons change (see below). • ↓: move the selected object down the list of objects • ↑: move the selected object up the list of objects • Delete: delete the selected object Menu Buttons: OBJECT-CREATION VIEW If you tap New in the primary Symbolic view (see above), the buttons at the bottom of the screen change to: • Cmds: the Commands menu has many of the same categories as it does in Plot view • Point: displays a menu for creating various types of points • Line: displays a menu for creating various types of straight objects (segments, lines, tangents, etc.) • Polygon: displays a menu for creating various types of polygons • Curve: displays a menu for drawing conic sections and so on • Plot: displays a menu for plotting functions and so on • Transform: displays a menu offering various types of transformations (reflection, rotation, etc.) There are other geometric commands at your disposal in the Catalog. Press the Toolbox key, tap the Ctlg menu key and scroll through the list. You can press any two-letter combination to jump to the approximate location of a command. See "Geometry Functions" help for more details on all geometric commands. Other Menu Buttons: OBJECT_CREATION VIEW • x: enters an x • y: enters a y • Cancel: cancels the operation • OK: finalizes the operation Note that the Geometry app uses lower-case x and y for defining plots and making calculations.
Geometry Numeric View	Numeric view enables you to create calculations and perform tests 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 numerical properties of that object. As in the Symbolic view, you can check an entry to make it visible in the Plot view of uncheck it to hide it in the Plot view. Menu Buttons: PRIMARY VIEW • Edit: modify a selected measurement or test • ✓: select or deselect a measurement or test. Selected measurements and tests appear in Plot view. • New: tap to create a new measurement or test. The buttons change (see below). • ↓: move the selected down the list • ↑: move the selected measurement or test up the list • Delete: delete the selected measurement or test Menu Buttons:OBJECT-CREATION VIEW If you tap New in the primary Numeric view (see above), the buttons at the bottom of the screen change to: • Cmds: displays a menu of all the geometry-specific measurements and tests • Vars: displays a menu of all the objects defined in Symbolic view • Cancel: cancels the operation • OK: finalizes the operation THE NUMERIC VIEW COMMANDS MENU • Cartesian: displays a menu for displaying coordinates of points, Cartesian equations and so forth • Measure: displays a menu of various measurements, such as distance, slope, and so forth • Tests: displays a menu of various tests you can perform, such as is_collinear (tests whether or not a set of points is collinear) In addition to creating tests and measurements, you can create calculations based on test results and measurements. Just tap New and enter the calculation as you would in the CAS or Home views. Each calculation you define in Numeric view appears on its own line and has the following parts: • Check Box: the check box determines whether the object defined is shown or hidden in Plot view; check to show and uncheck to hide the object • Color Box: tap the color box to open the color picker; tap a color to select a color and close the picker - the text label and numerical result will appear in Plot view drawn in this color • Label Field: the label field contains the default label for your calculation; tap Label to change the label. • Definition Field: the definition field is where the command is entered to define your calculation. Open the Cmds menu and choose a command; you can also enter functions directly from the keyboard. Note that the Geometry app uses lower-case x and y for defining plots and making calculations.
Geometry Variables	Apart from the modes and Plot view variables (which are common to all apps), the Geometry app has the following additional Plot view variables: • PixSize: determines the size of each square pixel in Cartesian units

Help Topics Tree	13217	Help Text
		• ScrollText: determines whether or not the current command text in Plot view scrolls manually or automatically. ScrollText := 0 manual scrolling (default) ScrollText := 1 automatic scrolling The other Geometry app vars are explained in HP Apps > Common App Variables > Common Plot View Variables. The Geometry app also lists as app vars each object in the Symbolic view.
Geometry Functions		This menu contains all the geometry-specific functions and commands.
Point		Geometry Point Functions This menu contains options commands for creating points, midpoints, and so forth.
point		Syntax: point(Real1, Real2) point(Expr1, Expr2) 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. Examples: point(3,4) creates a point whose coordinates are (3,4). This point may be selected and moved later. point(abscissa(GA), ordinate(GB)) 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.
element		Point On Syntax: element(Object, Real) or element(Real1..Real2) Creates a point on a geometric object whose abscissa is a given value or creates a real value on a given interval as a slider bar. The value you set using element(Real1..Real2) can be used as a coefficient in a function you subsequently define in Symbolic view and plot in Plot view. In addition, it can be used in a measurement or calculation in Numeric view. 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 on the slider name to open the slider bar and manipulate it. There is an Edit menu key that you can tap to define the slider more accurately, create animations, and so forth. Press Esc to close the slider bar at the new value or tap anywhere else on the screen.
midpoint		Syntax: midpoint(Segment) or midpoint(Point1, Point2) 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. Example: midpoint(0,6+6i) → point(3,3)

center		Syntax:
		center(Circle)
		Returns the center of a circle. The circle can be defined by the circle command or by name (e.g., GC).
		
		Examples:
		center(circle(x^2+y^2-x-y)) → point(1/2,1/2)
		center(circumcircle(0,1,1+i)) → point(1/2,1/2)

single_inter		Single Intersection Syntax: single_inter(Curve1, Curve2, [Point]) Returns the intersection of Curve1 and Curve2 that is closest to Point. In Plot view, this command will prompt for two curves. After that, a point will appear; move this point to the intersection desired and press Enter. You can move the point later to change intersections if you wish. Example: single_inter(line(y=x),circle(x^2+y^2=1), point(1,1)) → point((1+i)*√2/2)
Random Point		Creates a random point in Plot view. Activate this command and press Enter to create a random point in the Plot view. Keep pressing Enter to create more random points or press Esc to quit.
inter		Intersections Syntax: inter(Curve1, Curve2) Returns the intersections of two curves as a vector. Example: inter(8-x^2/6,x/2-1) → [[6, 2] [-9, -11/2]], indicating that there are two intersections-one at (6,2) and the other at (-9,-11/2).
Trace		Syntax: trace(point)

Help Topics Tree	13217	Help Text
		Trace is an option found in the Options menu of the Plot view of the Geometry app. It is a toggle for turning tracing off and on for the selected point. There is also an option to clear an existing trace from Plot view. Example: trace(GA)
	Clear Trace	Clear Trace erases the current trace in Plot view. It does not stop further tracing.
Line		Geometry Line Functions This menu contains all the geometrical functions specific to straight objects (segments, lines, etc.).
	segment	Syntax: segment(Point1, Point2) Draws a segment defined by its endpoints. Examples: segment(1+2*i, 4) draws the segment defined by the points with coordinates (1, 2) and (4, 0). segment(GA,GB) draws segment AB. draws segment AB.
	half_line	Ray Syntax: half_line(Point1, Point2) Given 2 points, draws a ray from the first point through the second point. Example: half_line(0,1+i) draws a ray starting at the origin and passing through the point at (1,1)
	line	Syntax: line(Point1, Point2) or line(a*x+b*y+c) or line(point1, slope=realm) Draws a line in the Plot view of the Geometry app or returns the equation of a line in CAS view. The arguments can be two points, a linear expression of the form a*x+b*y+c, or a point and a slope. Examples: line(2+i,3+2*i) 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	Syntax: parallel(Point, Line) Given a point and a line, returns the equation of the line through the point that is parallel to the given line. Examples: parallel(GA,GB) draws the line through point A that is parallel to line B. parallel(point(3,-2),line(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	Syntax: perpendicular(Point, Line) or perpendicular(Point1, Point2, Point3) 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. 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+2*i,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	Syntax: tangent(Curve, Point) Draws the tangent(s) to a given curve through a given point. The point does not have to be a point on the curve. Examples: tangent(plotfunc(x²), point(1,1)) draws the tangent to the graph y=x² through the point (1,1); that is, the line whose equation is y=2*x-1. tangent(plotfunc(x²), GA) draws the tangent to the graph of y=x² through point A. Point A can then be moved and the tangent will move with it. 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_line	Median Syntax: median_line(Point1, Point2, Point3) Given three points that define a triangle, draws the median of the triangle that passes through the first point and contains the midpoint of the segment defined by the other two points. In CAS view, returns the equation of the median line. Example:

Help Topics Tree	13217	Help Text
		median_line(0,8*i,4) draws the line whose equation is y=2x; that is, the line through the first vertex of the triangle at (0,0) and the point at (2,4), the midpoint of the segment with endpoints (0, 8) and (4, 0).
	altitude	Syntax: altitude(Point1, Point2, Point3) 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. Example: altitude(point(6,6), point(-2,3), point(5,1)) draws a line passing through point (6,6) that is perpendicular to the line passing through both points (-2,3) and (5,1).
	bisector	Syntax: bisector(Point1, Point2, Point3) 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. Examples: bisector(0,-4*i,4) bisector(0,1,i) bisector(GA,GB,GC) draws the bisector of∡BAC. bisector(0,-4i,4) draws the line given by y=–x
Polygon		Geometry Polygon Functions This menu contains all the geometrical functions specific to polygons.
	triangle	Syntax: triangle(Point1, Point2, Point3) Draws a triangle, given its three vertices. Example: triangle(GA,GB,GC) draws ΔABC
	isosceles_triangle	Isosceles Triangle Syntax: isosceles_triangle(Point1, Point2, Angle, [Var]) 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. 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	Right Triangle Syntax: right_triangle(Point1, Point2, Realk) 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. Example: right_triangle(GA,GB,1) draws an isosceles right triangle with its right angle at point A, and with both legs equal in length to segment AB.
	quadrilateral	Syntax: quadrilateral(Point1, Point2, Point3, Point4) Draws a quadrilateral from a set of four points. Example: quadrilateral(GA,GB,GC,GD) draws quadrilateral ABCD.
	square	Syntax: square(Point1, Point2) Draws a square, given two consecutive vertices as points. 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.
	parallelogram	Syntax: parallelogram(Point1, Point2, Point3) 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. 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	Syntax: rhombus(Point1, Point2, Angle) 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. 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	Syntax:

Help Topics Tree	13217	Help Text
		rectangle(Point1, Point2, Point3) or rectangle(Point1, Point2, Realk) 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. 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	Syntax: polygon(Point1, Point2, …, Pointn) Draws a polygon from a set of vertices. Example: polygon(GA,GB,GD) draws ΔABD
	isopolygon	Regular Polygon Syntax: isopolygon(Point1, Point2, Realn) 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. Examples: isopolygon(point(0,0,0),point(3,3,3),point(0,0,3),-5) isopolygon(GA,GB,6) draws a regular hexagon whose first two vertices are the points A and B.
Curve		Geometry Curve Functions This menu contains all the geometrical functions specific to curves.
	circle	Syntax: circle(Point1, Point2) or circle(Point1, Point2-Point1) or circle(equation) Draws a circle, given the endpoints of the diameter, or a center and radius, or an equation in x and y. 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 a clockwise arc. circle(GA,GB,0,π/2) draws a quarter-circle with diameter AB.
	circumcircle	Syntax: circumcircle(Point1, Point2, Point3) Draws the circumcircle of a triangle; that is, the circle circumscribed about a triangle. Example: circumcircle(GA,GB,GC) draws the circle circumscribed about ΔABC
	excircle	Syntax: excircle(Point1, Point2, Point3) Given three points that define a triangle, draws the excircles 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 whose 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	Syntax: incircle(Point1, Point2, Point3) Draws the incircle of a triangle, the circle tangent to all three sides of the triangle. Examples: incircle(0,4,4+4*i) incircle(GA,GB,GC) draws the incircle of ΔABC.
	ellipse	Syntax: ellipse(Point1, Point2, Point3) or ellipse(Point1, Point2, Realk) 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. 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.

Help Topics Tree	13217	Help Text
	hyperbola	Syntax: hyperbola(Point1, Point2, Point3) or hyperbola(Point1, Point2, Realk) 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. 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	Syntax: parabola(Point, Line) or parabola(Point, Realk) or parabola(Expr) 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 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. parabola(x-y²+y-2) draws the graph of the parabolic equation x=y²-y+2
	conic	Syntax: conic(Expr) Plots the graph of a conic section defined by an expression in x and y. Example: conic(x²+y²-81) draws a circle with center at (0,0) and radius of 9
	locus	Syntax: locus(Point,Element, [tstep=Value])) 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. The optional tstep statement can be used to control the default level of detail.
Plot		Geometry Plot Functions This menu contains a set of commands for plotting graphs in Plot view of the Geometry app.
	plotfunc	Plot Function Syntax: plotfunc(Expr) Used in the Geometry app Plot or Symbolic views, or in CAS view. Draws the plot of a function, given an expression in the independent variable x. Note the use of lowercase x. Example: plotfunc(3*sin(x)) draws the graph of y=3*sin(x).
	plotparam	Plot Parametric Syntax: plotparam(f(Var)+i*g(Var), Var= Interval, [tstep=Value]) Used in the Geometry app Symbolic view. 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. 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)
	plotpolar	Plot Polar Syntax: plotpolar(Expr,Var=Interval, [Step]) or plotpolar(Expr, Var, Min, Max, [Step]) Used in the Geometry app to draw a polar graph in Plot view. Examples: plotpolar(sin(2*x),x,0,π,tstep=0.1) plotpolar(f(x),x,a,b) draws the polar curve r=f(x) for x in [a,b]
	plotseq	Plot Sequence Syntax: plotseq(f(Var), Var={Start, Xmin, Xmax}, Integern) Used in the Geometry app Symbolic view. 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
	plotimplicit	Plot Implicit

Help Topics Tree	13217	Help Text
		Syntax: plotimplicit(Expr, [XIntrvl, YIntrvl]) Used in the Geometry app Plot or Symbolic views, or CAS view. 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. Examples: plotimplicit((x+5)^2+(y+4)^2-1,[x=-6..-4,y=-5..-3]) plotimplicit((x+5)^2+(y+4)^2-1) plots a circle, centered at the point (-5, -4), with a radius of 1
	plotinequation	Plot Inequation Syntax: plotinequation(Expr,[x=xrange,y=yrange],[xstep],[ystep]) Plots the graph of the solution of inequations with two variables. Example: plotinequation([x+y>3,x^2<y],[x,y],xstep=0.2,ystep=0.2)
	plotfield	Plot Slopefield Syntax: plotfield(Expr, VectorVar, [xstep=Val, ystep=Val, Option]) Used in the Geometry app or CAS view. 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. 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.
	plotode	Plot ODE Syntax: plotode(Expr, [Var1, Var2, ...], [Val1, Val2. ...], [tstep=Value]) Used in the Symbolic or Plot views of the Geometry app or in CAS view. 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. The optional tstep can be used to control the level of detail of the plot. Examples: 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. plotode(5*[-y,x],[t=0..1,x,y],[0,0.3,0.7],tstep=0.5,plan)
	plotlist	Plot List Syntax: plotlist(Matrix) Used in the Plot or Symbolic views of the Geometry app, or CAS view, this command plots a set of n points and connects them with segments. The points are defined by a m x 2 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
Transform		Geometry Transform Functions This menu contains all the geometrical functions specific to transformations.
	translation	Syntax: translation(Vector, Object) Translates a geometric object along a given vector. The vector is given as the difference of two points (head-tail). Examples: translation(0-i,GA) translates object A down one unit. translation(GB-GA,GC) translates object C along the vector AB.
	reflection	Syntax: reflection(Line, Object) or reflection(Point, Object) Reflects a geometric object over a line or through a point. The latter is sometimes referred to as a half-turn. 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	Syntax: rotate(Point, Angle, Object) Rotates a geometric object, about a given center point, through a given angle. Example: rotate(GA,angle(GB,GC,GD),GK) rotates the geometric object labeled K, about point A, through an angle equal to∡CBD.
	homothety	Dilation Syntax:

Help Topics Tree				13217	Help Text
					homothety(Point, Realk, Object) Dilates a geometric object, with respect to a center point, by a scale factor. Examples: 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. homothety(point(0,0),1/3,point(9,9)) creates an image point at (3,3).
				similarity	Syntax: similarity(Point, Realk, Angle, Object) Dilates and rotates a geometric object about the same center point. 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	Syntax: projection(Curve, Point) Draws the orthogonal projection of a point onto a curve. Example: projection(circle($x^2+y^2=4$),point(6,6)) creates a point on the circle at ($\sqrt{2}$,$\sqrt{2}$)
				inversion	Syntax: inversion(Point1, Realk, Point2) Draws the inversion of a point, with respect to another point, by a scale factor. 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	Syntax: reciprocation(Circle, [Obj1, Obj2,...Objn]) 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. 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)]
			Cartesian		Geometry Cartesian Functions This menu contains commands that are essentially Cartesian in nature. These include the coordinates of points, the equations of lines and curves, and slider bars among others.
				abscissa	Syntax: abscissa(Point) or abscissa(Vector) Returns the x coordinate of a point or the x length of a vector. Example: abscissa(GA) the x-coordinate of the point A.
				affix	Syntax: affix(Point) or affix(Vector) Returns the coordinates of a point or both the x- and y-lengths of a vector as a complex number. Examples: affix(point(3,2)) returns 3+2*i If GA is a point at (1, -2), then affix(GA) returns 1-2*i.
				coordinates	Syntax: coordinates(Point) or coordinates(Vector) Given a point, returns a vector with its 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. Example: coordinates(point(1+2*i)) → [1,2]
				ordinate	Syntax: ordinate(Point) or ordinate(Vector) Returns the ordinate of a point or the y-length of a vector. Example: ordinate(point(1+2*i)) → 2
				polar_coordinates	Polar Coordinates Syntax: polar_coordinates(Point) Returns a vector containing the polar coordinates of a point. Example: polar_coordinates(point(1+2*i)) → $\sqrt{5}$ atan(2)
				equation	Equation of Syntax:

Help Topics Tree	13217	Help Text
		equation(Curve) or equation(Point) Returns the Cartesian equation of a curve in x and y, or the Cartesian coordinates of a point. Examples: equation(line(1-i,i)) → y=-2*x+1 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 $x^2 + y^2 = 1$.
	parameq	Parametric Syntax: parameq(Obj) Returns a parametric equation for a geometric object. The parametric equation is true for all complex numbers that represent points on the object. Examples: parameq(circle(0,1)) → e^(i*t) parameq(line(i,1-i))
Measure		Geometry Measure Functions This menu contains all the geometrical measurement specific functions
	distance	Syntax: distance(Point1, Point2) or distance(Point, Curve) Returns the distance between two points or between a point and a curve. Examples: distance(1+i,3+3i) → 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 2√3.
	radius	Syntax: radius(Circle) Returns the radius of a circle. Examples: radius(circle(-1,1-i)) → √2 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	Syntax: perimeter(Polygon) or perimeter(Circle) Returns the perimeter of a polygon or the circumference of a circle. Examples: perimeter(0,1,i) → √2+2 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 2π. 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.
	area	Syntax: area(Circle) or area(Polygon) or area(Function, Value1, Value2) Returns the area of a circle or polygon. Can also return the area under a function between two x-values. Examples: If GA is defined to be the unit circle, then area(GA) returns π. If GA is defined to be plotfunc($4-x^2/4$), then area(GA,-4,4) returns 64/3 or 21.333… In CAS view, area($4-x^2/4$,x=-4..4) returns 64/3 as well.
	angle	Syntax: angle(Vertex, Point2, Point3) Returns the measure of a directed angle. The first point is taken as the vertex of the angle and the next two points in order give the measure and orientation. Examples: angle(i,1,1+i,"b") returns the measure of∡BAC
	slope	Syntax: slope(Line) or slope(Point1, Point2) Given a line or two points that define a line, returns the slope of the line. Example: slope(line(1,2*i)) → -2
	arcLen	Arc Length Syntax: arcLen(Expr, Real1, Real2) 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. Examples: arcLen(x^2,x,-2,2) → 9.29…

Help Topics Tree					13217	Help Text
						arcLen({sin(t),cos(t)},t,0,π/2) → 1.57…
			Test			Geometry Test Functions This menu contains all the geometrical test specific functions
				is_collinear		is_collinear Function Syntax: is_collinear(Point1, Point2, ..., Pointn) 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. Example: is_collinear(point(0,0), point(5,0), point(6,1)) → 0
				is_concyclic		is_concyclic Function Syntax: is_concyclic(Point1, Point2, Point3, Point4)) Takes a set of 4 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. Example: is_concyclic(point(-4,-2), point(-4,2), point(4,-2), point(4,2)) → 1
				is_element		is_element Function Syntax: is_element(Point, Object) Tests if a point is on a geometric object. Returns a number (1 to number of sides) representing the segment containing the point and 0 otherwise. Examples: is_element(point((√(2)/2),(√(2)/2)),circle(0,1)) → 1 is_element(point(0,0.5),square(0,1)) → 4
				is_parallel		is_parallel Function Syntax: is_parallel(Line1, Line2) Tests whether or not two lines are parallel. Returns 1 if they are and 0 otherwise. Example: is_parallel(line(2x+3y=7),line(2x+3y=9) → 1
				is_perpendicular		is_perpendicular Function Syntax: is_perpendicular(Line1, Line2) Similar to is_orthogonal. Tests whether or not two lines are perpendicular. Returns 1 if they are or 0 if they are not. Example: is_perpendicular(line(y=x),line(y=-x)) → 1
				is_isosceles		is_isosceles Function Syntax: is_isosceles(Point1, Point2, Point3) 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. Examples: is_isosceles(point(0,0), point(4,0), point(2,4)) → 3 is_isosceles(triangle(0,i,1+i)) → 2
				is_equilateral		is_equilateral Function Syntax: is_equilateral(Point1, Point2, Point3) Takes three points and tests whether or not they are vertices of a single equilateral triangle. Returns 1 if they are and 0 otherwise. Example: is_equilateral(triangle(0,2,1+i*√3)) → 1
				is_parallelogram		is_parallelogram Function Syntax: is_parallelogram(Point1, Point2, Point3, Point4) 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. Example: is_parallelogram(point(0,0), point(2,4), point(0,8), point(-2,4)) → 2
				is_conjugate		is_conjugate Function Syntax: is_conjugate(Circle, Point1, Point2, [Point3]) or is_conjugate(Line1, Line2, Line3, [Line4]) Tests whether or not two points or two lines are conjugates for the given circle. Returns 1 if they are and 0 otherwise.
			Zoom			Geometry Zoom Functions This menu contains options for zooming in Plot view of the Geometry app.

Help Topics Tree 13217	Help Text
Geometry Symbolic Functions	There are two symbolic functions in the Geometry app. These were designed to allow the user to edit and delete symbolic definitions of geometric objects. These commands are Instruction and DelInstruction.
Instruction	The Instruction command provides access to the list of symbolic definitions in the Symbolic view of the Geometry App. Each Symbolic view definition is numbered, with 1 being the first definition. Instruction(n) returns the value of instruction n (equivalent to typing its name). Instruction(n,0) returns a textual representation of instruction n. Instruction(n,1) returns a list containing all the information for instruction n: {name, definition, value, color, visible(0/1), plotDetail (0-7), filled (0/1), traced (0/1), legend visible (0/1), {} or {font, dpi, x, y} text position for sliders and measures, animation data: {} or {animation type (0-3), steps per sec (0-15), pause (0-15), start (real), stop(real), steps(real)}} Instruction(n, k) returns the kth-1 element from the meta data list, where k is between 2 and 12 inclusive (see above). Instruction(n):= Sets the value of instruction n from a string. Instruction(n,1):= sets all instruction data using the same list format as described above. Instruction(n, k):= sets meta data object k-1 to a specific value, where k is between 2 and 12 inclusive (see above). if n=0 or if n is greater than the number of current instructions, adds an instruction at the end of the instruction list. if n<0, inserts an instruction at position -n. n can also be the variable name, either using 'name' or "name"
DelInstruction	Delete Instruction The DelInstruction command allows you to delete one or more symbolic definitions of geometric objects in Symbolic view. Each Symbolic view definition is numbered, with 1 being he first definition. DelInstruction erases ALL instructions. DelInstruction(n) erases instruction n. DelInstruction(a,b) erases instructions a to b.
Spreadsheet app	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 that content. Tap Start or press Enter to launch the app. The app opens in Numeric view. The menu keys are: • Format: opens the Format menu • Go To: jumps to a specific cell • Select: toggles selection mode on and off • Go: determines where the selection goes after Enter is pressed. Toggles between the options of right and down. The following gestures are supported in Numeric view: • tap to select a cell • tap and hold, the drag to select a rectangular array of cells • drag to scroll the window • flick to initiate kinetic scrolling of the window in the desired direction • open/close pinch vertically to increase the height of the row that contains the currently selected cell • open/close pinch horizontally to increase/decrease the width of the column that contains the currently selected cell
Navigation and Selection	You can move about a spreadsheet by using the rocker wheel, tapping or dragging, 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 rocker wheel to move to the diagonally opposite cell. Tapping on Sel or another cell deselects the selection.
Cell Referencing	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 CR (where C is the column letter and R the row number). Thus B7 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. These must be in lowercase. Thus cell B1 can be referred to as B1,b1,B1 or b1 whereas M1 can only be referred to as m1 or m1. (G, L, M, and Z are names reserved for graphic objects, lists, matrices, and complex numbers.) Row, Col, and Cell are also variables that can be used for referencing. See Spreadsheet Variables for more information.
Naming Cells	Cells, rows, and columns can be named. The name can then be used in a formula. A named cell is displayed with a blue border. To name a cell, row, or column:

Help Topics Tree 13217	Help Text
	1. Select the cell, row, or column. Method 1: 2. Enter a name 3. Tap Name in the menu Method 2: 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*11.
Entering Content	Syntax: Row Cell A cell can contain any valid calculator object: a real number (3.14), a complex number (a + b*i), 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 set content into 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 =. For example, suppose that you want to add cell A1 (which contains 7) to cell B2 (which contains 12). Entering A1+ B2 in, say, A4 yields 19, 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. When entering a formula using =, you have the option to have the result evaluated numerically or using the CAS to generate exact or symbolic results. Once you enter =, you will notice that there is a CAS menu button. If you activate this button by tapping on it (a dot will indicate it is active), then when you press the Enter key, the results will be evaluated using the CAS. When you select a cell, row, or column that has a CAS-active formula in it, you will see "CAS" in red letters above the equal sign. A single formula can be used to generate a value for 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 will display the sine of the cell’s row number. A similar process enables you to populate every cell in a row with the same formula. Note that no content was placed in these cells. For example, set A1 to 1 and A to =Cell(Row-1,1)+Cell(Row-2,1) to display a Fibonacci sequence in column A. 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). for example: =COMB(Row, Cell) will create a sheet with Pascal's triangle
Importing from a Statistics App	You can import data from the Statistics 1Var and Statistics 2Var apps (and from any app customized from a statistics app). The procedure below imports dataset D1 from the Statistics 1Var App. 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. On the other hand, if you start the formula in Step 2 with =, then the entire list D1 of the Statistics 2Var app will be pasted into the selected cell.
Sort Cells	To sort a group of cells: 1. Select cells to be sorted. 2. Tap Sort. A menu appears giving you the option of choosing what column to sort by. 3. Choose the column. A sub-menu appears given you two sort options: ascending (↑) and descending (↓). 4. Choose a sort option. The values in the selected cells are sorted accordingly.
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). 3. The Paste choose box opens with your selections displayed as a list and selected. Tap on the selection (or press rocker wheel right) to see a list of paste options. You can choose to paste either the value, formula, format, both value and format, or both formula and format. 4. Tap on a paste option or use the rocker wheel to select an option and press Enter (or tap OK). The paste operation is completed.
External References	You can refer to data in a spreadsheet from outside the Spreadsheet app by using the reference CR. For example, in Home view you can refer to cell A6 in the built-in spreadsheet by entering A6. Thus the formula 6*A6 would multiply whatever value is currently in cell A6 by 6. If Spreadsheet is not the current app, you can still use Cell content by fully qualifying the cell name using AppName.CellName. For example, Spreadsheet.A6 or Savings.A6 if you have saved your spreadsheet as 'Savings'. An external reference can also be to a named cell, as in 5*Savings.TOTAL.

Help Topics Tree	13217	Help Text
		In the same way, you can also enter references to spreadsheet cells in the CAS. Note that a reference to a spreadsheet name is case-sensitive.
Expressions in Spreadsheet		Any expression valid in Home or CAS can be used in the Spreadsheet app. This includes use of Home, CAS, user and App variables.
Name a Cell, Row or Column		Enter a name for the cell, row, or column and tap OK.
Go To		You can go to any cell in the spreadsheet directly by tapping Go To. Enter the definition of the cell you want to go to then tap OK. The definition can be a cell reference (such as B7) or the given name of a cell. The definition can also be a selection such as A1:F7 in which case all the cells are selected.
Spreadsheet Choose		Syntax: =CHOOSE(Variable_or_number, "title", list) The CHOOSE function has a special application in the spreadsheet. Placing a CHOOSE function as a cell formula will do 4 things: - The cell value (result of the evaluation of CHOOSE) will be the selected item in the choose box - The cell display will show a dropdown menu - Tapping on the cell will open the choose and let you change the selection - When the user changes the selected item in the choose through the UI, the title_Changed function of the app program will be called. Note that the 'list' can be hard-coded {"up", "down"}, calculated MAKELIST(X^2, X, 1, 10), or extracted from the spreadsheet (A1:A5). The first parameter of the choose can be either a variable name or a cell reference. Examples: If the formula in B1 is =CHOOSE(1, "Direct", {"Up","Down"}) Creates a choose box that let the user choose Up or Down If the formula in B2 is =CHOOSE(C2, "VarBased", {"Up","Down"}) Creates a choose box similar to the previous example, except that the selected item is stored in C2 If the formula in B3 is =CHOOSE(1, "Calculated", MAKELIST(X^2, X, 1, 10)) Allows the user to choose between the first 10 square integers And finally, assuming that the spreadsheet has data in cells A1 to A5 If the formula in B4 is =CHOOSE(C3, "SheetData", A1:A3) Allows the user to choose between the first 5 squares in Column A
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 will depend on what is selected. The full list of options is: • Name: displays an input form for you to give a name to whatever is selected • Number Format: Auto, Standard, Fixed, Scientific, Engineering, Floating or Rounded • 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 ↔: horizontal alignment—Auto, Left, Center, Right • Align ↕: vertical alignment—Auto, Top, Center, Bottom • Column ↔: displays an input form for you to specify the required width of the selected columns; only available if you have selected the entire spreadsheet or one or more entire columns. • Row ↕: displays an input form for you to specify the required height of the selected rows; only available if you have selected the entire spreadsheet or one or more entire rows. • show “ “: show quote marks around strings in the body of the spreadsheet—Auto, Yes, No • Textbook: display formulas in textbook format—Auto, Yes, No • Caching: turn this option on to speed up calculations in spreadsheets with many formulas; only available if you have selected the entire spreadsheet
Spreadsheet Variables		Syntax: ColWidth RowHeight Row Col Cell Apart from the modes variables (which are common to all apps), the Spreadsheet app has the following Numeric variables: • ColWidth • RowHeight • Row • Col • Cell
	ColWidth	ColWidth Variable ColWidth(Integer) allows you to set and get the width of columns. Integer1 ► ColWidth(Integer2) sets the width of column Integer2 (A=1, B=2, etc.) to Integer1 pixels. Here, both Integer1 and Integer2 are positive. If Integer2 is not specified, sets the default width for columns in the spreadsheet to Integer1 pixels.

Help Topics Tree 13217	Help Text
	ColWidth(Integer) returns the width of the column specified by Integer (A=1, B=2, etc.). You can also set the column width from the Format menu.
RowHeight	RowHeight Variable RowHeight(Integer) allows you to set and get the height of rows. Integer1 ► RowHeight(Integer2) sets the height of row Integer2 to Integer1 pixels. If Integer2 is not specified, sets the default height for rows in the spreadsheet to Integer1 pixels. RowHeight(Integer) returns the height of the row specified by Integer. You can also set the column width from the Format menu.
Row	Row Variable Row is a variable that indicates currently calculated cell row number. This is mostly used when creating generic expression that need to work anywhere in the spreadsheet or for columns or full spreadsheet expressions. Example steps: 1. Select column A 2. Type =expand((x+1)^Row) 3. Tap the CAS menu button 4. Tap the OK menu button Column A will now contain the expansions of $(x+1)^1$, $(x+1)^2$, $(x+1)^3$, etc.
Col	Col Variable Col is a variable that indicates currently calculated cell column number (A=1, B=2, etc.). This is mostly used when creating generic expression that need to work anywhere in the spreadsheet or for rows or full spreadsheet expressions. Example steps: 1. Tap on the upper left corner of the spreadsheet (where the HP logo is) to select the entire spreadsheet 2. Type =COMB(Row-1,Col-1) 3. Tap the OK menu button The spreadsheet will now be filled with Pascal's triangle.
Cell References and Cell	In most cases, you will be referencing cells directly by their Row-Column (RC) names as in A1 or D6 just like in your usual spreadsheet. Only advanced formulas creators or users that need access to spreadsheet data from outside of the spreadsheet numerical view will need to understand the full complexity of cell references. Examples: A1:= 100 stores the value 100 in cell A1. A1:= A2+A3 stores value of A2+A3 in A1 using the current values of A2 and A3. A1:= 'A2+A3' sets A1 to the formula A2+A3. Syntax: Cell(RowNumber, ColNumber, [n]) For slightly more complex formulas, Cell(r, c) where r is a row number and c a column number (A=1, B=2 …) is equivalent to ColNameRowNumber. For example, Cell(1,1) is equivalent to A1. Valid references are: [$]R[$]C[(n)] or [$]CellName[(n)] or [$]R1[$]C1:[$]R2[$]C2:[(n)] or [$]CellName1:[$]CellName2[(n)] or a mix of both name and RC syntaxes [$]R:[$]R[(n)] or RowName[(n)] or [$]C:[$]C[(n)] or ColName[(n)] Where R(1/2) is a Row name or number and C(1/2) is a Column name or number gives full access to a cell or selection definition or to the cell's attributes. For the Cell access method, note that Cell(0, Col) gives access to the specified column, Cell(Row, 0) gives access to the specified row and Cell(0,0) gives access to the sheet definition itself. GETTING THE CONTENT OF CELLS AND SELECTIONS: If n is not specified and the reference is not used as a Sto destination, the value of the cell/selection is returned. If the reference is to a single cell, the cell value/content/attributes will be returned. If the reference is to a single row or column, a list of value/content/attributes will be returned, one for each cell. If the reference is to a selection, a list of list of value/content/attributes will be retuned, one for each column. Note: a cell with no associated value is considered as having a value of 0. If n is specified, the table bellow indicates what attribute of the cell will be returned. MODIFYING THE CONTENT OF CELLS AND SELECTIONS: If n is not specified and the reference is used as a Sto destination, the expression associated with the cell/selection is modified. If a single input is used as the source for more than one destination, the input is duplicated for all destinations. If the input is an expression, relative cell references are updated as needed. For example: A1:=1; A2:A10:='A1+1';

Help Topics Tree 13217	Help Text
	If n is specified, the table below indicates what attribute of the cell will be modified. CELL ATTRIBUTES (n) -1: all attributes. If the cell has nothing defined, returns -1, else return a list of 11 objects. 0: value (read only, you can not set the cell value) 1: formula 2: name 3: number format: Standard 0, Fixed 1, Scientific 2, Engineering 3, Floating 4, Rounded 5, unspecified –1 4: number of decimal places: 1 to 11, or unspecified = –1 5: font: 0 to 6, unspecified = –1 6: foreground color: contents color (color, or -1 if unspecified) 7: background color: cell fill color (color, or -1 if unspecified) 8: horizontal alignment: Left = 0, Center = 1, Right = 2 , 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 Note: As a general rule, -1 means unspecified or auto.
Cursor	Cursor Variable Syntax: Cursor Cursor(n) Cursor returns a list representing the cursor position and selection stop location. Cursor(n) returns the nth object of the list that Cursor would have returned Cursor:= {row, col, [selectionRow, selectionCol]} Sets the cursor position If a selection is specified when setting Cursor, then the selection is activated else it is deactivated Example: Cursor → {row, col, [selectionRow, SelectionCol]}
CellName	CellName Variable Syntax: CellName CellName("name") CellName returns the list of all the named cells in the spreadsheet CellName("name") returns a list with the row and column of the named cell if it exists. Else returns 0. Examples: CellName → { "name1", "name2"...} CellName("name") → { row, column }
Spreadsheet App Functions	The Spreadsheet app has a set of spreadsheet specific functions. These functions can be categorized in 2 groups. 1. Functions that need to pay special attention to cell that are empty (but still return a 0, for example AVERAGE) 2. Functions designed to speed up calculations by returning more than 1 result at once.
SUM	SUM Function Syntax: SUM([Input]) Calculates the sum of a range of numbers. 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	AVERAGE Function Syntax: AVERAGE([Input]) Calculates the arithmetic mean of a range of numbers. 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 AVERAGE(B7:C23). An error is returned if a cell in the specified range contains a non-numeric object. Empty cells are ignored.
AMORT	AMORT Function Syntax: AMORT(Range, n, i, pv, pmt[, ppyr=12, cpyr=ppyr, Grouping=ppyr, Beg=false, Fix=current], "Configuration"]) Range is the cell range where the results are placed. If only one cell is specified, then the range is automatically calculated. Configuration is a string that defines if a header row needs to be created (starts with H) and what result to place in which column. h: This column contains the row headers S: This column contains the start of the period E: This column contains the end of the period P: This column contains the Principal paid this period

Help Topics Tree	13217	Help Text
		B: This column contains the balance at the end of the period
		I-: his column contains the interest paid this period
		For example:
		"H h E P" means Put headers and compute End and Principal only.
		n, i, pv, pmt are the number of periods for the loan, the interest rate, the present value, and the per period payment.
		ppyr and cpyr are the number of payments per year and the number of compounding periods per year.
		Grouping is the number of periods that need to be grouped together in the amortization table.
		beg is 1 when payment is at the beginning of each period; otherwise it is 0.
		fix is the number of decimal places displayed in the result of calculations.
STAT1		STAT1 Function
		Syntax:
		STAT1(Input_Range, [Mode], ["Configuration"])
		The STAT1 function provides a range of one-variable statistics.
		Input_Range is the data source (such as A1:D8).
		Mode: Defines how to treat the input. The valid values are:
		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 one data set. If two columns are selected, the mode defaults to frequency.
		Configuration: Indicates which values you want to place in which row and if you want row or columns headers. Place the symbol for each value in the order that you want to see the values appear in the spreadsheet.
		The valid values for Configuration are:
		H (Place column headers)
		h (Place 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" the first column will contain row headers, the first row will be the number of items in the input data and the second will be the standard deviation.
		Examples:
		STAT1(A25:A37)
		STAT1(A25:A37,"h n σ")
STAT2		STAT2 Function
		Syntax:
		STAT2(Input_Range, [Mode], ["Configuration"])
		The STAT2 function provides a range of two-variable statistics.
		Input_Range is the data source (such as A1:D8).
		Mode: Defines how to treat the input. The valid values are:
		1 = Single data. Each column pair is treated as a paired dataset.
		2 = Frequency data. Columns are used in groups of 3 and the third column is treated as the frequency of appearance of the paired columns.
		If more than two columns are specified, each additional pair is treated as a different input data set. If only one pair is selected, it is treated as one data set. If three columns are selected, the mode defaults to frequency.
		Configuration: Indicates which values you want to place in which row and if you want row or columns headers. Place the symbol for each value in the order that you want to see the values appear in the spreadsheet.
		The valid values for Configuration are:
		H (Place column headers)
		h (Place row headers)
		MeanX
		Σx
		Σx^2
		sx

Help Topics Tree 13217	Help Text
	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" the first column will contain row headers, the first row will be the number of items in the input data, the second will be the x mean, and the third will be the y standard deviation. Examples: STAT2(A25:B37) STAT2(A25:B37,"h n σy")
REGRS	REGRS Function Syntax: REGRS(Input_range, [model], ["configuration"]) Attempts to fit the input data to a function specified by model (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 modes 1-6) - int (intercept, only valid for modes 1-6) - cor (correlation, only valid for modes 1-6) - cd (Coefficient of determination, only valid for modes 1-6, 8-10) - sCov (Sample covariance, only valid for modes 1-6) - pCov (Population covariance, only valid for modes 1-6) - L (L parameter for mode 7) - a (a parameter for modes 7-11) - b (b parameter for modes 7-11) - c (c parameter for modes 8-11) - d (d parameter for modes 8, 10-11) - e (e parameter for mode 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)
PredY	PredY Function Syntax: PredY(mode, x, parameters) Returns the predicted y value for a given x value. mode governs the regression model used: 1: y= sl*x+int 2: y= sl*ln(x)+int 3: y= int*exp(sl*x)

Help Topics Tree 13217	Help Text
	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= cx²+bx+a
	10: y= dx³+cx²+bx+a
	11: y= ex⁴+dx³+cx²+bx+a
	parameters is either one argument (a list of the coefficients of the regression line), or the n coefficients one after another.
PredX	PredX Function
	Syntax:
	PredX(mode, y, parameters)
	Returns the predicted x value for a given y value.
	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= cx²+bx+a
	10: y= dx³+cx²+bx+a
	11: y= ex+dx³+cx²+bx+a
	parameters is either one argument (a list of the coefficients of the regression line), or the n coefficients one after another.
HypZ1mean	HypZ1mean Function
	Syntax:
	HypZ1mean(input_list, ["configuration"])
	HypZ1mean(SampMean, SampSize, NullPopMean, PopStdDev, SigLevel, Mode, ["configuration"])
	The one-sample Z-test for a mean.
	input_list:
	A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values.
	Input Parameters:
	SampMean
	SampSize
	NullPopMean
	PopStdDev
	SigLevel
	Mode:
	Specifies how to calculate the statistic:
	1 = Less than
	2 = Greater than
	3 = Not equal
	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).
	h : Create header cells
	acc : 0 or 1 to reject or fail to reject the null hypothesis
	tZ : Test Z-value
	tM: Test Mean
	prob : lower-tail probability
	cZ : Critical Z
	cx1 : Critical □1
	cx2 : Critical □2
	std : Standard deviation
HypZ2mean	HypZ2mean Function
	Syntax:
	HypZ2mean(input_list, ["configuration"])
	HypZ2mean(SampMean, SampMean2, SampSize, SampSize2, PopStdDev, PopStdDev2, SigLevel, Mode, ["configuration"])
	The two-sample Z-test for the difference of two means.
	input_list:
	A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values.
	Input Parameters:
	SampMean

Help Topics Tree	13217	Help Text
		SampMean2 SampSize SampSize2 PopStdDev PopStdDev2 SigLevel Mode: Specifies how to calculate the statistics: 1 = Less than 2 = Greater than 3 = Not equal 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) . h : Create header cells acc : 0 or 1 to reject or fail to reject the null hypothesis tZ : Test Z tM : Test Mean prob : Probability cZ : Critical Z cx1 : Critical �1 cx2 : Critical �2
HypZ1prop		HypZ1prop Function Syntax: HypZ1prop(input_list, ["configuration"]) HypZ1prop(SuccCount, SampSize, NullPopProp, SigLevel, Mode, ["configuration"]) The one-sample Z-test for a proportion. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SuccCount SampSize NullPopProp SigLevel Mode: Specifies how to calculate the statistics: 1 = Less than 2 = Greater than 3 = Not equal 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). h : Create header cells acc : 0 or 1 to reject or fail to reject the null hypothesis tZ : Test Z-value tP : Test proportion of successes prob : Lower-tail probability cZ : Critical Z-value cp1 : Lower critical proportion of successes associated with the critical Z-value cp2 : Upper critical proportion of successes associated with the critical Z-value std : Standard deviation
HypZ2prop		HypZ2prop Function Syntax: HypZ2prop(Input_List, ["configuration"]) HypZ2prop(SuccCount1, SuccCount2, SampSize1, SampSize2, SigLevel, Mode, ["configuration"]) The two-sample Z-test for comparing two proportions. Input_List: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SuccCount1 SuccCount2 SampSize1 SampSize2 Mode: Specifies how to calculate the statistics: 1 = Less than 2 = Greater than 3 = Not equal

Help Topics Tree				13217	Help Text
					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). h : Create header cells acc : 0 or 1 to reject or fail to reject the null hypothesis tZ : Test Z-value tdP : Test proportion of successes prob : Lower-tail probability cZ : Critical Z-value cp1 : Lower critical proportion of successes associated with the critical Z-value cp2 : Upper critical proportion of successes associated with the critical Z-value std : Standard deviation
				HypT1mean	HypT1mean Function Syntax: HypT1mean(input_list, ["configuration"]) HypT1mean(SampMean, SampStdDev, SampSize, NullPopMean, SigLevel, Mode, ["configuration"]) The one-sample T-test for a mean. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SampMean SampStdDev SampSize NullPopMean SigLevel Mode: Specifies how to calculate the statistics: 1 = Less than 2 = Greater than 3 = Not equal 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). h : Create header cells acc : 0 or 1 to reject or fail to reject the null hypothesis tT : Test T-value tM : Test mean prob : Lower-tail probability df : Degrees of freedom cT : Critical T-value cX : Critical value of the mean associated with the critical T-value
				HypT2mean	HypT2mean Function Syntax: HypT2mean(input_list, ["configuration"]) HypT2mean(SampMean1, SampMean2, SampStdDev1, SampStdDev2, SampSize1, SampSize2, pooled, SigLevel, Mode, ["configuration"]) The two-sample T-test for the difference of two means. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SampMean1 SampMean2 SampStdDev1 SampStdDev2 SampSize1 SampSize2 pooled: 0 (not pooled) or 1 (pooled) SigLevel Mode: Specifies how to calculate the statistics: 1 = Less than 2 = Greater than 3 = Not equal 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). h : Create header cells acc : 0 or 1 to reject or fail to reject the null hypothesis

Help Topics Tree 13217	Help Text
	tT : Test T-value tM : Test mean prob : Lower-tail probability df : Degrees of freedom cT : Critical T-value cdx : Critical value of the delta mean associated with the critical T-value
ConfZ1mean	ConfZ1mean Function Syntax: ConfZ1mean(input_list, ["configuration"]) ConfZ1mean(SampMean, SampSize, PopStdDevm, ConfLevel, ["configuration"]) The one-sample Normal confidence interval for a mean. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SampMean SampSize PopStdDevm ConfLevel 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). h : Create header cells Z : Critical Z-value low : Lower bound of the confidence interval up : Upper bound of the confidence interval
ConfZ2mean	ConfZ2mean Function Syntax: ConfZ2mean(input_list, ["configuration"]) ConfZ2mean(SampMean1, SampMean2, SampSize1, SampSize2, PopStdDev1, PopStdDev2, ConfLevel, ["configuration"]) The two-sample Normal confidence interval for the difference of two means. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SampMean1 SampMean2 SampSize1 SampSize2 PopStdDev1 PopStdDev2 ConfLevel 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). h : Create header cells Z : Critical Z-value low : Lower bound of the confidence interval up : Upper bound of the confidence interval
ConfZ1prop	ConfZ1prop Function Syntax: ConfZ1prop(input_list, ["configuration"]) ConfZ1prop(SuccCount, SampSize, ConfLevel, ["configuration"]) The one-sample Normal confidence interval for a proportion. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SuccCount SampSize ConfLevel 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). h : Create header cells Z : Critical Z-value low : Lower bound of the confidence interval up : Upper bound of the confidence interval
ConfZ2prop	ConfZ2prop Function Syntax:

Help Topics Tree 13217	Help Text
	ConfZ2prop(input_list, ["configuration"]) ConfZ2prop(SuccCount1, SuccCount2, SampSize1, SampSize2, ConfLevel, ["configuration"]) The two-sample Normal confidence interval for the difference of two proportions. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SuccCount1 SuccCount2 SampSize1 SampSize2 ConfLevel 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). h : Create header cells Z : Critical Z-value low : Lower bound of the confidence interval up : Upper bound of the confidence interval
ConfT1mean	ConfT1mean Function Syntax: ConfT1mean(input_list, ["configuration"]) ConfT1mean(SampMean, SampStdDev, SampSize, ConfLevel, ["configuration"]) The one-sample Student’s T confidence interval for a mean. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SampMean SampStdDev SampSize ConfLevel 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). h : Create header cells df: Degrees of freedom T : Critical T-value low : Lower bound of the confidence interval up : Upper bound of the confidence interval
ConfT2mean	ConfT2mean Function Syntax: ConfT2mean(input_list, ["configuration"]) ConfT2mean(SampMean1, SampMean2, SampStdDev1, SampStdDev2, SampSize1, SampSize2, pooled, ConfLevel, ["configuration"]) The two-sample Student’s T confidence interval for the difference of two means. input_list: A list of input variables (see Input Parameters below). This can be a range reference, a list of cell references, or a simple list of values. Input Parameters: SampMean1 SampMean2 SampStdDev1 SampStdDev2 SampSize1 SampSize2 pooled : 0 (not pooled) or 1 (pooled) ConfLevel 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). h : Create header cells df: Degrees of freedom T : Critical T-value low : Lower bound of the confidence interval up : Upper bound of the confidence interval

CellHasData	Syntax:
	CellHasData()
	CellHasData(Row,Column)
	CellHasData(Row_1, Column_1,Row_2, Column_2)

Help Topics Tree 13217	Help Text
	CellHasData returns the number of cells containing data in the designated set of cells. If the command is given no parameters, all cells containing data will be counted. If a single Row and Column is specified, only the cell at that location will be counted. If either Row or Column is -1, all cells containing data in the specified column or row will be counted. If a second Row and Column is specified, all cells will be counted in a rectangular area where the upper left corner is the cell at Row_1, Column_1 and the lower right corner is the cell at Row_2, Column_2. Examples: CellHasData() counts all cells containing data in the entire spreadsheet. CellHasData(3,4) counts only the cell at row 3, column 4. CellHasData(-1,7)counts all cells containing data in column 7. CellHasData(4,-1) counts all cells containing data in row 4. CellHasData(3,4,6,8) counts all cells containing data from row 3, column 4 to row 6, column 8.
ClearCell	Syntax: ClearCell() ClearCell(Row,Column) ClearCell(Row_1, Column_1,Row_2, Column_2) ClearCell clears each spreadsheet cell designated by pairs of Row and Column, including all elements of a list of lists of Row and Column pairs. CellHasData clears each cell in the designated set of cells. If the command is given no parameters, all cells will be cleared. If a single Row and Column is specified, only the cell at that location will be cleared. If either Row or Column is -1, all cells in the specified column or row will be cleared. If a second Row and Column is specified, all cells will be cleared in a rectangular area where the upper left corner is the cell at Row_1, Column_1 and the lower right corner is the cell at Row_2, Column_2. Examples: ClearCell() clears the entire spreadsheet. ClearCell(3,4) clears the cell at row 3, column 4. ClearCell(-1,7)clears all cells in column 7. ClearCell(4,-1) clears all cells in row 4. ClearCell(3,4,6,8) clears all cells from row 3, column 4 to row 6, column 8.
Statistics 1Var app	The 1-Variable Statistics 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 1-Variable Statistics app starts with the Numeric view, which is used to enter data. The Symbolic view is used to define an analysis by specifying which column contains data and which column (if any) contains frequencies. The Plot view is used to display statistical plots of 1-variable data, including histograms, box-and-whisker plots, normal quantile plots and other types of plots. To launch the Statistics 1Var app, go to the Application Library and tap the Statistics 1Var app icon. You can also use the rocker wheel to select the Statistics 1Var app icon, then tap Start or press Enter to launch the app.
Stats 1Var Symbolic View	Statistics 1Var Symbolic View Press Symb to enter the Symbolic view. You can define up to 5 1-variable analyses (H1-H5), choosing for each analysis a data column and an optional frequency column. For the data column, you can enter either the name of a column (D0-D9) or a mathematical expression involving the name of a column (e.g. D1-9.8). There is also a Plot field for each analysis where you choose the graphical representation most fitting for your purposes. The plot options include: ● 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, with either single stems (10's) or split stems (5's) ● Pie charts The menu buttons are: • Edit: enables you to edit the selected value • Choose: select the plot type or graph color • ✓: toggles between making an analysis active or inactive • Column: select the name of a column from Numeric view • Show: displays the fit equation in full-screen mode with horizontal and vertical scrolling enabled • Eval: evaluates the highlighted expression, resolving any references to other definitions Each active data set (H1-H5) will be used for graphing purposes in Plot view and also for displaying summary statistics in Numeric view when Stats is tapped.
Stats 1Var Plot View	Statistics 1Var Plot View

Help Topics Tree 13217	Help Text
	Press Plot to enter the Plot view. This view displays the selected 1-variable statistical plots for the active analyses (H1-H5). The menu is similar to the Function Plot view, with options for zooming and tracing. Tap Menu to toggle the menu on and off. The menu buttons are: • Zoom: opens the Zoom menu, with options to zoom in or out, etc. • Trace: toggles the tracing cursor on and off • Defn: displays the definition of the function being traced • Menu: toggles the menu on and off Use the rocker wheel left/right to trace along a statistical plot. Use the rocker wheel up/down to switch from one plot to another. Press + to zoom in on the current cursor location and press - to zoom out. Set the zoom factor under the Zoom menu. All of the gestures common to the Plot views are supported here as well. See Plot View for more details.
Stats 1Var Plot Setup	Statistics 1Var Plot Setup Press Shift Plot to enter the Statistics 1-Var Plot Setup. Page 1 of the Plot Setup contains settings that control the appearance of 1-variable statistical plots. The fields are: • H Width: the bin width for histograms • H Rng: the range for the data to plot • X Rng: the horizontal range of the graph window • Y Rng: the vertical range of the graph window • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are: • Edit: opens an edit box to edit the value of the selected field • Page 1/2 ▼: displays the second page of the setup Tap Page 1/2 ▼ to view the second page of the setup. Here the fields are: • Axes: toggles axes on and off • Labels: toggles axis labels on and off • Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors The menu buttons on the second page are: • ✓: toggles the current setting on or off • Choose: make a choice from a choose box • ▲ Page 2/2: returns to the first page of the setup
Stats 1Var Numeric View	Statistics 1Var Numeric View Press Num to return to this view at any time. This view contains a table with up to ten columns of data, named D1 through D9 and D0. The menu buttons are: • Edit: opens an input line to edit the chosen value • More: opens a menu with options for editing the list • Go To: jumps to a specific element in the list. Useful for very large lists. • Sort: sorts the current column in either ascending or descending order • Make: generates a column of data based on an algebraic formula • Stats: provides summary statistics on the currently defined analyses (see Symbolic view) Enter your data manually or store list data in D1, D2, etc. Use the Make feature to create data based on an algebraic formula. You can also paste data copied from another app or from the List and Matrix Editors. You can name each data column as well. Tap on the column header and then either edit the name or start typing to enter a new name. The More Menu The More menu contains the following options for editing a list: • Insert o Row: Inserts a new row in the current list. The new row contains 0 as its element. • Delete o Column: Deletes the contents of the current list. To delete a single element, select it and press the Delete key. • Select o Row: Selects the current row. Once selected, the row can be copied. o Column: Selects the current list. Once selected, the list can be copied. o 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 on a cell to start selection, then drag to select a rectangular array of elements. Once selected, the array can be copied. o Swap Ends: Swaps the starting and ending cells for the selected rectangular array of cells. • Selection: Toggles selection mode on and off. You can also tap and hold on a cell, then drag to select. • Swap o Column: Swaps the contents of two columns (lists).

Help Topics Tree	13217	Help Text
Make Column		The Make Column Data wizard is basically a shortcut to using the MAKELIST command and storing the results in Numeric view. The fields in this wizard are: • Expression: enter the generating expression for your list of real numbers • Var: declare the independent variable from your expression. All other variables in your expression will be taken as constants. Note that your variable may be a dummy; that is, it may not appear in your expression at all. • Start: enter the starting vale for your variable • Stop: enter the final value for your variable • Step: enter the step-value for your variable • Col: a drop-down list to choose the destination for your list of real numbers Fill in the fields and tap OK to generate the column of numbers and save them to the list you specified, or tap Cancel to return to Numeric view without creating a list. Use this wizard to easily create a list of random integers, a sample distribution, and so on.
Statistics 1Var Variables		To display the variables relating to the Statistics 1Var app, press Vars, tap App and select Statistics 1Var. The Statistics 1Var app has the following variables: • Results (see below) • Symbolic (see below) • Plot (see (see below) • Numeric (see below) • Modes (see Common App Variables above)
Results Variables		Statistics 1Var Results Variables The Statistics 1Var app variables store results from the calculations performed when the Stats button is tapped in the Numeric view of the app or when the Do1VStats command is executed.
NbItem		NbItem App Variable NbItem : The number of data points in the current 1-variable analysis (H1-H5).
MinVal		MinVal App Variable MinVal : The minimum value of the data set in the current 1-variable analysis (H1-H5).
Q_1		Q_1 App Variable Q_1 : The value of the first quartile in the current 1-variable analysis (H1-H5).
MedVal		MedVal App Variable MedVal : The median in the current 1-variable analysis (H1-H5).
Q_3		Q_3 App Variable Q_3 : The value of the third quartile in the current 1-variable analysis (H1-H5).
MaxVal		MaxVal App Variable MaxVal : The maximum value in the current 1-variable analysis (H1-H5).
ΣX		ΣX App Variable ΣX : The sum of the data set in the current 1-variable analysis (H1-H5).
ΣX2		ΣX2 App Variable ΣX2 : The sum of the squares of the data set in the current 1-variable analysis (H1-H5).
MeanX		MeanX App Variable MeanX : The mean of the data set in the current 1-variable analysis (H1-H5).
sX		sX App Variable sX : The sample standard deviation of the data set in the current 1-variable analysis (H1-H5).
σX		σX : The population standard deviation of the data set in the current 1-variable analysis (H1-H5).
serrX		serrX App Variable serrX : The standard error of the data set in the current 1-variable analysis (H1-H5).
ssX		ssX App Variable ssX : The sum of the squared deviations of x from the mean of x of the data set in the current 1-variable analysis (H1-H5).
Symbolic Variables		Statistics 1Var Symbolic Variables The Statistics 1Var symbolic variables are H1 to H5. These variables contain the data values for a 1-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: • A string or expression (in single quotes) that defines the data list • A string or expression (in single quotes) that optionally defines the frequencies for each of the values in the data list • The plot type number • The option number • The color for the plot The plot type number is an integer from 1-9 that controls which statistical plot type is used with each of the variables H1-H5. The correspondence is shown below. 1 Histogram (default) 2 Box and Whisker 3 Normal Probability 4 Line 5 Bar 6 Pareto

Help Topics Tree 13217	Help Text
	7 Control 8 Dot 9 Stem and Leaf The option number is an integer from 0-2 which controls any option available for the plot type. The correspondence is shown below. 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, no frequencies, and draw a Box and Whisker plot without outliers using a blue color.
Numeric Variables	Statistics 1Var Numeric Variables The Statistics1Var Numeric variables are D1 through D9 and D0. They each represent a single dataset and contain the values in that dataset. These are all list variables and are compatible with the Statistics 2Var list variables C0-C9 as well as the Home variables L0-L9.
Plot Variables	Statistics 1Var Plot Variables In addition to the common Plot view variables (see Common App Variables above), the Statistics 1Var app has three app-specific variables: • Hmin – minimum value to be included in the histogram • Hmax – maximum value to be included in the histogram • Hwidth – width of a histogram bar (bin width)
Hwidth	Hwidth Variable Hwidth : The width of a bar for histogram plots (bin width).
Hmin	Hmin Variable Hmin : The minimum value to be included in the histogram.
Hmax	Hmax Variable Hmax : The maximum value to be included in the histogram.
Statistics 1Var App Functions	The Statistics 1Var app has a 3 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	Do1VStats App Function Syntax: Do1VStats(Hn) Performs the same calculations as pressing the Stats menu key in the Statistics 1Var app's Numeric view 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.
SetFreq	SetFreq App Function Syntax: SetFreq(Hn, Dn) or SetFreq(Hn, Num) Set frequency. The syntax may be either SetFreq(Hn, Dn) or SetFreq(Hn, Num). 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 column variables 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. Examples: SetFreq(H1, 7) sets the frequency for each value in the data set for analysis H1 to be 7. SetFreq(H3, D3) sets the frequencies for the data set in H3 to be column D3.
SetSample	SetSample App Function Syntax: SetSample(Hn, Dn) 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 to one of the column variables D0-D9. Example: SetSample(H2,D4) sets analysis H2 to use column D4 for its data.
CHECK	CHECK App Function Syntax: CHECK(n) Checks (selects) the corresponding definition in Symbolic view. The integer n must be between 0 and 5.
UNCHECK	UNCHECK App Function Syntax: UNCHECK(n) Unchecks (deselects) the corresponding definition in Symbolic view. The integer n must be between 0 and 5.
ISCHECK	ISCHECK App Function Syntax: ISCHECK(n) Returns 1 or 0 depending if the corresponding definition in Symbolic view is selected or not. The integer n must be between 0 and 5.

Help Topics Tree	13217	Help Text
Statistics 2Var app		The 2-Variable Statistics 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 2-Variable Statistics app starts with the Numeric view which is used to enter data. The Symbolic view is used to specify which columns contain data. To launch the Statistics 2Var app, go to the Application Library and tap the Statistics 2Var app icon. You can also use the rocker wheel to select the Statistics 2Var app icon, then tap Start or press Enter to launch the app.
Stats 2Var Symbolic View		Statistics 2Var Symbolic View Press Symb to enter the Statistics 2-Var Symbolic view. You can define up to 5 2-variable analyses, named S1-S5. Each data set definition has the following fields: • Sn: defines the independent and dependent columns as well as an optional frequency column for the data in the dependent column. You can also select a point type and a color for the scatter plot. • Type: chooses a function type to fit to your data • Fit: contains the equation of your fit as well as a color picker to the left of Fit so you can choose a color for the graph of the fit. For the independent and dependent columns, you can enter mathematical expressions in terms of a column name (e.g. 2-C1). Each active data set (S1-S5) will be used for graphing purposes in Plot view and also for displaying summary statistics in Numeric view when Stats is tapped. The menu buttons are: • Edit: enables you to edit the selected value • Choose: select the plot type or graph color • ✓: toggles between making an analysis active or inactive • Column: a choose box for selecting the name of a column • Fit: toggles the fit on and off in Plot view • Show: displays the fit equation in full-screen mode with horizontal and vertical scrolling enabled • Eval: evaluates the highlighted expression, resolving any references to other definitions
Stats 2Var Plot View		Statistics 2Var Plot View Press Plot to enter the Stats 2-Var Plot view. This view displays the scatter plots for the active analyses. Tap Menu to toggle the menu on and off. The menu buttons are: • Zoom: zooms in or out on the graph(s) • Trace: toggles tracing on and off • Go To: opens an input form to jump the tracer to a specific x-value • Fit: toggles displaying a fit for each scatter plot • FCN: opens the Functions menu • Fit: this is a duplicate of the Fit menu key described above • Sketch: sketch your own fit with your finger (see below) • Defn: shows the definition of the current graph being traced • Menu: toggles the menu on and off Sketch, Transform, and Definition Selecting the Sketch option returns you to Plot view, with a message at the bottom of the display to sketch a function fit with your finger. Sketch your fit with your finger; tap OK if you like your sketch, or just sketch a new fit if you do not like your original sketch. When you get the fit you want, tap OK. The fit type for the current dataset in Symbolic view (S1-S5) will be changed to User-Defined and the expression (in X) of your fit will be saved as the user-defined fit. Select Transform to translate and dilate the current function graph. Drag to translate and pinch vertically or horizontally to dilate the graph. The expression will respond accordingly. Tap Simplify to simplify the current expression. Tap Form to select an alternate form for your function equation. Select Definition to view the expression for the current graph. In the Definition Box, you can tap Edit to edit the expression; when you are done editing, tap OK to see the change in the graph. You can also tap Transform to manipulate the current graph. Tap OK when you are done to return to the Definition Box. Tap the down-arrow menu button again to close the box. Use the rocker wheel left/right to trace along a scatter plot or a fit. Use the rocker wheel up/down to move from one scatter plot to the next, or from the scatter plot to the fit. Press + to zoom in on the current cursor location and press - to zoom out. Set the zoom factor under the Zoom menu. All of the gestures common to the Plot views are supported here as well. See Plot View for more details.
Stats 2Var Plot Setup		Statistics 2Var Plot Setup Press Shift Plot to enter page 1 of the Stats 2-Var Plot Setup. This view is similar to page 1 of the Stats 1-Var Plot Setup, except that you can choose different marks for each scatter plot. On the first page, the fields are: • S1Mark-S5Mark: choose a style for the data point marks for each scatter plot • X Rng: the horizontal graphing range • Y Rng: the vertical graphing range • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are:

Help Topics Tree 13217	Help Text
	• Edit: opens an edit box to edit the value of the selected field • Page 1/2 ▼: displays the second page of the setup Tap Page 1/2 ▼ to view the second page of the setup. Here the fields are: • Axes: toggles axes on and off • Labels: toggles axis labels on and off • Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors • Method: choose the method used to plot the fit • Connect: connect the scatter plot points with segment (this is not a fit) • Fit: toggle the fit plotting off and on in Plot view The menu buttons on the second page are: • ✓: toggles the current setting on or off • Choose: make a choice from a choose box • ▲ Page 2/2: returns to the first page of the setup
Stats 2Var Numeric View	Statistics 2Var Numeric View Press Num to return to this view at any time. This view contains a table with up to ten columns of data, named C1 through C9 and C0. The menu buttons are: • Edit: opens an input line to edit the chosen value • More: opens a menu with options for editing the list • Go To: jumps to a specific element in the list. Useful for very large lists. • Sort: sorts the current column in either ascending or descending order • Make: generates a column of data based on an algebraic formula • Stats: provides summary statistics on the currently defined analyses (see Symbolic view) Enter your data manually or store list data in C1, C2, etc. Use the Make feature to create data based on an algebraic formula. You can also paste data copied from another app or from the List and Matrix Editors. You can name each data column as well. Tap on the column header and then either edit the name or start typing to enter a new name. The More Menu The More menu contains the following options for editing a list: • Insert o Row: Inserts a new row in the current list. The new row contains 0 as its element. • Delete o Column: Deletes the contents of the current list. To delete a single element, select it and press the Delete key. • Select o Row: Selects the current row. Once selected, the row can be copied. o Column: Selects the current list. Once selected, the list can be copied. o 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 on a cell to start selection, then drag to select a rectangular array of elements. Once selected, the array can be copied. o Swap Ends: Swaps the starting and ending cells for the selected rectangular array of cells. • Selection: Toggles selection mode on and off. You can also tap and hold on a cell, then drag to select. • Swap o Column: Swaps the contents of two columns (lists).
Make Column	The Make Column Data wizard is basically a shortcut to using the MAKELIST command and storing the results in Numeric view. The fields in this wizard are: • Expression: enter the generating expression for your list of real numbers • Var: declare the independent variable from your expression. All other variables in your expression will be taken as constants. Note that your variable may be a dummy; that is, it may not appear in your expression at all. • Start: enter the starting vale for your variable • Stop: enter the final value for your variable • Step: enter the step-value for your variable • Col: a drop-down list to choose the destination for your list of real numbers Fill in the fields and tap OK to generate the column of numbers and save them to the list you specified, or tap Cancel to return to Numeric view without creating a list. Use this wizard to easily create a list of random integers, a sample distribution, and so on.
Statistics 2Var Stats View	In the Statistics 2Var app Numeric view, tap Stats to see summary statistics for all active analyses displayed in the Stats view. By default, the usual 2-variable statistics are displayed and the Stats menu key has the active white dot in it. The menu keys in this view are: • More: opens a menu that lets you select and copy multiple cells. You can then paste the contents of these cells elsewhere. • Stats: displays common 2-variable summary statistics, such as the correlation coefficient and the sample and population covariances. • X: displays summary statistics for the independent variable, such as the mean of x and its standard deviation.

Help Topics Tree	13217	Help Text
		• Y: displays summary statistics for the dependent variable, such as the mean of Y and its standard deviation. • OK: returns to Numeric view Use the More menu to select one or more of the statistics on a page and then press Shift View (Copy) to copy the array to the clipboard. You can then press Shift Menu (Paste) to open the clipboard and paste the array anywhere in the system, such as the List or Matrix editors or the Spreadsheet app Numeric view. When you are done examining summary statistics, press OK to return to Numeric view.
Statistics 2Var Variables		To display the variables relating to the Statistics 2Var app, press Vars, tap App and select Statistics 2Var. The Statistics 2Var app has the following variables: • Results (see below) • Symbolic (see below) • Plot (see (see Common App Variables above) • Numeric (see below) • Modes (see Common App Variables above)
Results Variables		Statistics 2Var Results Variables The Statistics 2Var app variables store results from the calculations performed when the Stats button is tapped in the Numeric view of the app or when the Do2VStats command is executed.
NbItem		NbItem App Variable NbItem contains the number of data points in the current 2-variable analysis (S1-S5).
Corr		Corr App Variable Corr contains the correlation coefficient from the latest calculation of summary statistics.
CoefDet		CoefDet App Variable CoefDet contains the coefficient of determination from the latest calculation of summary statistics. This value is based on the fit type chosen.
sCov		sCov App Variable sCov contains the sample covariance of the current 2-variable statistical analysis (S1-S5).
σCov		σCov App Variable σCov contains the population covariance of the current 2-variable statistical analysis (S1-S5).
ΣXY		ΣXY App Variable ΣXY contains the sum of the X·Y products for the current 2-variable statistical analysis (S1-S5).
MeanX		MeanX App Variable MeanX contains the mean of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
ΣX		ΣX App Variable ΣX contains the sum of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
ΣX2		ΣX2 App Variable ΣX2 contains the sum of the squares of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
sX		sX App Variable sX contains the sample standard deviation of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
ΣX		ΣX App Variable ΣX contains the population standard deviation of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
serrX		serrX App Variable serrX contains the standard error of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
ssX		ssX App Variable ssX contains the sum of the squared deviations of x from the mean of x of the independent values (X) of the current 2-variable statistical analysis (S1-S5).
MeanY		MeanY App Variable MeanY contains the mean of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
ΣY		ΣY App Variable ΣY contains the sum of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
ΣY2		ΣY2 App Variable ΣY2 contains the sum of the squares of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
sY		sY App Variable sY contains the sample standard deviation of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
σY		σY App Variable σY contains the population standard deviation of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
serrY		serrY App Variable serrY contains the standard error of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
ssY		ssY App Variable

Help Topics Tree	13217	Help Text
		ssY contains the sum of the squared deviations of y from the mean of y of the dependent values (Y) of the current 2-variable statistical analysis (S1-S5).
Symbolic Variables		Statistics 2Var Symbolic Variables The Statistics 2Var app variables are S1-S5. These variables contain the data that define a 2-variable statistical analysis. S1 returns a list of the objects that define S1. Each list contains the following items, in order: • A string or expression (in single quotes) that defines the independent variable data list • A string or expression (in single quotes) that defines the dependent variable data list • A string or expression that optionally defines the frequencies for the dependent data list • The fit type number (see below) • The fit expression • The scatter plot color • The scatter plot point mark type number • 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-S5. The correspondence is shown below. 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 scatter plot point mark type number is an integer from 1 to 9 that controls which graphic is used to represent each point in a scatter plot. The correspondence is shown below. 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 yet, a blue scatter plot with mark type 1, and a blue fit plot.
Numeric Variables		Statistics 2Var Numeric Variables The Statistics 2Var Numeric app variables are C1 through C9 and C0. They each represent a single dataset and contain the values in that dataset. These are all list variables and are compatible with the Statistics 1Var list variables D0-D9 as well as the Home variables L0-L9.
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		PredX App Function Syntax: PredX(Y_value) Predict X. Uses the fit from the first active analysis (S1-S5) found to predict an x-value given the Y-value.
PredY		PredY App Function Syntax: PredY(X_value) Predict Y. Uses the fit from the first active analysis (S1-S5) found to predict a y-value given the x-value.
Resid		Resid App Function Syntax: Resid(Sn) or Resid() Residuals. Calculates a list of residuals, based on column data and a fit defined in the Symbolic view via S1-S5. Example:

Help Topics Tree 13217	Help Text
	Resid() looks for the first active, defined analysis in the Symbolic view (S1-S5). Resid(S3) uses analysis S3
Do2VStats	Do2VStats App Function Syntax: Do2VStats(Sn) Performs the same calculations as pressing the Stats menu key in the Statistics 2Var app's Numeric view 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.
SetDepend	SetDepend App Function Syntax: SetDepend(Sn, Cn) Set dependent column. Sets the dependent column for one of the statistical analyses S1-S5 to one of the column variables C0-C9. Example: SetDepend(S1, C3) sets the dependent column for analysis S1 to column C3.
SetIndep	SetIndep App Function Syntax: SetIndep(Sn, Cn) Set independent column. Sets the independent column for one of the statistical analyses S1-S5 to one of the column variables C0-C9. Example: SetIndep(S1, C2) sets the independent column for analysis S1 to column C2.
CHECK	CHECK App Function Syntax: CHECK(n) Checks (selects) the corresponding definition in Symbolic view. The integer n must be between 0 and 5.
UNCHECK	UNCHECK App Function Syntax: UNCHECK(n) Unchecks (deselects) the corresponding definition in Symbolic view. The integer n must be between 0 and 5.
ISCHECK	ISCHECK App Function Syntax: ISCHECK(n) Returns 1 or 0 depending if the corresponding definition in Symbolic view is selected or not. The integer n must be between 0 and 5.
Inference app	The Inference app contains tools for inferential statistics, including creating confidence intervals and hypothesis testing. This app can import summary statistics from any column of the Statistics 1-Var or 2-Var apps. Confidence intervals and hypothesis tests are based on the Normal Z-distribution or Students T-distribution. Results can be displayed both numerically and graphically. To launch the Inference app, go to the Application Library and tap the Inference app icon. You can also use the rocker wheel to select the Inference app icon, then tap Start or press Enter to launch the app. Although the intervals and tests in this app are limited to the Normal and Student's-t distributions, HP Prime has a full set of probability density functions. Press the Toolbox key, tap Math and select Probability. You will see categories for Density, Cumulative, and Inverse (Cumulative). Under each you will find Normal, T, Chi-Square, F, Binomial, and Poisson.
Inference Symbolic View	The Inference Symbolic view contains settings to define a confidence interval or hypothesis test. This view also allows you to select a Chi-Square test or perform inference for linear regression. The fields are: • Method: chooses between hypothesis test, confidence interval, Chi-Square, Inference for regression, or ANOVA • Type: chooses a specific calculation within the selected Method, where available • Alt Hypoth: chooses one of three alternative hypotheses (hypothesis test only) • Expected: chooses between entering expected probabilities and expected counts for the Chi-Square Goodness of Fit (GoF) test The only menu button is: • Choose: make a choice from a choose box Once you have selected a specific calculation, press the Num key to go to Numeric view and enter the data for the calculation.
Inference Plot View	The Inference Plot view displays graphically the results you see when you tap Calc from the Numeric view. Not all calculations include a Plot view.
Inference Plot Setup	Press Shift Plot to enter the Inference Plot Setup. Page 1 of the Plot Setup contains settings that control the appearance of inference plots. The fields are: • H Width: the bin width for histograms • H Rng: the range for the data to plot • X Rng: the horizontal range of the graph window • Y Rng: the vertical range of the graph window • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing

Help Topics Tree **13217** **Help Text**

Inference Numeric View

The Inference app Numeric view is designed for you to enter data required for the calculation selected in Symbolic view.

Hypothesis Tests and Confidence Intervals

For hypothesis tests and confidence intervals, this view contains fields for the sample statistics (e.g. sample mean and sample size), the population parameters (e.g. null hypothesis mean, and standard deviation, σ), and the significance level.

For hypothesis tests and confidence intervals, the menu buttons are:

• Edit: opens an input box to edit the chosen value

• Import: imports statistics (such as n, mean, etc.) from any column in the Statistics 1Var or Statistics 2Var apps (or any app based on these two)

• Calc: computes and displays the results numerically in a table

Enter values in the fields and then tap Calc to see the results.

Chi-Square Tests

For the Chi-Square GoF test, Numeric view presents two lists. The first list, ObsList, is for the observed counts. The second list is either ProbList for the expected probabilities or ExpList for the expected counts, depending on the choice you made in Symbolic view for the Expected field. Enter the observed counts in the first list and either the expected probabilities or counts in the second list. Tap Calc to see the results.

The menu buttons here are:

• Edit: opens an input line to edit the chosen value

• More: opens a menu with options for editing the list

• Go To: jumps to a specific element in the list. Useful for very large lists.

• Make: generates a column of data based on an algebraic formula

• Calc: displays the test results

For the Chi-Square 2-way test, Numeric view presents a matrix named ObsMat, in which you enter the matrix of observed counts. When you are done, tap Calc to see the results. The top of the screen will display the Chi-Square statistic value, the probability, and the degrees of freedom.

The menu buttons are the same as for the GoF test, except that there is no Make option. There is also a Go menu key that toggles between moving the cursor right, down, or not at all after Enter is pressed.

Inference for Linear Regression

For all linear regression options, Numeric view presents two lists; Xlist for the x-values (the explanatory variable) and Ylist for the y-values (the response variable). Enter your data in these two variables. Press the Plot key to view a scatter plot of your data. Use the rocker wheel up/down to view a scatter plot of the residuals and a normal probability plot of the residuals. These plots will help you assess whether or not your data meet the criteria for the linear t-test, etc. When you are ready, tap Calc. A wizard will open to guide you through the rest of the steps, such as entering a confidence level or an x-value, depending on the calculation you chose in Symbolic view.

The menu buttons here are the same as for the Chi-Square GoF test Numeric view.

ANOVA

For analysis of variance, Numeric view presents a set of columns. Enter each data set involved in the analysis into its own column and then tap Calc. The results of the analysis of variance will be displayed.

The menu buttons here are the same as for the Chi-Square GoF test Numeric view.

Inference Calc View

Tap Calc to see the calculation results. The Inference Results screen is displayed. This view varies depending on the specific calculation results being displayed.

In general, results are displayed in a table whenever possible. Each value is labeled and help text is provided to describe each value.

Hypothesis Tests

For hypothesis tests, Calc displays the test result, the test values, the probability, and the critical values. The menu keys here include the More menu to assist you in copying results to paste elsewhere. Tap the OK menu key to return to Numeric view when you are done.

Confidence Intervals

For confidence intervals, Calc displays the C-value, the degrees of freedom, the critical value, and the lower and upper bounds of the confidence interval. The menu keys here are the same as for Hypothesis Tests.

Chi-Square GoF

For the goodness of fit test, Calc displays the Chi-Square value and its probability, along with the degrees of freedom. The menu keys are:

• More: opens the More menu

• Stats: displays the default results described above

• Exp: switches to display the expected counts

• Cont: switches to display the Chi-Square contributions by category

• OK: returns to Numeric view

Chi-Square Two-Way Test

For the two-way test, Calc displays the Chi-Square value and its probability, along with the degrees of freedom. The menu keys are the same as for the goodness of fit test.

ANOVA

For the 1-way analysis of variance, Calc displays the degrees of freedom, the f-distribution value, etc. The menu keys are More (to select multiple cells to copy and paste elsewhere) and OK (to return to Numeric view).

See the individual help pages for:

• Hypothesis Tests

Help Topics Tree 13217	Help Text
	• Confidence Intervals • Chi-Square Goodness of Fit Test • Chi-Square 2-Way Test • Linear Regression T-Test • Confidence Interval for Slope • Confidence Interval for Intercept • Confidence Interval for Mean Response • Prediction Interval for Future Response • ANOVA
Hypothesis Test Results	Once you tap Calc in Numeric view, the results of your hypothesis test are displayed in a table. Generally, these results include the following values: • Result: 0 to reject or 1 to fail to reject the null hypothesis • Test value: the Z- or t-value calculated for the test • Test statistic value: the value of the statistic under scrutiny associated with the Test value • P: the probability • Critical test value: the boundary test value(s) for your test • Critical statistic value: the boundary value(s) for the statistic under scrutiny The menu buttons are: • More: opens the More menu, with options for selecting multiple cells to copy and then paste elsewhere • OK: return to the Numeric view of the app Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view. Now you can press the Plot key to see the results graphically as well.
Confidence Intervals Results	Once you tap Calc in Numeric view, the results of your confidence interval calculation are displayed in a table. Generally, these results include the following values: • C: the confidence level you entered • Critical test values: the boundary test values associated with your confidence level • Lower: the lower bound of the confidence interval • Upper: the upper bound of the confidence interval The menu buttons are: • More: opens the More menu, with options for selecting multiple cells to copy and then paste elsewhere • OK: return to the Numeric view of the app Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view. Now you can press the Plot key to see the results graphically as well.
Chi-Square GoF Results	Once you tap Calc in Numeric view, the results of your Chi-Square goodness of fit calculation are displayed in a table containing the following values: • χ^2: the Chi-Square statistic value • P: the probability • DF: the degrees of freedom The menu keys are: • More: opens the More menu, with options for selecting multiple cells to copy and then paste elsewhere • Stats: tap to display the test results (on by default) • Exp: tap to display the list of the expected counts • Cont: tap to display the list of Chi-Square contributions • OK: tap to return to Numeric view Use the rocker wheel or tap to scroll about the table. Use the More menu to select multiple cells to copy and paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells. Tap the OK menu key to return to Numeric view.
Chi-Square 2-Way Test Results	Once you tap Calc in Numeric view, the results of your Chi-Square 2-way test are displayed in a table containing the following values: • χ^2: the Chi-Square statistic value • P: the probability • DF: the degrees of freedom The menu keys are: • More: opens the More menu, with options for selecting multiple cells to copy and then paste elsewhere • Exp: tap to display the matrix of the expected counts (tap OK to exit) • Cont: tap to display the matrix of Chi-Square contributions (tap OK to exit) • OK: tap to return to Numeric view Use the rocker wheel or tap to scroll about the table. Use the More menu to select multiple cells to copy and paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells. Tap the OK menu key to return to Numeric view.

Help Topics Tree				13217	Help Text
				Linear Regression T-Test Results	Once you tap Calc in Numeric view, the linear regression t-test results are displayed in a table containing the following values: • Test T: the calculated test t-value • P: the probability • DF: the degrees of freedom • β0: the linear regression equation intercept • β1: the linear regression equation slope • serrLine: the standard error about the line • serrSlope: the standard error about the slope • serrInt: the standard error about the intercept • r: the correlation value • R^2: the coefficient of determination Use the rocker wheel or tap to scroll about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view.
				CI for Slope Results	Once you tap Calc in Numeric view, the results of the confidence interval calculation are displayed in a table containing the following values: • C: the confidence level you entered in the wizard • Crit. T: the critical t-value associated with your confidence level • DF: the degrees of freedom • β1: the linear regression equation slope • serrLine: the standard error about the line • Lower: the lower bound of the confidence interval for the slope • Upper: the upper bound of the confidence interval for the slope Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view.
				CI for Intercept Results	Once you tap Calc in Numeric view, the results of the confidence interval calculation are displayed in a table containing the following values: • C: the confidence level you entered in the wizard • Crit. T: the critical test t-value associated with your confidence level • DF: the degrees of freedom • β0: the linear regression equation intercept • serrLine: the standard error about the line • Lower: the lower bound of the confidence interval for the intercept • Upper: the upper bound of the confidence interval for the intercept Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view.
				CI for Mean Response Results	Once you tap Calc in Numeric view, the results of the confidence interval calculation are displayed in a table containing the following values: • C: the confidence level you entered in the wizard • ŷ: the mean response value for the x-value you entered in the wizard • DF: the degrees of freedom • serr ŷ: the standard error about the mean response • Lower: the lower bound of the confidence interval for the mean response • Upper: the upper bound of the confidence interval for the mean response Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view.
				Prediction Interval Results	Once you tap Calc in Numeric view, the results of the prediction interval calculation are displayed in a table containing the following values: • C: the confidence level you entered in the wizard • ŷ: the mean response value for the future x-value you entered in the wizard • DF: the degrees of freedom • serr ŷ: the standard error about the mean response • Lower: the lower bound of the confidence interval for the mean response • Upper: the upper bound of the confidence interval for the mean response Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view.
				ANOVA results	Once you tap Calc in Numeric view, the results of your 1-way analysis of variance are displayed in a table containing the following values: • F: the F-value • P: the probability associated with the F-value

Help Topics Tree 13217	Help Text
	• DF: the degrees of freedom of the treatments • 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 Use the rocker wheel or tap to move about the table. Tap the More menu key to open a menu for options to assist you in selecting multiple cells to copy and the paste elsewhere. You can also tap and hold on a cell, then drag to select a rectangular array of cells to copy and paste elsewhere. Tap the OK menu key to return to Numeric view.
Inference App Variables	Besides the common app vars, the Inference app has Symbolic, Numeric, and Results app vars. Each of these categories contains the app vars used in the corresponding view of the app.
Results Variables	Inference App Results Variables The Inference App Results variables store calculations performed when the Calc menu button is tapped in the Inference Numeric view or when the DoInference command is executed.
Result	Result App Var For hypothesis tests, contains 0 or 1 to indicate rejection of or failure to reject the null hypothesis.
TestScore	TestScore App Var TestScore contains the Z- or t-distribution value calculated from the hypothesis test or confidence interval inputs.
TestValue	TestValue App Var TestValue contains the value of the experimental variable associated with the current value in the app variable TestScore.
CritScore	CritScore App Var CritScore contains the value of the Z- or t-distribution associated with the input α-value
CritVal1	CritVal1 App Var 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	CritVal2 App Var CritVal2 contains the upper critical value of the experimental variable associated with the positive TestScore value which was calculated from the input α-level.
Prob	Prob App Var Prob contains the probability associated with the TestScore value.
DF	DF App Var DF contains the degrees of freedom for the t-tests.
ContribList	ContribList App Var ContribList is a list that contains the Chi-Square contributions for the last Chi-Square goodness of fit (GoF) test.
ExpMat	ExpMat App Var ExpMat is a matrix that contains the expected count data from the last Chi-Square 2-Way test calculation.
ContribMat	ContribMat App Var ContribMat is a matrix that contains the Chi-Square contributions for the last Chi-Square 2-way test.
Slope	Slope App Var Slope contains the value of the slope from the last linear regression t-test.
Inter	Inter App Var Inter contains the value of the intercept from the last linear regression t-test.
corr	corr App Var corr contains the value of the correlation from the last linear regression t-test.
coefDet	coefDet App Var coefDet contains the value of the coefficient of determination from the last linear regression t-test.
serrLine	serrLine App Var serrLine contains the value of the standard error of the line from the last linear regression t-test.
serrSlope	serrSlope App Var serrSlope contains the value of the standard error of the slope from the last linear regression t-test or confidence interval for slope.
serrInter	serrInter App Var serrInter contains the value of the standard error of the intercept from the last linear regression t-test or confidence interval for the intercept.
Yval	Yval App Var Yval contains the value of ŷ from the last prediction interval or mean response interval calculation.
serrY	serrY App Var serrY contains the value of the standard error of ŷ from the last prediction interval or mean response interval calculation.
Xval	Xval App Var Xval contains the value of the explanatory variable (X) from the last mean response interval or prediction interval calculation.
SS	SS App Var

Help Topics Tree	13217	Help Text
		SS contains the value of the sum of squares of the treatments from the last ANOVA 1-way calculation.
	SSerr	SSerr App Var SSerr contains the value of the sum of squares of the errors from the last ANOVA 1-way calculation.
	MS	MS App Var MS contains the value of the mean squares for the treatments from the last ANOVA 1-way calculation.
	MSerr	MSerr App Var MSerr contains the value of the mean squares for the errors from the last ANOVA 1-way calculation.
	Fval	Fval App Var Fval contains the value of the mean squares for the treatments from the last ANOVA 1-way calculation.
	DFerr	DFerr App Var DFerr contains the value of the degrees of freedom of the errors from the last ANOVA 1-way calculation.
Numeric Variables		Inference App Numeric Variables The Inference app Numeric Variables correspond to the fields in the Numeric view for the various tests and confidence intervals.
	Alpha	Alpha App Var Sets the alpha level for the hypothesis test. Alpha:=n, where 0<n<1, sets the alpha-level to n.
	Conf	Conf App Var Sets the confidence level for the confidence interval. Conf:=n, where 0<n<1, sets the confidence level to n.
	Mean$_1$	Mean$_1$ App Var 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. n ▶ Mean$_1$ sets the value of Mean$_1$ to n.
	Mean$_2$	Mean$_2$ App Var For a 2-mean test or interval, sets the value of the mean of the second sample. n ▶ Mean$_2$ sets the value of Mean$_2$ to n.
	σ_1	σ_1 App Var Sets the population standard deviation for a hypothesis test or confidence interval involving 1 or 2 means and the Normal distribution. For a test or interval involving the difference of two means, sets the population standard deviation of the first sample. n ▶ σ_1 sets the value of σ_1 to n.
	σ_2	σ_2 App Var For a test or interval involving the difference of two means and the Normal distribution, sets the population standard deviation of the second sample. n ▶ σ_2 sets the value of σ_2 to n.
	s_1	s_1 App Var Sets the sample standard deviation for a hypothesis test or confidence interval. For a test or interval involving the difference of two means, sets the sample standard deviation of the first sample. n ▶ s_1 sets the value of s_1 to n.
	s_2	s_2 App Var For a test or interval involving the difference of two means, sets the sample standard deviation of the second sample. n ▶ s_2 sets the value of s_2 to n.
	x_1	x_1 App Var 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. n ▶ x_1 sets the value of x_1 to n.
	x_2	x_2 App Var For a test or interval involving the difference of two proportions, sets the number of successes of the second sample. n ▶ x_2 sets the value of x_2 to n.
	n_1	n_1 App Var 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. n ▶ n_1 sets the value of n_1 to n.
	n_2	n_2 App Var For a test or interval involving the difference of two means or two proportions, sets the size of the second sample. n ▶ n_2 sets the value of n_2 to n.
	μ_0	μ_0 App Var Sets the assumed value of the population mean for a hypothesis test. n ▶ μ_0 sets the value of μ_0 to n.
	π_0	π_0 App Var Sets the assumed proportion of successes for the one-proportion Z-test. n ▶ π_0 sets the value of π_0 to n.

Help Topics Tree	13217	Help Text
	Pooled	Pooled App Var Determine whether or not the samples are pooled for tests or intervals using the Student's T-distribution involving two means. 0 ▸ Pooled for not pooled (default) 1 ▸ Pooled for pooled
	ObsList	ObsList App Var ObsList is a list that contains the observed counts for each category from the last Chi-Square goodness of fit (GoF) test.
	ProbList	ProbList App Var ProbList is a list that contains the probabilities for each category from the last Chi-Square goodness of fit (GoF) test.
	ExpList	ExpList App Var ExpList is a list that contains the expected counts for each category from the last Chi-Square goodness of fit (GoF) test.
	ObsMat	ObsMat App Var ObsMat is a matrix that contains the observed count data from the last Chi-Square 2-Way test calculation.
	Xlist	Xlist App Var Xlist is a list that contains the data for the explanatory (X) variable from the last inference for regression calculation.
	Ylist	Ylist App Var Ylist is a list that contains the data for the response (Y) variable from the last inference for regression calculation.
Symbolic Variables		Inference App Symbolic Variables There are four Inference App Symbolic Variables, each of which corresponds to one of the four possible fields in the Symbolic view of the Inference app: • Method • Type • AltHyp • DataType
	InfType	InfType App Var InfType determines the type of hypothesis test, confidence interval, Chi-Square test, or inference for regression calculation. Their function depends upon the value of the variable Method. With Method=0 for hypothesis tests, the constant values and their meanings are as follows: • InfType:= 0 for Z-Test: 1 mean • InfType:= 1 for Z-Test: 2 means • InfType:= 2 for Z-Test: 1 proportion • InfType:= 3 for Z-Test: 2 proportions • InfType:= 4 for T-Test: 1 mean • InfType:= 5 for T-Test: 2 means With Method=1 for confidence intervals, the constant values and their meanings are as follows: • InfType:= 0 for Z-Int: 1 mean • InfType:= 1 for Z-Int: 2 means • InfType:= 2 for Z-Int: 1 proportion • InfType:= 3 for Z-Int: 2 proportions • InfType:= 4 for T-Int: 1 mean • InfType:= 5 for T-Int: 2 means With Method=2 for Chi-Square tests, the constant values and their meanings are as follows: • InfType:= 0 for Chi-Square GoF • InfType:= 1 for Chi-Square 2-Way test With Method=3 for inference for regression, the constant values and their meanings are as follows: • InfType:= 0 for Linear regression t-test • InfType:= 1 for confidence interval for slope • InfType:= 2 for confidence interval for intercept • InfType:= 3 for confidence interval for mean response • InfType:= 4 for prediction interval for a future response
	Method	Method App Var Method determines whether the Inference app is set to calculate hypothesis test results, confidence intervals, Chi-Square tests, or inference for regression calculations. • Method := 0 for Hypothesis Tests • Method := 1 for Confidence Intervals • Method := 2 for Chi-Square Tests • Method := 3 for Inference for regression
	AltHyp	AltHyp App Var AltHyp determines the alternative hypothesis used for hypothesis testing. • AltHyp := 0 for μ<μ0 • AltHyp := 1 for μ>μ0 • AltHyp := 2 for μ≠μ0

Help Topics Tree	13217	Help Text

DataType		DataType App Var
		For the Chi-Square goodness of fit (GoF) test, DataType determines whether the expected list contains probabilities or counts.
		DataType:= 0 for count data
		DataType:= 1 for probabilities

Inference App Functions		The functions specific to the Inference app are listed in this section.
DoInference		DoInference App Function Syntax: DoInference() Calculate confidence interval or test hypothesis. Performs the same calculations as tapping Calc in the Inference app's Numeric view, and stores the results in the appropriate Inference app Results variables. The results depend on the contents of the Inference app Symbolic view variables Method, Type, and Alt Hypoth.
HypZ1mean		One-sample Z-test for Mean Syntax: HypZ1mean(SampMean, SampSize, NullPopMean, PopStdDev, SigLevel, Mode) Mode: Specifies which alternative hypothesis to use. 1: Less than 2: Greater than 3: Not equal Returns a list containing (in order): • Reject (0) or fail to reject (1) the null hypothesis • Test Z-value • Input sample mean value • Upper-tail probability • Upper critical Z-value associated with the input α-level • Critical value of the statistic associated with the critical Z-value Example: HypZ1mean(0.461368,50,0.5,0.2887,0.05,1) → {1,−0.946205374811,0.461368,0.172021922639,−1.64485362695,0.432843347747,0.432843347747}
HypZ2mean		Two-sample Z-test for Means Syntax: HypZ2mean(SampMean, SampMean2, SampSize,SampSize2, PopStdDev, PopStdDev2, SigLevel, Mode) Mode: Specifies which alternative hypothesis to use: 1: Less than 2: Greater than 3: Not equal Returns a list containing (in order): • Reject (0) or fail to reject (1) the null hypothesis • Test Z-value • Test Δ⍰ value • Upper-tail probability • Upper critical Z-value associated with the input α-level • Critical value of Δ⍰ associated with the critical Z-value Example: HypZ2mean(0.461368,0.522851,50,50,0.2887,0.2887,0.05,1) → {1,−1.06482507793,−0.061483,0.1434775472,−1.64485362695,−0.156456848420,−0.156456848420}
HypZ1prop		One-sample Z-test for Proportion Syntax: HypZ1prop(SuccCount, SampSize, NullPopProp, SigLevel, Mode) Mode: Specifies which alternative hypothesis to use. 1: Less than 2: Greater than 3: Not equal Returns a list containing (in order): • Reject (0) or fail to reject (1) the null hypothesis • Test Z-value • Test π value • Upper-tail probability • Upper critical Z-value associated with the input α-level • Critical value of π associated with the critical Z-value Example: HypZ1prop(21,50,0.5,0.05,1) → {1,−1.13137084989,0.42,0.128949517646,−1.64485362695,0.385189688145,0.385189688145}
HypZ2prop		Two-sample Z-test for Proportions Syntax: HypZ2prop(SuccCount1, SuccCount2, SampSize1, SampSize2, SigLevel, Mode) Mode: Specifies which alternative hypothesis to use.

Help Topics Tree 13217	Help Text
	1: Less than 2: Greater than 3: Not equal Returns a list containing (in order): • Reject (0) or fail to reject (1) the null hypothesis • Test Z-value • Test Δπ value • Upper-tail probability • Upper critical Z-value associated with the input α-level • Critical value of Δπ associated with the critical Z-value Example: HypZ2prop(21,26,50,50,0.05,1) → {1,−1.00180487462,−0.1,0.42,0.52,0.158218921229,−1.64485362695,−0.263363033245,−0.263363033245}
HypT1mean	One-sample t-test for Mean Syntax: HypT1mean(SampMean, SampStdDev, SampSize, NullPopProp, SigLevel, Mode) Mode: Specifies which alternative hypothesis to use. 1: Less than 2: Greater than 3: Not equal Returns a list containing (in order): • Reject (0) or fail to reject (1) the null hypothesis • Test T-value • Input ▯ value • Upper-tail probability • Degrees of freedom • Upper critical T-value associated with the input α-level • Critical value of the statistic associated with the critical T-value Example: HypT1mean(0.461368,0.2776,50,0.5,0.05,1) → {1,−0.984039955720,0.461368,0.16496500389,49,−1.67655089261,0.434181011953,0.434181011953}
HypT2mean	Two-sample t-test for Means Syntax: HypT2mean(SampMean1, SampMean2, SampStdDev1, SampStdDev2, SampSize1, SampSize2, SigLevel, Pooled, Mode) Pooled: Specifies whether or not the samples are pooled 0: not pooled 1: pooled Mode: Specifies which alternative hypothesis to use. 1: Less than 2: Greater than 3: Not equal Returns a list containing (in order): • Reject (0) or fail to reject (1) the null hypothesis • Test T-value • Test Δ▯ value • Upper-tail probability • The degrees of freedom • Upper critical T-value associated with the input α-level • Critical value of Δ▯ associated with the critical T-value Example: HypT2mean(0.461368,0.522851,0.2776,0.2943,50,50,0.05,0,1) → {1,−1.07460751332,−0.061483,0.142599075544,97.6674459454,−1.66060517920,−0.156493491707,−0.156493491707}
ConfZ1mean	One-sample Normal CI for Mean Syntax: ConfZ1mean(SampMean, SampSize, PopStdDevm, ConfLevel) One-sample Normal confidence interval for a mean Returns a list containing (in order): • Lower critical Z-value • Lower bound of the confidence interval • Upper bound of the confidence interval Example: ConfZ1mean(0.461368,50,0.2887,0.95) → {−1.95996398454,0.381345913182,0.541390086818}
ConfZ2mean	Two-sample Normal CI for Mean Syntax: ConfZ2mean(SampMean1, SampMean2, SampSize1, SampSize2, PopStdDev1, PopStdDev2, ConfLevel) Two-sample Normal confidence interval for the difference of two means

Help Topics Tree 13217	Help Text
	Returns a list containing (in order): • Lower critical Z-value • Lower bound of the confidence interval • Upper bound of the confidence interval Example: ConfZ2mean(0.461368,0.522851,50,50,0.2887,0.2887,0.95) → {−1.95996398454,−0.174651320467,5.16853204673ᴇ−2}
ConfZ1prop	One-sample Normal CI for Proportion Syntax: ConfZ1prop(SuccCount, SampSize, ConfLevel) One-sample Normal confidence interval for a proportion Returns a list containing (in order): • Lower critical Z-value • Lower bound of the confidence interval • Upper bound of the confidence interval Example: ConfZ1prop(21,50,0.95) → {−1.95996398454,0.283195075475,0.556804924525,0.42}
ConfZ2prop	Two-sample Normal CI for Proportions Syntax: ConfZ2prop(SuccCount1, SuccCount2, SampSize1, SampSize2, ConfLevel) Two-sample Normal confidence interval for the difference of two proportions Returns a list containing (in order): • Lower critical Z-value • Lower bound of the confidence interval • Upper bound of the confidence interval Example: ConfZ2prop(21,26,50,50,0.95) → {−1.95996398454,−0.294659060430,9.46590604295ᴇ−2,0.42,0.52}
ConfT1mean	One-sample t-test for Mean Syntax: ConfT1mean(SampMean, SampStdDev, SampSize, ConfLevel) One-sample Student's T confidence interval for a mean Returns a list containing (in order): • Degrees of freedom • Lower critical t-value • Lower bound of the confidence interval • Upper bound of the confidence interval Example: ConfT1mean(0.461368,0.2776,50,0.95) → {49,−2.00957523712,0.382474952915,0.540261047085}
ConfT2mean	Two-sample t-test for Means Syntax: ConfT2mean(SampMean, SampMean2, SampStdDev, SampStdDev2, SampSize, SampSize2, Pooled, ConfLevel) Two-sample Student's T confidence interval for the difference of two means Pooled: Specifies whether or not the samples are pooled 0: not pooled 1: pooled Returns a list containing (in order): • The degrees of freedom • The lower critical t-value • The lower bound of the confidence interval • The upper bound of the confidence interval • The midpoint of the interval Example: ConfT2mean(0.461368,0.522851,0.2887,0.2887,50,50,0.95,0) → {98.0000000000,−1.98446745450,−0.176066150823,5.31001508231ᴇ−2,−0.061483}
Chi2GOF	χ² Goodness of Fit Syntax: Chi2GOF(List1, List2, Value) 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: • χ² statistic value • Probability • Degrees of freedom Example: Chi2GOF({10,10,12,15,10,6},{0.24,0.2,0.16,0.14,0.13,0.13},0) → {7.95179952323,0.158912133127,5}
Chi2TwoWay	χ² Two-Way Test

Help Topics Tree	13217	Help Text
		Syntax: Chi2TwoWay(Matrix) Given a matrix of count data, returns a list containing: • χ² statistic value • Probability • Degrees of freedom Example: Chi2TwoWay([[30,35,30],[11,2,19],[43,35,35]]) → {14.4302681482,6.04117951525ᴇ−3,4}
LinRegrTTest		Linear Regression t-test Syntax: LinRegrTTest(List1, List2, AltHyp) Given a list of explanatory (X) variable data, a list of response (Y) variable data, and an integer (0, 1, or 2), performs a linear regression t-test using the given bivariate data sets. The last argument determines the nature of the alternative hypothesis used for the test, as shown in the following list: • AltHyp := 0 for μ<μ0 • AltHyp := 1 for μ>μ0 • AltHyp := 2 for μ≠μ0 The test returns a list containing the following values in the order shown: • Test t-value: the t-value associated with the test • P: the probability associated with the test result • DF: the degrees of freedom • β0: the intercept of the linear regression equation • β1: the slope of the linear regression equation • serrLine: the standard error about the line • serrSlope: the standard error of the slope • serrInter: the standard error of the intercept • r: the correlation coefficient • R²: the coefficient of determination Example: LinRegrTTest({1,2,3,4},{3,2,0,-2},0) → {−9.81495457622,5.11086672135ᴇ−3,2,5,−1.7,0.387298334621,0.173205080757,0.474341649025,−0.9897 78266557,0.979661016949}
LinRegrTConfSlope		Linear Regression CI for Slope Syntax: LinRegrTConfSlope(List1, List2, C-value) Linear regression confidence interval for the slope Given a list of explanatory (X) variable data, a list of response (Y) variable data, a value for AltHyp 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 • β1: 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 Example: LinRegrTConfSlope({1,2,3,4},{3,2,0,-2},0.95) → {0.95,4.30265272974,2,−1.7,0.173205080757,−2.44524131352,−0.954758686475}
LinRegrTConfInt		Linear Regression CI for Intercept Syntax: LinRegrTConfInt(List1, List2, C-value) Linear regression confidence interval for the intercept Given a list of explanatory (X) variable data, a list of response (Y) variable data, 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 • β0: 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 Example: LinRegrTConfInt({1,2,3,4},{3,2,0,-2},0.95) → {0.95,4.30265272974,2,5,0.474341649025,2.95907260898,7.04092739101}
LinRegrTMeanResp		Linear Regression CI for Mean Resp. Syntax: LinRegrTMeanResp(List1, List2, X_value, C-value) Linear regression confidence interval for a mean response

Help Topics Tree 13217	Help Text
	Given a list of explanatory (X) variable data, a list of response (Y) variable data, 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 • T: the t-value associated with the confidence level • DF: the degrees of freedom • Ŷ: the mean response for the given X-value • serr Ŷ: the standard error of the mean response • Lower: the lower bound of the confidence interval for the mean response • Upper: the upper bound of the confidence interval for the mean response Example: LinRegrTMeanResp({1,2,3,4},{3,2,0,-2},2.5,0.95) → {2.5,0.95,4.30265272974,2,0.75,0.193649167310,−8.32051183415E−2,1.58320511834}
LinRegrTPredInt	Linear Regression Prediction Interval Syntax: LinRegrTPredInt(List1, List2, X-value, C-value) Linear regression prediction interval for a future response Given a list of explanatory (X) variable data, a list of response (Y) variable data, 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 • T: the t-value associated with the 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 • Lower: the lower bound of the prediction interval for the mean response • Upper: the upper bound of the prediction interval for the mean response Example: LinRegrTPredInt({1,2,3,4},{3,2,0,-2},2.5,0.95) → {2.5,0.95,4.30265272974,2,0.75,0.433012701892,−1.11310328381,2.61310328381}+Q350
AnovaOneWay	ANOVA One-Way Syntax: AnovaOneWay({list1},{list2},[{list3}] ... [{List14}]) Calculates a one-way analysis of variance using up to 14 treatment groups. Returns a list of results containing: • F: the F-value • P: the probability associated with the F-value • DF: the degrees of freedom of the treatments • 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 Example: AnovaOneWay({7,4,6,8,6,6,2,9},{5,5,3,4,4,7,2,2},{2,4,7,1,2,1,5,5}) → {3.59459459459,0.045439700366,2,30.0833333333,15.0416666666,21,87.875,4.18452380952}
Data Streamer app	The Data Streamer app simplifies the collection of data from sensors via the HP StreamSmart 410 data streamer. The Data Streamer app collects the data, then lets you identify the exact data set you wish to send to the Statistics 1Var or 2Var apps for analysis and modeling. Tap Start or press Enter to launch the app. Please refer to the HP StreamSmart 410 User Guide for more information.
Data Streamer Plot View	Press Plot to return to this view at any time. The Plot view is the default view for this app. The Plot view displays up to four data streams, one for each of the active channels on the HP StreamSmart 410. The menu keys are: • CHAN: select a channel (stream) to trace, etc. • PAN/ZOOM: toggle between panning (scrolling) and zooming with direction keys • PAN: scroll up, down, left, and right • ZOOM: zoom in or out vertically or horizontally • SCOPE: switch to oscilloscope mode • START/STOP: stop stream flow or start a new stream With the PAN menu key active, use the rocker wheel up/down to center the active stream in the graphing window. Press the PAN/ZOOM menu key to toggle to ZOOM. Now use the rocker wheel left/right to slow down or speed up the streams, respectively. Both panning and zooming can be done while data is streaming for an interactive experience. Press the STOP menu key and then the EXPORT menu key to isolate the data you want and export it to the Statistics 1Var or 2Var apps for analysis. Press Num to enter the Numeric view of the app.

Help Topics Tree 13217	Help Text
Data Streamer Numeric View	Press Num to return to this view at any time. The Numeric view displays the incoming data numerically instead of graphically. This numerical view is useful for monitoring data numerically, such as during selected events experiments. Menu Buttons: • Add: add a reading to the current data set as specified in Setup • Setup: select experiment type and destination for data • 1s˙: select the sensor refresh rate: 0.5, 1, or 1.5 seconds (1 second is the default) • Stats: go directly to the Statistics application (as specified in Setup) to view and analyze the current data set Tap Setup to define the type of selected events experiment you wish to perform and to select the app and column(s) destination for your final data set(s). In the Setup view, there are fields and menu buttons for defining your selected events experiment, as defined below. Fields: • App: select Statistics 1Var or Statistics 2Var as the destination for your data (other apps will appear if you have saved version of the statistics apps) • Method: select events only or events with entry. If you select events with entry, each time you press the Add menu button in Numeric view, you will be prompted to add a numerical entry for the data point. • Entry: select a column in your chosen statistics app for your entry (Events with Entry experiment method only) • CH1-CH4: select columns in your chosen statistics app for your data set(s) Menu Buttons: • Choose: choose an app, method, column, etc. • ✓: select or deselect a channel for export • Cancel: return to Numeric view without making any changes • OK: save the changes and return to Numeric view
Solve app	The Solve app enables you to define up to ten expressions each with as many variables as you like. You can solve a single expression for one of its variables, based on a seed value. You can also solve a system of equations (linear or non-linear). If two or more of your equations share one or more variables, then the current or solved values of those variables are carried over as you move from one equation to the other. To launch the Solve app, go to the Application Library and tap the Solve app icon. You can also use the rocker wheel to select the Solve app icon, then tap Start or press Enter to launch the app.
Solve Symbolic View	Use this view to enter and edit up to ten equations (or expressions), named E0 to E9. Each equation can use any defined variable (including A to Z and θ). Highlight one of the ten fields and begin entering an equation or expression, or tap Edit to edit an existing expression. The menu buttons are: • Edit: opens an input box to edit the selected definition • ✓: selects a definition for solving • =: a typing aid for entering the equal sign • Show: displays the highlighted equation in full-screen mode, with horizontal and vertical scrolling enabled. • Eval: resolves references when one equation is defined in terms of another • Choose: select a color for the graph Press Num to display the Numeric view. This is where the solving occurs.
Solve Plot View	The Plot View gives you a graphical representation of the selected expression in the Symbolic view, if only one expression is selected. The left and right sides of the current expression are plotted as two separate graphs. The variable that is highlighted in the Numeric View is taken as the independent variable for graphing purposes. The point(s) where these two graphs intersect are solutions to the equation. If there is no 'right side', '=0' is used as an implied right side. The menu buttons are: • Menu: this toggle reveals and hides the Plot menu, with options for zooming and tracing • Zoom: enters the Zoom menu, with options to zoom in or out • Trace: toggles tracing cursor off and on • Go To: takes the tracing cursor to the point on the function with a given x-value • Defn: displays the symbolic definition of the current function If more than one equation is selected in Symbolic view, then Plot view is not available.
Solve Plot Setup	The Solve Plot Setup enables you to control the appearance of the graph window, including the appearance of the cursor, whether or not the axes are drawn, etc. The Setup has two pages. On the first page of the setup, the fields are: • X Rng: the horizontal graphing range • Y Rng: the vertical graphing range • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are: • Edit: edit the value of the selected field • PAGE 1/2 ▼: go to the second page of the setup On the second page of the setup, the fields are:

Help Topics Tree 13217	Help Text
	• Axes: toggle axes on and off • Labels: toggle axis labels on and off • Grid Dots: toggle grid dots on and off • Grid Lines: toggle grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors • Method: choose between Adaptive, Fixed-Step Segments, and Fixed-Step Dots The menu buttons on the second page are: • ✓ : toggle the current setting on or off • Choose: make a choice from a choose box • ▲ PAGE 2/2 : return to the first page of the setup The Method field requires an explanation. By default, the HP Prime uses the Adaptive method, an advanced method that gives very accurate results. You can choose the more traditional method, called Fixed-Step Segments, which samples x-values, computes their corresponding y-values, and then plots and connects the points. Or you can choose Fixed-Step Dots, which works like Fixed-Step Segments but does not connect the points.
Solve Numeric View	The Solve Numeric view is used to enter values for all known variable(s) and then to solve for the unknown(s). If only one equation is checked in the numerical view, enter the values of the known variables, then select the unknown variable and tap Solve. If more than one equation is checked in the numerical view, enter the values of the known variables, check the variables to solve for and tap Solve. Note that the current values for the unknown variable can be used as a seed for the solving algorithm. The menu buttons are: • Edit: edit the current variable's value • Info: get information about a solution • Defn: view the expression that is being solved • Solve: solve for the currently selected variable Note: if Info contains the message Extremum, this indicates that it is highly probable that there is no solution to the equation or system.
Solve App Variables	To display the variables relating to the Solve app, press Vars, tap App and select Solve. The Solve app has variables in the following categories: • Plot (see Common App Variables) • Modes (see Common App Variables) Symbolic View Variables The Solve Symbolic app variables are E1 through E9 and E0. They can contain any expression. The independent variable is selected by highlighting it in the Numeric View. En := expression, where n is an integer between 0 and 9 inclusive Example: E3:='A+2*X=3*B' Example: A+2*X=3*B
SOLVE	Solve App Function Syntax: SOLVE(En,Var[,Guess]) Solves an equation for one of its variables. The argument En may be an equation or expression or it may be the name of one of the Solve Symbolic variables E0-E9. Solves the equation En for the variable Var, using the value of Guess as a seed for the solving algorithm. If En is an expression, then the value of the variable Var that makes the expression equal to zero is returned. Example: SOLVE(X^2-X-2,X,3) → 2 with the Solve app running This function can also return a list containing an information message and one or more numbers. Example: SOLVE(SIN(X)+2, X) returns {Error: Extremum found,−1.57079487496} because sin(x)+2 cannot be equal to zero.
Linear Solver app	The Linear Solver app solves linear systems of 2 equations in 2 variables or 3 equations in 3 variables. To launch the Linear Solver app, go to the Application Library and tap the Linear Solver app icon. You can also use the rocker wheel to select the Linear Solver app icon, then tap Start or press Enter to launch the app.
Linear Solver Numeric View	The Linear Solver opens in the Numeric view, the only view for this app. By default, the app opens ready to solve 3×3 systems of linear equations. Note the dot on the 3×3 menu button to indicate it is active. Tap the 2×2 menu button to switch to solving 2x2 systems of linear equations. Enter the coefficients of each variable in each equation as well as the constant term. The solution to the system appears in real time at the bottom of the screen. The menu buttons are: • Edit: opens a line to edit the chosen value • 2×2: solves a 2×2 system of 2 linear equations with 2 variables • 3×3: solves a 3×3 system of 3 linear equations with 3 variables
Linear Solver Variables	Apart from the modes variables (which are common to all apps), the Linear Solver app has two variables:

Help Topics Tree 13217	Help Text
	• LSystem • LSolution
LSystem	LSystem App Var Contains a 2x3 or 3x4 matrix which represents a 2x2 or 3x3 linear system. matrix ▸ LSystem, where matrix is either a matrix or the name of one of the matrix variables M0-M9.
LSolution	LSolution App Var Contains a vector with the last solution found by either the Linear Solver app or the LSolve app function.
Linear Solver App Functions	This section lists the functions specific to the Linear Solver app.
Solve2×2	Solve2×2 App Function Syntax: Solve2x2(a,b,c,d,e,f) Solves the 2x2 linear system represented by: ax+by=c dx+ey=f Example: Solve2x2(2,-1,5,5,2,8) → {2,-1}
Solve3×3	Solve3×3 App Function Syntax: Solve3x3(a,b,c,d,e,f,g,h,i,j,k,l) Solves the 3x3 linear system represented by: ax+by+cz=d ex+fy+gz=h ix+jy+kz=l Example: Solve3x3(2,1,2,1,4,0,3,-5,0,5,4,13) → {1,5,-3}
LinSolve	LinSolve App Function Syntax: LinSolve(matrix) Solve linear system. Solves the NxN linear system represented by an Nx(N+1) matrix. LinSolve([[a,b,c],[d,e,f]]) solves the linear system: ax+by=c dx+ey=f Examples: LinSolve([[2,-1,5],[5,2,8]]) → {2,-1} M2:=[[1,−3,5,−14],[2,1,−6,20],[3,−2,1,0]]; LinSolve(M2) → {1,0,−3}
Triangle Solver app	A triangle has 3 sides, each of a specific length, as well as 3 angles, each of a specific measure. Specifying 3 of these 6 values fully defines the triangle, as long as one of these is the length of a side. In the ambiguous case, specifying three of the values (including one side length) defines the triangle in terms of two alternatives. The Triangle Solver app allows you to enter 3 known values (one of which must always be a side length) and to calculate the 3 others-or the two alternatives for the 3 others. To launch the Triangle Solver app, go to the Application Library and tap the Triangle Solver app icon. You can also use the rocker wheel to select the Triangle Solver app icon, then tap Start or press Enter to launch the app. The Triangle Solver app opens in the Numeric view, which is the only view this app has.
Triangle Solver Numeric View	Make sure that your angle measure mode is appropriate. If the angle information you have is in degrees and your current angle measure mode is radians or grads, change the mode to degrees by pressing the Radians menu key to toggle it to Degrees. The fields are: • a: the length of one side of the triangle • b: the length of another side • c: the length of the third side • A: the measure of the angle opposite Side a • B: the measure of the angle opposite Side b • C: the measure of the angle opposite Side c The menu buttons in the main Numeric view are: • Edit: opens an edit line to edit the current value of a field • Degrees/Radians: toggles between degrees and radians angle measure • Rect: toggles between a simple solver for right triangles and the general solver • Solve: use the current values to solve for the other unknowns Using the rocker wheel, move to a field whose value you know, enter the value and press Enter. Repeat for each known value. Note that one of the values must be the length of a side. Tap Solve and the app will display the remaining lengths and angle measures. If the Alt menu button appears, it means that there are two possible solutions. Tap Alt to toggle between the two solutions. If you are determining the properties of a right triangle, a simpler input form is available by tapping the menu button with a triangle symbol.
Triangle Solver Variables	Apart from the modes variables (which are common to all apps), the Triangle Solver app variables correspond to the fields in the app's Numeric view.
SideA	SideA App Variable

Help Topics Tree 13217	Help Text
	SideA - The length of the side opposite the angle A. n ▸ SideA, where n>0, sets the value of SideA to n.
SideB	SideB App Variable SideB - The length of the side opposite the angle B. n ▸ SideB, where n>0, sets the value of SideB to n.
SideC	SideC App Variable SideC - The length of the side opposite the angle C. n ▸ SideC, where n>0, sets the value of SideC to n.
AngleA	AngleA App Variable AngleA - The measure of angle A. The value of this variable will be interpreted according to the angle mode setting (Degrees or Radians). n ▸ AngleA, where n>0, sets the value of AngleA to n.
AngleB	AngleB App Variable Angle B- The measure of angle B. The value of this variable will be interpreted according to the angle mode setting (Degrees or Radians). n ▸ AngleB, where n>0, sets the value of AngleB to n.
AngleC	AngleC App Variable AngleC - The measure of angle C. The value of this variable will be interpreted according to the angle mode setting (Degrees or Radians). n ▸ AngleC, where n>0, sets the value of AngleC to n.
TriType	Triangle Type Variable Corresponds to the status of the TriType menu key in the Numeric view of the Triangle Solver app. It determines whether a general triangle solver or a right triangle solver is used. 0 ▸ TriType for the general triangle solver (default) 1 ▸ TriType for the right triangle solver
Triangle Solver App Functions	The Triangle Solver app has a group of functions which allow solving a complete triangle from the input of 3 measures of the triangle. The names of these commands use A to signify an angle, and S to signify a side length. To use these commands, enter 3 inputs in the specified order given by the command name. These commands all return a list of 6 items consisting of the three arguments entered with the command and the three unknown values (lengths of sides and measures of angles).
AAS	AAS App Function Syntax: 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). Example: AAngle:=2; AAS(30,60,1) → {1.73205080757,2,90} (Degrees mode)
ASA	ASA App Function Syntax: 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). Example: AAngle:=2; ASA(30,2,60) → {1,1.73205080757,90} (Degrees mode)
SAS	SAS App Function Syntax: 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. Example: AAngle:=2; SAS(2,60,1) → {1.73205080757,30,90} (Degrees mode)
SSA	SSA App Function Syntax: 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. Example: AAngle:=2; SSA(1,2,30) → {1.73205080757,90,60} (Degrees mode)
SSS	SSS App Function Syntax: SSS(side,side,side)

Help Topics Tree 13217	Help Text
	Takes as arguments the lengths of the three sides of a triangle and returns the measures of the angles opposite them, in order. Example: AAngle:=2; SSS(3,4,5) → {36.8698976458,53.1301023542,90} (Degrees mode)
DoSolve	DoSolve App Function Syntax: 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).
Finance app	The Finance App solves a set of common financial problems. In Symbolic view you can select from a list of common financial problems. You can then enter Numeric view to solve your selected problem. Some of the problems also have a plot view. To launch the Finance app, go to the Application Library and tap the Finance app icon. You can also use the rocker wheel to select the Finance app icon, then tap Start or press Enter to launch the app.
Finance Symbolic View	The Finance Symbolic View allows you to choose which financial calculation you would like to perform in the Finance App. These include: • TVM (Time Value of Money): Used for compound interest calculations that involve regular, uniform cash flows • Interest conversion: Converts between nominal and effective interest rates • Date calculation: Calculates the difference between two dates • Cash flow: Calculates the return on investment and value of cash flows • Depreciation: Calculates the decrease in asset value over time • Break-even: Used to find the break even point between number of units sold, fixed costs, manufacturing costs, sales price, and a desired profit • Percent change: Calculates a new price, cost, or value based on margin, markup, total percentage or percent change • Bond: Calculates bond yield or bond price • Black-Scholes: Uses the Black-Scholes mathematical model to value European call and put options First select your desired calculation from the drop down menu. Then press Num to enter the Finance Numeric View and solve the chosen equation. For a more complete description of how to use each of the finance options see the help text for the Finance Numeric View group.
Finance Amortization Graph	The Finance Amortization Graph displays the amortization schedule graphically. Use the rocker wheel left/right to move from payment group to payment group. For each payment group, the principal and interest paid during the interval are displayed numerically at the bottom of the display.
Finance Plot Setup	Press Shift Plot to enter the Finance Plot Setup. Page 1 of the Plot Setup contains settings that control the appearance of finance plots. On the first page, the fields are: • X Rng: the horizontal range of the graph window • Y Rng: the vertical range of the graph window • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are: • Edit: opens an edit box to edit the value of the selected field • Page 1/2 ▼: displays the second page of the setup On the second page, the fields are: • Axes: toggles axes on and off • Labels: toggles axis labels on and off • Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors The menu buttons on the second page are: • ✓: toggles the current setting on or off • Choose: make a choice from a choose box • ▲ Page 2/2: returns to the first page of the setup
Finance Numeric View	The Finance Numeric View is where you can find solutions to financial problems in the Finance App.
TVM View	Time Value of Money Time Value of Money (TVM) allows you to solve TVM and amortization problems. You can perform compound interest calculations and 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. 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. • N: The total number of compounding periods or payments.

Help Topics Tree	13217	Help Text
		• 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. Enter the values you know, select the quantity that you want to solve for and tap Solve. Tap Amort to display and explore the amortization table for your cash flow. Press Plot to see the amortization graph.
Finance Amortization Table		The Finance Amortization Table displays the amortization schedule. The schedule is a table displaying, for each payment group, the principal and interest paid during the group as well as the balance remaining at the end of the group. • Size: choose between small, medium, and large font size • TVM: returns to the TVM view Use the rocker wheel or drag to scroll through the table. Press the TVM menu key to return to the TVM page when you are done.
Interest Conversion View		Interest Conversion allows you to convert between the nominal interest rate (a rate that is compounded after a given period that must be specified) and the effective interest rate (the amount of interest effectively charged over a year). • Nom I%: Nominal interest rate: the stated annual interest rate compounded as represented by P/Yr, such as 18% compounded monthly (P/Yr=12). • Eff I%: Effective annual interest rate taking compounding into account. • P/YR: Number of periods per year that the nominal interest rate is compounded.
Date Calculation View		Date calculations allows you to calculate the difference between two dates or calculate a date some number of days from another date. • Date 1: The first date in YYYY.MMDD format. This must be a Gregorian date and may not exceed 9999.1231. • Date 2: The second date in YYYY.MMDD format. This must be a Gregorian date and may not exceed 9999.1231. • Difference: The difference between the two dates in number of days (limited to plus or minus 1,000,000 days ~2700years). • Cal. 360: Specifies a 30 day per month, 360 day per year calendar should be used for calculations. The 360 day calendar is useful for measuring durations in financial markets. Enter data in any two of the fields, highlight the remaining field, and tap Solve.
Cash Flow View		Cash Flow is used to solve problems where cash flows occur over regular intervals. Problems with regular, equal, periodic cash flows are handled more easily using the TVM function. In the numeric screen you will see a table where you can enter the Cash Flow data. The top of the table has fields to enter these three items: • Invest I%: Investment or discount interest rate. The rate for cash flows that do not need to be liquid and highly available, so this rate reflects a higher return commensurate with increased risk. • Safe I%: Safe investment interest rate. This rate assumes that funds required to cover negative cash flows are placed in investments that are highly liquid and easy to withdraw at will, making them “safely” available with minimum risk and therefore a lower return. • #CF/Yr: The number of cash flows per year On the lower part of the table you will enter your cash flow data. • CF#: A number that represents the position of the cash flow in the list, where 0 is the initial investment. This number is automatically created as you enter data. • Nb CF: The number of consecutive occurrences of the cash flow. • Cash Flow: The amount of the cash flow. Once you have completed your list tap Calc to see the analysis of your cash flow data. • Internal Rate of Return (IRR): The discount rate that returns a Net Present Value of 0 for the entered cash flows, by discounting all cash flows with Invest I%. • Modified IRR (MIRR): Modified Internal Rate of Return. An improved IRR calculation discounting negative cash flows with Safe I% and positive cash flows with Invest I%. • Financial MRR: (FMRR) Financial Management Rate of Return. A more complicated IRR calculation than MIRR, where negative cash flows are removed by prior positive cash flows before discounting with Safe I% and then subsequent positive cash flows are discounted with Invest I%. • Total: The sum of all the cash flows, equivalent to NPV if Invest I% is 0. • Net Present Value (NPV): Value of cash flows at the time of the initial cash flow, discounting future cash flows by Invest I%. • Net Future Value (NFV): Value of the cash flows at the time of the last cash flow, discounting earlier cash flows by Invest I%. • Net Uniform Series (NUS): Per-period payment of a regular periodic cash flow of equivalent present value to the cash flow list. • Discounted PayBack: The number of periods required for the investment to return value if the cash flows are discounted by Invest I%.

Help Topics Tree 13217	Help Text
	• PayBack: The number of periods required for the investment to return value.
Depreciation View	Depreciation allows you to calculate the loss in value of assets over time. The Type field in the Finance Symbolic view allows you to choose between the following methods for calculating depreciation: • Straight line: Calculates depreciation presuming an asset loses a certain percentage of its value annually at an amount evenly distributed throughout its useful life. • Sum-of-the-years digits: An accelerated depreciation method where the depreciation in year y is (Life-y +1)/SOY of the asset, where SOY is the sum-of-the-years of asset life. For an asset with a 5-year life, SOY=5+4+3+2+1=15. • Declining balance: An accelerated depreciation method that presumes an asset will lose the majority of its value during the first few years of its useful life. • DB with SL cross over: Declining balance with Straight Line cross over is an accelerated depreciation method that presumes an asset will lose the majority of its value in the first few years of its useful life and then revert to a consistent depreciation during the latter part of its life, calculated with the straight line method. • French straight line: Similar to the straight line method, using the actual calendar date the asset was first placed into service. • French amortization: An accelerated depreciation method with a cross over to the French straight line type. Once you have selected the depreciation type you will enter information into the following fields in the Finance Numeric view: • Cost: The starting cost of the asset to be depreciated. • Salvage: The salvage value of the asset at the end of its useful life. • Life: The expected useful life of the asset in years. • First Use: The month (or date for French depreciation types) the asset is first placed into service. Note: month can be entered with a decimal to indicate first use after the first of the month. For example, if the asset was placed into service in the middle of March, enter 3.5. • Factor: The declining balance factor as a percentage. Used for Declining balance and DB with SL cross over types only. After entering data into all of the fields, press Calc to view a calculated table of results beginning with the first year and ending with the last year of useful life. • Depreciation: Depreciation amount for the year. • Depr Value: Remaining depreciable value at the end of the year. • Book Value: Remaining book value at the end of the year.
Break-even View	The break-even function allows you to study problems involving a profit when a quantity of items with a cost to manufacture and a fixed price to develop and market is sold at a given price. This tool solves the equation Fixed + Quantity * Cost = Quantity * Sales + Profit. • Fixed: Fixed cost to develop and market a product. • Quantity: Quantity of units sold. • Cost: Manufacturing or production cost per unit sold. • Price: Price per unit sold. • Profit: Expected profit. Enter the known information into any four of the fields, move the cursor to the value you wish to calculate, and tap Solve.
Percent Change View	Percent change provides two types of percentage calculation tools: Markup / Margin or Percent Total / Percent Change. The Type field in the Finance Symbolic View allows you to choose between the following methods for calculating business percentages: • Markup / Margin: Calculates markup as a percent of cost or margin as a percent of price. • Total / Change: Calculates new value based on total percent of old value or based on percent change from old value Markup / Margin uses the following inputs: • Cost: Total cost to purchase or manufacture the item • Price: Sales price for the item • Markup: A percentage of Cost: ((Price - Cost)/Cost)*100 • Margin: A percentage of Price: ((Price - Cost)/Price)*100 Total / Change uses the following inputs: • Old: The old value for a percent change calculation or the total amount for a part/total calculation. • New: The new value for a percent change calculation or the part of the total for a part/total calculation. • Total: Total Percentage: (New/Old)*100 • Change: Percent Change: ((New-Old)/Old)*100 Enter information into two of the fields, move the cursor to the value you wish to calculate, and tap Solve.
Bond View	Bond allows you to calculate the price or yield of a bond. • Set. Date: Settlement date. The day on which transfer of cash or assets is completed and is usually a few days after the trade was done. Uses format YYYY.MMDD. • Mat. Date: Maturity date or call date. This date always coincides with a coupon date and is the date the bond will be redeemed. Uses format YYYY.MMDD. • Coupon: Coupon rate as an annual percentage. The coupon rate is the fixed annual interest rate paid by the issuer to a bondholder. • Call: Call value. Default is call price per 100.00 face value. A bond at maturity has a call value of 100% of its face value

Help Topics Tree 13217	Help Text
	• Cal. 360: Specifies a 30 day per month, 360 day per year calendar should be used for calculations. The 360 day calendar is useful for measuring durations in financial markets.
	• Semi-annual: Sets payment frequency to be semi-annual instead of annual.
	Enter in all of the known information into all the above fields. Select either Yield or Price and tap Solve.
	• Yield: Yield percent to maturity (or call) date for a given price.
	• Price: Price per 100.00 face value for a given yield percentage.
	The following Results fields are displayed on tapping Solve.
	• Accrued Interest: Interest accrued from the last coupon or payment date until the settlement date for a given yield.
	• Modified Duration: A measure of bond price sensitivity to yield changes, derived from Macaulay duration.
	• Macaulay Duration: A measure of bond price sensitivity to yield changes.
Black-Scholes View	Black-Scholes is a mathematical model useful for valuing European call and put options. Options give the holder the right to buy or sell units of an underlying asset for a period of time at a specified price. A call option is the right to buy and a put option is the right to sell. Specifically, a call option gives the holder of the option the ability to buy a specified number of shares of a stock at a specified price before a certain date, regardless of the actual price of the stock on that date. A put option gives the holder of the option the ability to sell a specified number of shares of a stock at a specified price before a certain date, also regardless of the actual price of the stock on that date.
	For example, assume a call option allows the purchase of 100 shares of a stock at 40.00 per share six months from now. At that six month point, if the stock is worth 50.00, the holder of the option can buy it for 40.00 and earn 10.00 per share immediately. If the stock is worth only 38.00 at that six month point, the option to buy at 40.00 would not be exercised, as it would lose 2.00 per share.
	The Black-Scholes computations assume a European option. This differs from an American option in that a European option can only be exercised at the end of its life, or at its maturity. All other things being equal, the price for an American option will usually be higher than for a European option, since the American option can be traded at any time until its expiration.
	• Stock price: Current underlying asset price, also known as spot price.
	• Strike price: Predetermined price at which the option agrees to buy or sell the underlying asset at maturity, also known as exercise price.
	• Time to maturity: Time remaining until maturity/expiration of the option in years.
	• Risk free%: Current risk-free interest rate (for example, the current US Treasury Bond rate).
	• Volatility %: Degree of unpredictable change of the stock price. This is usually approximated by the standard deviation of the variation of the stock price.
	• Dividend %: Estimation of the average dividend yield of the stock as a percentage of its price.
	Enter in values for all of the fields. Once they are all entered, tap Solve to calculate Call price and Put price.
	• Call price: Estimated fair market value for a call option at expiration (a call option is the right to purchase the asset at a given price).
	• Put price: Estimated fair market value for a put option at expiration (a put option is the right to sell the asset at a given price).
Finance Numeric View	The Finance Symbolic View allows you to choose which financial calculation you would like to perform in the Finance App. These include:
	• TVM (Time Value of Money): Used for compound interest calculations that involve regular, uniform cash flows
	• Interest conversion: Converts between nominal and effective interest rates
	• Date calculation: Calculates the difference between two dates
	• Cash flow: Calculates the return on investment and value of cash flows
	• Depreciation: Calculates the loss in value of assets over time
	• Break-even: Used to find the break-even point between number of units sold, fixed costs, manufacturing costs, sales price, and a desired profit
	• Percent change: Calculates a change based on a percentage
	• Bond: Calculates the yield or price of a bond
	• Black-Scholes: Uses the Black-Scholes equation to value investments
	First select your desired calculation from the drop down menu. Then press Num to enter the Finance Numeric View and solve the chosen equation.
	For a more complete description of how to use each of the finance options, see the help text for the Finance Numeric View group.
Finance App Variables	Apart from the modes variables (which are common to all apps), the Finance app variables correspond to the fields in the Finance app Numeric view.
Symbolic Variables	There are two Finance App Symbolic Variables, each of which corresponds to one of the two possible fields in the Symbolic view of the Finance app:
	• Method
	• FinType
Method	Method determines the current calculation type in the Finance app.
	• Method := 0 for TVM
	• Method := 1 for interest conversion
	• Method := 2 for date calculation
	• Method := 3 for cash flow
	• Method := 4 for depreciation
	• Method := 5 for break-even
	• Method := 6 for percent change

Help Topics Tree		13217	Help Text
			• Method := 7 for bond • Method := 8 for Black-Scholes
		FinType	FinType determines the type of calculation for depreciation or percent calculations. With Method=4 for depreciation, the constant values and their meanings are as follows: • FinType:= 0 for straight line • FinType:= 1 for sum-of-the-years digits • FinType:= 2 for declining balance • FinType:= 3 for declining balance with crossover • FinType:= 4 for French straight line • FinType:= 5 for French amortization With Method=6 for percent calculations, the constant values and their meanings are as follows: • FinType:= 0 for margin/markup • FinType:= 1 for percent/change
	TVM Variables		After a TVM (Time Value of Money) calculation is performed in the Finance app, the values are stored in the TVM Variables.
		NbPmt	NbPmt App Variable NbPmt - The number of payments in an investment or loan. n ▸ NbPmt, where n>0, sets the value of NbPmt to n.
		IPYR	IPYR App Variable IPYR - The interest rate per year of an investment or loan. n ▸ IPYR sets the value of IPYR to n.
		PV	PV App Variable PV - The present value of an investment or loan. n ▸ PV sets the value of PV to n.
		PMT	PMT App Variable PMT - The value of a payment for an investment or loan. n ▸ PMT sets the value of PMT to n.
		FV	FV App Variable FV - The future value of an investment or loan. n ▸ FV sets the value of FV to n.
		PPYR	PPYR App Variable PPYR - The number of payments made per year for an investment or loan. n ▸ PPYR, where n>0, sets the value of PPYR to n.
		CPYR	CPYR App Variable CPYR - The number of compounding periods per year for an investment or loan. The default value is 12. n ▸ CPYR, where n>0, sets the value of CPYR to n.
		BEG	BEG App Variable BEG determines whether interest is compounded at the beginning or end of the compounding period. 0 ▸ BEG for compounding at the end of the period (default) 1 ▸ BEG for compounding at the beginning of the period
		GSize	GSize App Variable GSize - The size of each group for the amortization table. The default value is 12. n ▸ GSize, where n>0, sets the value of GSize to n.
	Interest Conversion Variables		After an Interest Conversion calculation is performed in the Finance app, the values are stored in the Interest Conversion Variables.
		NomInt	NomInt - The nominal interest rate. n ▸ NomInt , where 0≤n≤100, sets the value of NomInt to n
		EffInt	EffInt - The effective interest rate. n ▸ EffInt , where 0 ≤ n ≤ 100, sets the value of EffInt to n
		IntCPYR	IntCPYR The number of times interest compounds per year n ▸ IntCPYR, where n>0, sets the value of IntCPYR to n.
	Date Calculation Variables		After a Date Calculation is performed, the data is stored in the Date Calculation variables.
		DateOne	DateOne - The first date used in a date calculation. Uses the format YYYY.MMDD. n ▸ DateOne, where n is YYYY.MMDD, sets the value of DateOne to n
		DateTwo	DateTwo - The second date used in a date calculation. Uses the format YYYY.MMDD n ▸ DateTwo, where n is YYYY.MMDD, sets the value of DateTwo to n
		DateDiff	DateDiff - The difference between the two dates. n ▸ DateDiff, where n>0, sets the value of DateDiff to n
		Date360	Date360 - Determines whether to use a standard Gregorian or 360-day year when doing a date calculation. 0 ▸ Date360 for standard 365-day year 1 ▸ Date360 for 360-day year
	Cash Flow Variables		When Cash Flow calculations are performed, the data is stored in the Cash Flow Variables.
		CFData	Syntax: CFData CFData(n)

Help Topics Tree	13217	Help Text

		CFData(n,option)
		CFData:={cash_flow1, cash_flow2, ... cash_flowN}
		CFData:=[cash_flow1, cash_flow2, ... cash_flowN]
		CFData:={{cash_flow1, count1},{cash_flow2,count2}, ... {cash_flowN,countN}}
		CFData:=[[cash_flow1, count1],[cash_flow2,count2], ... [cash_flowN,countN]]
		CFData(n):=cash_flow
		CFData(n):={cash_flow, count}
		CFData(n):=[cash_flow, count]
		CFData provides access to the cash flow information and is a list of lists. Each sublist contains cash flow and count. Count defaults to 1 if not specified.
		CFData(n) references the cashflow and count pair numbered n. The initial cashflow is number 0.
		
		CFData(n,option) references either the cash flow or the count of the nth pair, depending on the value of option. 1 is cash flow, 2 is count.
		An entire list of lists or matrix representing the cash flow information can be stored in a single operation.
		
		Examples:
		CFData:={-100,60,60}
		CFData:=[-100,60,60]
		CFData:={{-100,1},{60,2}}
		CFData:=[[-100,1],[60,2]]
		CFData(0)
		CFData(0,1)

	InvestInt	InvestInt - The cash flow investment interest rate. n ▶ InvestInt , where 0 ≤ n ≤100, sets the value of InvestInt to n
	SafeInt	SafeInt - The cash flow safe interest rate. n ▶ SafeInt , where 0 ≤ n ≤100, sets the value of SafeInt to n
	CFPYR	CFPYR - The number of cash flows per year. n ▶ CFPYR , where 1 ≤ n ≤ 12, sets the value of CFPYR to n
	IRR	IRR - The Internal Rate of Return of a cash flow.
	MIRR	MIRR - The Modified Internal Rate of Return of a cash flow.
	FMRR	FMRR stores Financial Management Rate of Return of a cash flow.
	TotalCF	TotalCF - The cash flow total..
	NPV	NPV - Net Present Value of a cash flow.
	NFV	NFV - Net Future Value of a cash flow.
	NUS	NUS - Net Uniform Series of a cash flow.
	DiscPayback	DiscPayback - The Discounted Payback period of a cash flow.
	Payback	Payback - The Payback period of a cash flow.
Depreciation Variables		After a Depreciation calculation is performed in the Finance App, the values are stored in the Depreciation Variables.
	CostAsset	CostAsset - The depreciable cost of an asset at time of purchase. n ▶ CostAsset, where n>0, sets the value of CostAsset to n
	SalvageAsset	SalvageAsset - The amount of money an asset can be sold or salvaged for at the end of its life. n ▶ SalvageAsset, where n>0, sets the value of SalvageAsset to n
	FirstAsset	FirstAsset - The month the asset is first placed into service. Normally, this will be 1. A decimal amount inidicates a partial month. n ▶ FirstAsset, where n≥1, sets the value of FirstAsset to n
	LifeAsset	LifeAsset - The expected useful life of a product. n ▶ LifeAsset, where n≥1, sets the value of LifeAsset to n.
	FactorDepr	FactorDepr - The depreciation factor as a percentage, used with the declining balance method. n ▶ FactorDepr, where n>0, sets the value of FactorDepr to n
	FirstDateAsset	FirstDateAsset - The date of first use for French style Depreciation, entered as YYYY.MMDD n ▶ FirstDateAsset, where n is YYYY.MMDD, sets the value of FirstDateAsset to n
Break-even Variables		After a Break-even calculation is performed in the Finance App, the values are stored in the Break-even Variables.
	FixedCost	FixedCost - The fixed cost of developing and marketing a product. n ▶ FixedCost, where n>0, sets the value of FixedCost to n.
	VariableCost	VariableCost - The manufacturing cost per unit. n ▶ VariableCost, where n>0, sets the value of VariableCost to n
	SalePrice	SalePrice - The sales price per unit. n ▶ SalePrice, where n>0, sets the value of SalePrice to n.
	Profit	Profit - The expected profit. n ▶ Profit, where n>0, sets the value of Profit to n.
	Quantity	Quantity - The number of units sold. n ▶ Quantity, where n>0, sets the value of Quantity to n.
Percent Change Variables		After Percent Change calculations are performed, the data is stored in the Percent Change variables.
	Price	Price - The sales price in markup calculations. n ▶ Price sets the value of Price to n
	Cost	Cost - The cost of an item in markup calculations.

Help Topics Tree			13217		Help Text
					n ▶ Cost sets the value of Cost to n
				Margin	Margin - The margin in markup calculations based on cost. n ▶ Margin sets the value of Margin to n
				Markup	Markup - The markup percentage in markup calculations. n ▶ Markup sets the value of Markup to n
				OldValue	OldValue - The old value in percent-change calculations and the total in part-total calculations. n ▶ OldValue sets the value of OldValue to n
				NewValue	NewValue - The new value in percent-change calculations and the part number in part-total calculations. n ▶ NewValue sets the value of NewValue to n
				Total	Total - The percentage of the total in part-total calculations. n ▶ Total sets the value of Total to n
				Change	Change - The percent change in percent-change calculations. n ▶ Change sets the value of Change to n
			Bond Variables		After a Bond calculation is performed in the Finance App, the values are stored in the Bond Variables.
				SetDate	SetDate - The settlement date of a bond. Dates should be entered as YYYY.MMDD n ▶ SetDate, where n is YYYY.MMDD, sets the value of SetDate to n
				MatDate	MatDate - The maturity date or call date of a bond. Dates should be entered as YYYY.MMDD n ▶ MatDate, where n is YYYY.MMDD, sets the value of MatDate to n
				CpnPer	CpnPer - The coupon percentage. n ▶ CpnPer sets the value of CpnPer to n
				CallPrice	CallPrice - The call price or value. n ▶ CallPrice sets the value of CallPrice to n
				YieldBond	YieldBond - The yield percent to maturity of a bond. n ▶ YieldBond sets the value of YieldBond to n
				PriceBond	PriceBond - The price per 100.00 value of a bond. n ▶ PriceBond sets the value of PriceBond to n
				Accrued	Accrued - The accrued interest of a bond. n ▶ Accrued sets the value of Accrued to n
				Modified	Modified - The modified duration of a bond. n ▶ Modified sets the value of Modified to n
				Macaulay	Macaulay - The Macaulay duration of a bond. n ▶ Macaulay sets the value of Macaulay to n
				Bond360	Bond360 - Determines whether to use a standard Gregorian or a 360-day calendar . 0 ▶ Bond360 for standard 365-day year 1 ▶ Bond360 for 360-day year
				SemiAnnual	SemiAnnual - Determines whether payments are made on an annual or semi-annual basis. 0 ▶ SemiAnnual indicates annual payments 1 ▶ SemiAnnual indicates semi-annual payments
			Black-Scholes Variables		After a Black-Scholes calculation is performed, the values are stored in the Black-Scholes variables.
				StockPrice	StockPrice - The stock price. This is the current underlying asset price, also known as spot price. n ▶ StockPrice sets the value of StockPrice to n
				StrikePrice	StrikePrice - The strike price. This is the predetermined price at which the option agrees to buy or sell the underlying asset at maturity, also known as exercise price. n ▶ StrikePrice sets the value of StrikePrice to n
				TimeMarket	TimeMarket - The time to market of an option. n ▶ TimeMarket sets the value of TimeMarket to n
				RiskFree	RiskFree - The risk free interest rate. n ▶ RiskFree sets the value of RiskFree to n
				Volatility	Volatility - The volatility of an asset. n ▶ Volatility sets the value of Volatility to n
				Dividend	Dividend - The dividend percentage. n ▶ Dividend sets the value of Dividend to n
				BSCall	BSCall - The call price of an option. n ▶ BSCall sets the value of BSCall to n
				BSPut	BSPut - The put price of an option. n ▶ BSPut sets the value of BSPut to n
		Finance App Functions			The functions specific to the Finance app are listed in this section.
			TVM Functions		The functions specific to time value of money are listed in this section.
				TvmNbPmt	TvmNbPmt App Function Syntax: TvmNbPmt(IPYR, PV, PMT, FV, [PPYR], [CPYR], [BEG]) Solves for the number of payments in an investment or loan. • IPYR: the annual interest rate • PV: the present value of the investment or loan • PMTV: the payment value

Help Topics Tree	13217	Help Text
		• FV: the future value of the investment or loan • PPYR: the number of payments per year • CPYR: the number of compounding periods per year • BEG: payments made at the beginning (1) or end (0) of the period The arguments PPYR, CPYR, and BEG are optional; if not supplied, PPYR=12, CPYR=PPYR, and BEG=0. Example: TvmNbPmt(6.5,150000,-948.10,-2.25) → 360
	TvmIPYR	TvmIPYR App Function Syntax: TvmIPYR(NbPmt, PV, PMT, FV, [PPYR], [CPYR], [BEG]) Solves for the interest rate per year of an investment or loan. • NbPmt: the number of payments • PV: the present value of the investment or loan • PMT: the payment value • FV: the future value of the investment or loan • PPYR: the number of payments per year • CPYR: the number of compounding periods per year • BEG: payments made at the beginning (1) or end (0) of the period The arguments PPYR, CPYR, and BEG are optional; if not supplied, PPYR=12, CPYR=PPYR, and BEG=0. Example: TvmIPYR(360,150000,-948.10,-2.25) → 6.50
	TvmPV	TvmPV App Function Syntax: TvmPV(NbPmt, IPYR, PMT, FV, [PPYR], [CPYR], [BEG]) Solves for the present value of an investment or loan. • NbPmt: the number of payments • IPYR: the annual interest rate • PMTV: the payment value • FV: the future value of the investment or loan • PPYR: the number of payments per year • CPYR: the number of compounding periods per year • BEG: payments made at the beginning (1) or end (0) of the period The arguments PPYR, CPYR, and BEG are optional; if not supplied, PPYR=12, CPYR=PPYR, and BEG=0. Example: TvmPV(360,6.5,-948.10,-2.25) → 150000.00
	TvmPMT	TvmPMT App Function Syntax: TvmPMT(NbPmt, IPYR, PV, FV, [PPYR], [CPYR], [BEG]) Solves for the value of a payment for an investment or loan. • NbPmt: the number of payments • IPYR: the annual interest rate • PV: the present value of the investment or loan • FV: the future value of the investment or loan • PPYR: the number of payments per year • CPYR: the number of compounding periods per year • BEG: payments made at the beginning (1) or end (0) of the period The arguments PPYR, CPYR, and BEG are optional; if not supplied, PPYR=12, CPYR=PPYR, and BEG=0. Example: TvmPMT(360,6.5,150000,-2.25) → -948.10
	TvmFV	TvmFV App Function Syntax: TvmFV(NbPmt, IPYR, PV, PMT, [PPYR], [CPYR], [BEG]) Solves for the future value of an investment or loan. • NbPmt: the number of payments • IPYR: the annual interest rate • PV: the present value of the investment or loan • PMT: the payment value • PPYR: the number of payments per year • CPYR: the number of compounding periods per year • BEG: payments made at the beginning (1) or end (0) of the period The arguments PPYR, CPYR, and BEG are optional; if not supplied, PPYR=12, CPYR=PPYR, and BEG=0. Example: TvmFV(360,6.5,150000,-948.10) → -2.25
Interest Conversion Functions		The functions specific to interest conversion are listed in this section.
	IntConvNom	Syntax: IntConvNom(effective_rate,compounds_per_year)

Help Topics Tree	13217	Help Text
		IntConvNom returns the nominal interest rate in an Interest Conversion calculation when given the effective_rate and the number of compounds_per_year. Example: IntConvNom(6.86,12) → 6.65
	IntConvEff	Syntax: IntConvEff(nominal_rate,compounds_per_year) IntConvEff returns the effective interest rate in an Interest Conversion calculation when given the nominal_rate and the number of compounds_per_year. Example: IntConvEff(6.65,12) → 6.86
	IntConvCPYR	Syntax: IntConvCPYR(nominal_rate,effective_rate) IntConvCPYR returns the number of compounding periods in a year in an Interest Conversion calculation when given the nominal_rate and the effective_rate. Example: IntConvCPYR(6.65,6.86) → 14.64
DateDays		Syntax: DateDays(first_date,second_date,[cal_360]) DateDays returns the difference between two days when given two dates (first_date and second_date as YYYY.MMDD). Optionally, a 1 in the third field, cal_360, will specify that a 360-day calendar (twelve 30 day months) should be used. Examples: DateDays(2013.1213,2016.0202) → 781 DateDays(2013.1213,2016.0202,1) → 769
	Cash Flow Functions	The functions specific to cash flow are listed in this section.
	CashFlowIRR	Syntax: CashFlowIRR(cash_flow_data,[cashflows_per_year]) CashFlowIRR returns the Internal Rate of Return for cash_flow_data. cashflows_per_year specifies the number of cash flows per year. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. These are examples of valid input forms: {cash_flow1, cash_flow2, ... cash_flowN} [cash_flow1, cash_flow2, ... cash_flowN] If you wish to specify the count of a cash flow, the cash flow should come first followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1,{cash_flow2,count2}, ... {cash_flowN,countN}} [[cash_flow1, count1],cash_flow2, ... [cash_flowN,countN]] Example: CashFlowIRR({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}) → 3.72
	CashFlowMIRR	Syntax: CashFlowMIRR(cash_flow_data, investment_rate, safe_investment_rate, [cashflows_per_year]) CashFlowMIRR returns the Modified Internal Rate of Return for cash_flow_data, investment_rate and safe_investment_rate. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Example: CashFlowMIRR({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 8, 5, 1) → 5.12
	CashFlowFMRR	Syntax: CashFlowFMRR(cash_flow_data, investment_rate, safe_investment_rate, [cashflows_per_year]) CashFlowFMRR returns the Financial Management Rate of Return for cash_flow_data, investment_rate and safe_investment_rate. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Example: CashFlowFMRR({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 8, 5, 1) → 4.98
	CashFlowTotal	Syntax: CashFlowTotal(cash_flow_data) CashFlowTotal calculates the total of all inputs for for cash_flow_data. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms:

Help Topics Tree	13217	Help Text
		{cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Example: CashFlowTotal({-1250000,-300000,{200000,3},-200000,700000,300000,500000}) → 350000
	CashFlowNPV	Syntax: CashFlowNPV(cash_flow_data, investment_rate, [cashflows_per_year]) CashFlowNPV calculates the Net Present Value for cash_flow_data and investment_rate. cashflows_per_year specifies the number of cash flows per year. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Example: CashFlowNPV({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 8, 1) → −300353.93
	CashFlowNFV	Syntax: CashFlowNFV(cash_flow_data, investment_rate, [cashflows_per_year]) CashFlowNFV calculates the Net Future Value for cash_flow_data and investment_rate. cashflows_per_year specifies the number of cash flows per year. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Example: CashFlowNFV({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 8, 1) → −555934.17
	CashFlowNUS	Syntax: CashFlowNUS(cash_flow_data, investment_rate, [cashflows_per_year]) CashFlowNUS calculates the Net Uniform Series for cash_flow_data and investment_rate. cashflows_per_year specifies the number of cash flows per year. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Example: CashFlowNUS({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 8, 1) → -52266.02
	CashFlowPB	Syntax: CashFlowPB(cash_flow_data, [investment_rate]) CashFlowPB(cash_flow_data, [investment_rate, cashflows_per_year]) CashFlowPB calculates the Discounted Pay Back period for cash_flow_data and investment_rate. investment_rate specifies the investment rate for the discounting. A value of 0 for investment_rate will calculate the Pay Back with no discounting. cashflows_per_year specifies the number of cash flows per year. If cashflows_per_year is not provided, then it is assumed to be 1. Enter cash_flow_data as a list or matrix. To indicate the same cash flow repeats more than once, enter the cash flow as list or matrix with the cash flow followed by the count. If you do not specify a count, then count will be assumed to be 1. These are examples of valid input forms: {cash_flow1, {cash_flow2, count2}, ... {cash_flowN, countN}} [[cash_flow1, count1], cash_flow2, ... [cash_flowN, countN]] Examples: CashFlowPB({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 8) → Error: No payback CashFlowPB({-1250000, -300000, {200000, 3}, -200000, 700000, 300000, 500000}, 0, 1) → 7.30
Depreciate		Syntax: Depreciate(method, cost, salvage, life, [first], [factor]) Depreciate returns the depreciation schedule when given the method of calculation, the depreciable cost at the time of purchase, the expected return amount from the salvage sale of the asset, the expected life in years, the moment of first use, and the factor of depreciation as a percentage. The moment of first use is expressed as a number corresponding to the month and fractional part of the month (example: 2.5 = 1/2 of the month of February) for methods 0 to 3 and as an actual date (in yyyy.mmdd format) for the methods 4 and 5. To input the method use the following numbers: 0: Straight Line 1: Sum Of Year Digits 2: Declining Balance

Help Topics Tree	13217	Help Text
		3: Declining Balance with Straight line crossover 4: French Straight Line 5: French Amortization The depreciation schedule is returned as a list of lists, where the list number corresponds to the depreciation year. {{Depreciation_Year_1,Depreciable_Value_Year_1,Book_Value_Year_1}, {Depreciation_Year_2,Depreciable_Value_Year_2,Book_Value_Year_2}, … {Depreciation_Year_n,Depreciable_Value_Year_n,Book_Value_Year_n}} Example: Depreciate(0,10000,500,2) → {{4750,4750,5250},{4750,0,500}}
Break-Even Functions		The functions specific to break-even are listed in this section.
	BrkEvFixed	Syntax: BrkEvFixed(quantity, cost, price, profit) BrkEvFixed returns the fixed cost to develop and market a product in a break-even calculation when given the cost per unit sold, the sales price per unit, the expected profit, and the quantity of units sold. Example: BrkEvFixed(3200,250,300,10000) → 150000
	BrkEvCost	Syntax: BrkEvCost (fixed_cost, quantity, price, profit) BrkEvCost returns the cost per unit in a break-even calculation when given the fixed_cost of marketing and development, the sales price per unit, the expected profit, and the quantity of units sold. Example: BrkEvCost(150000,3200,300,10000) → 250
	BrkEvPrice	Syntax: BrkEvPrice(fixed_cost, quantity, cost, profit) BrkEvPrice returns the unit price in a break-even calculation when given the fixed_cost of marketing and development, the manufacturing cost per unit, the expected profit, and the quantity of units sold. Example: BrkEvPrice(150000,3200,250,10000) → 300
	BrkEvProfit	Syntax: BrkEvProfit(fixed_cost, quantity, cost, price) BrkEvProfit returns the profit in a break-even calculation when given the fixed_cost, cost of manufacturing for each unit, price of each unit, and the quantity of units sold. Example: BrkEvProfit(150000,3200,250,300) → 10000
	BrkEvQuant	Syntax: BrkEvQuant(fixed_cost,cost,price,profit) BrkEvQuant returns the quantity of units sold in a break-even calculation when given the fixed_cost of marketing and development, the manufacturing cost per unit, the sales price per unit, and the expected profit. Example: BrkEvQuant(150000,250,300,10000) → 3200
Percent Change Functions		The functions specific to percent change are listed in this section.
	ChangePrice	Syntax: ChangePrice(cost,percentage,option) ChangePrice returns the sales price of an item in a markup calculation when given the cost and percentage. If percentage is a markup, use 0 for option. For margin percentage, set option to 1. Examples: ChangePrice(35,14.29,0) → 40 ChangePrice(35,12.5,1) → 40
	ChangeCost	Syntax: ChangeCost(price,percentage,option) ChangeCost returns the cost of an item in a markup calculation when given the sales price and percentage. If percentage is a markup, use 0 for option. For margin percentage, set option to 1. Examples: ChangeCost(40,14.29,0) → 35 ChangeCost(40,12.5,1) → 35
	PercentMargin	Syntax: PercentMargin(cost,price) PercentMargin returns the markup percentage as a percentage of cost, or the margin, in markup calculations when given the sales price and the cost of the item. Example: PercentMargin(100,125) → 25
	PercentMarkup	Syntax: PercentMarkup(cost,price) PercentMarkup returns the markup as a percentage of price in markup calculations when given the sales price and cost of an item. Example: PercentMarkup(100,125) → 20

Help Topics Tree				13217	Help Text
				ChangeOld	Syntax: ChangeOld(new,percentage,option) ChangeOld returns the old number in a percent change calculation when given the new number and percentage. When option is 0, percentage is a total percentage value and ChangeOld will use a part-total calculation. (new / (percentage / 100)) When option is 1, percentage is a percent change value andChangeOld will use a percent change calculation. (new / (1 + percentage / 100)) Examples: ChangeOld(50,25,0) → 200 ChangeOld(50,25,1) → 40
				ChangeNew	Syntax: ChangeNew(old,percentage,option) ChangeNew returns the new number in a percent change calculation when given the old number and percentage, When option is 0, percentage is a total percentage value andChangeNew will perform a part-total calculation. (old * (percentage/100)) When option is 1, percentage is a percent change value and ChangeNew will perform a percent change calculation. (old * (1 + percentage/100)) Examples: ChangeNew(120,25,0) → 30 ChangeNew(120,25,1) → 150
				PercentTotal	Syntax: PercentTotal(old,new) PercentTotal calculates the part-total percentage when given the old and new numbers. Example: PercentTotal(65,25) → 38.46
				PercentChange	Syntax: PercentChange(old,new) PercentChange calcuates the percent change when given the old and new number. Example: PercentChange(65,25) → −61.54
			Bond Functions		The functions specific to bonds are listed in this section.
				BondYield	Syntax: BondYield(settlement_date,maturity_date,price,coupon_percent,call_value,semi_annual,cal360) BondYield returns the yield percent to maturity (or call) date when given settlement_date (YYYY.MMDD), maturity_date (YYYY.MMDD), price per 100.00 face value, coupon_percent, and call_value. The last two parameters specify whether the payments are made on an annual or semi_annual basis (enter 0 for annual and 1 for semi_annual) and whether to use an actual or cal360 calendar (0 for actual and 1 for cal360). Example: BondYield(2010.0428,2020.0604,6.75,100,115.74,1,0) → 4.77
				BondPrice	Syntax: BondPrice(settlement_date,maturity_date,coupon_percent,call_value,yield_percent,semi_annual,cal360) BondPrice returns the price per 100.00 face value when given the settlement_date (YYYY.MMDD), the maturity_date (YYYY.MMDD) or call date, the yield_percent, the coupon_percent, and the call_value. The last two numbers specify whether the payments are made on an annual or semi_annual basis (enter 0 for annual and 1 for semi_annual) and whether to use an actual or cal360 calendar (0 for actual and 1 for cal360). Example: BondPrice(2010.0428,2020.0604,6.75,100,4.77,1,0) → 115.72
			BlackScholes		Syntax: BlackScholes(stock_price,strike_price,time_to_maturity,risk_free_interest_rate,stock_volatility,stock_dividend) BlackScholes returns the Call price and Put price for options when given the stock_price, the strike_price, the time_to_maturity, the risk_free_interest_rate, the stock_volatility, and the stock_dividend percentage. Example: BlackScholes(74,72,5,0.3,8.21,2.73) → {2.40,8.77}
	Explorer app				The Explorer app is designed to explore the relationships between the parameters of a function and the shape of the graph of the function. There are two ways of exploring in Plot view. You manipulate a graph and note the corresponding changes in its equation, or you can edit the parameters in an equation and note the corresponding changes in its graphical representation. The app also has a test mode. In test mode, you test your skill at matching equations to graphs. Press the Apps key to open the Apps library, and then select Explorer.
		Explorer Symbolic View			The app opens in Symbolic view, where you select the function family you would like to explore. The explorer app supports the following function families: • Linear • Quadratic • Cubic

Help Topics Tree 13217	Help Text
	• Exponential • Logarithmic • Sinusoidal Tap on the field and tap Choose to select a function family to explore.
Explorer Plot View	Plot view displays an equation along with its graph. The equation and graph depend on the choice of function family made in Symbolic view. Depending on the function family, Plot view may also display numerical values associated with the graph, such as the intercepts, etc. There are often multiple types (or levels) of equation available for you to explore. You choose between them by tapping the menu key labeled Lev 1 or Lev 2, and so on. Tap and drag the graph to translate it. The equation updates automatically. Pinch to dilate vertically or horizontally. Again, the equation updates automatically. The original graph is shown dotted for comparison purposes. The form of the equation 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. You can also tap on the equation and edit the equation parameters directly. Press Enter or tap OK to see the graph update. In Plot view, the common menu keys are: • Lev n: toggles between various forms of the selected function family • Test: enters Test mode
Explorer App Functions	The functions specific to the Eplorer app are listed in this section.
LinearSlope	Solve For Slope Syntax: LinearSlope(x1,y1,x2,y2) Solve for slope. Takes as input the coordinates of two points (x1,y1) and (x2,y2) and returns the slope of the line containing those two points. Example: LinearSlope(3,4,2,2) → 2
LinearYIntercept	Solve for Y Intercept Syntax: LinearYIntercept(x, y, m) 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. Example: LinearYIntercept(2,3,-1) → 5
QuadSolve	Solve quadratic Syntax: QuadSolve(a, b, c) Given the coefficients of a quadratic equation a*x^2+b*x+c=0, returns the real solutions. Example: QuadSolve(1,0,-4) → {-2, 2}
QuadDelta	Solve discriminant Syntax: QuadDelta(a, b, c) Given the coefficients of a quadratic equation a*x^2+b*x+c=0, returns the value of the discriminant in the Quadratic Formula. Example: QuadDelta(1,0,-4) → 16
Parametric app	The Parametric app allows you to explore the simultaneous variation of two variables, each of which depends on a parameter T. The values of these two equations when T varies are treated as the x and y coordinates of a point which is displayed in the Cartesian plane. These equations are displayed in the symbolic view in the form X=f(T) and Y=g(T). Once you have defined a pair of parametric equations, you can plot the graph or explore a table of values for the equations. To launch the Parametric app, go to the Application Library and tap the Parametric app icon. You can also use the rocker wheel to select the Parametric app icon, then tap Start or press Enter to launch the app.
Parametric Symbolic View	The Parametric Symbolic view contains definitions for up to ten parametric equations, each one defining X=f(T) and Y=g(T). The menu buttons are: • Edit: opens an input box to edit the selected parametric definition • ✓: toggles the selected definition on or off for plotting and table-building • T: a typing aid for entering definitions in terms of T • Show: displays the selected definition in full-screen mode with horizontal and vertical scrolling enabled • Eval: resolves references to other equations, such as X2(T)=X1(T)/5 • Choose: select a color for the graph Highlight one of the definition fields and begin entering an expression in T, or tap Edit to open an edit line to edit an existing expression.
Parametric Plot View	Press Plot to enter the Parametric Plot view. This view displays the graphs of parametric equations defined and checked in Symbolic view. The functionality here is the same as in the Function Plot view, except that the Fcn functions do not apply here. Tap Menu to open the menu. The menu buttons are: • Zoom: enters the Zoom menu, with options to zoom in or out, etc.

Help Topics Tree 13217	Help Text
	• Trace: toggles the tracing cursor off and on • Go To: takes the tracing cursor to the point on the graph with a given value of T. • Defn: displays the symbolic definition of the current graph • Menu: toggles the menu off and on Use the rocker wheel left/right or tap to trace along a graph. Use the rocker wheel up/down to switch from one graph to another. Press + to zoom in on the current cursor location and press - to zoom out. Set the zoom factor under the Zoom menu. You can also use all the gestures common to the Plot views. See Plot View for more details.
Parametric Plot Setup	Press Shift Plot to enter the Parametric Plot setup. This view enables you to control the appearance of the graph window, including the appearance of the cursor, whether or not the axes are drawn, etc. The Setup has two pages. On the first page, the fields are: • T Rng: the range of values for the parameter T • T Step: the step value for the parameter T • X Rng: the horizontal graphing range • Y Rng: the vertical graphing range • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are: • Edit: opens an edit line to edit the value of the selected field • Page 1/2 ▼: displays the second page of the setup Tap Page 1/2 ▼ to view the second page of the setup. Here the fields are: • Axes: toggles axes on and off • Labels: toggles axis labels on and off • Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors • Method: choose between Adaptive, Fixed-Step Segments, and Fixed-Step Dots The menu buttons on the second page are: • ✓: toggles the current setting on or off • Choose: make a choice from a choose box • ▲ Page 2/2: returns to the first page of the setup The Method field requires an explanation. By default, the Prime uses the Adaptive method, an advanced method that gives very accurate results. You can choose the more traditional method, called Fixed-Step Segments, which samples x-values, computes their corresponding y-values, and then plots and connects the points. Or you can choose Fixed-Step Dots, which works like Fixed-Step Segments but does not connect the points.
Parametric Numeric View	Press Num to enter the Parametric Numeric View. The Parametric Numeric View is designed to create and explore a table of X/Y/T values, based on the function(s) defined in the Symbolic View. Place the highlight bar in any row of the T-column, enter any real value, and tap OK. The table will reconfigure. You can also zoom in or out on any row in the table. Press + to zoom in on a row of the table and - to zoom out. The menu buttons are: • Zoom: zooms in or out on a highlighted row of the table. Note that in Numeric view, zooming changes the increment between consecutive x-values. Zooming in decreases the increment; zooming out increases the increment. The values in the row you zoom in or out on remain the same. • More: opens a menu with editing options • Go To: jumps to a specified value of the independent variable • Defn: displays the definition of the selected column The More menu The More menu contains the following options: • Select • Row: selects the row that contains the currently selected cell; the row can then be copied to paste elsewhere • Swap Ends: this option is available once a multi-cell selection has been made. Swaps the beginning and ending cells of the current selection. • Include Headers: the same as Select Row, except that the row headers are selected as well • Selection: toggles selection mode on and off • Font size: select from a small, medium, or large font size You can also use any of the gestures common to the Numeric views. See Numeric View for more details.
Parametric Numeric Setup	Press Shift Num to enter the Parametric Numeric setup. This view enables you to control the appearance of the table in the Numeric View, including which T-value is at the top of the table, the step between T-values, and the zoom factor is for zooming in and out on a row of the table. The fields are: • Num Start: the first value of T shown in the table • Num Step: the table step value (increment) for T • Num Zoom: the zoom factor for zooming • Num Type: choose between table types • Automatic: provides T-, X-, and Y-values

Help Topics Tree 13217	Help Text
	• Build Your Own: you supply T-values; the app provides the corresponding X- and Y-values for each checked definition in Symbolic view The menu buttons are: • Edit: opens an edit line to edit the current value in a field • Choose: select table type • Plot→: sets Num Start and Num Step so that the Numeric view table independent variable values match the independent variable values while tracing in Plot view
Parametric Variables	To display the variables relating to the Parametric app, press Vars, tap App and select Parametric. The Parametric app has variables in the following categories: • Symbolic (see immediately below) • Plot (see Common App Variables) • Numeric (see Common App Variables) • Modes (see Common App Variables) Symbolic View Variables The Parametric app variables are X0-X9 and Y0-Y9. These variables are always defined in pairs and contain algebraic expressions dependent on the variable T. 'Xn := f(T)' 'Yn := g(T)' where n is an integer between 0 and 9 inclusive and f(T) and g(T) are algebraic expressions dependent on T. Example: X1 := '4*COS(6*T)' Y1 := '4*SIN(T)'
Sequence app	Sequences are expressions depending on an integer parameter N>0. Sequences can be defined explicitly or recursively in terms of the previous one or two terms. The Sequence app allows you to explore up to 10 sequences, named U1 to U9. Some of the most famous sequences are the recursive definitions of factorial: U1(1)=1 U1(N)=N*U1(N-1) and the Fibonacci sequence: U1(1)=1 U1(2)=1 U1(N)=U1(N-1)+U1(N-2) Once you have defined a sequence, you can view a table of its values or plot its graph. To launch the Sequence app, go to the Application Library and tap the Sequence app icon. You can also use the rocker wheel to select the Sequence app icon, then tap Start or press Enter to launch the app. Examples: N*U1(N-1) U1(N-1)+U1(N-2)
Sequence Symbolic View	The HP Prime Sequence app allows you to define sequences explicitly or recursively. Use this view to enter and manage up to ten sequence definitions. Explicit definitions define U(N) in terms of N. Backward recursive definitions can define U(N) in terms of U(N-1) or both U(N-1) and U(N-2). Similarly, forward recursive definitions can define U(N+1) in terms of U(N) or U(N+2) in terms of both U(N+1) and U(N+2). Finally, N can start at 1 (the default value), 0, or any positive integer. The first two fields in Symbolic view contain the first two numerical values in the sequence. For an explicitly-defined sequence, these can both be left blank. For a recursively-defined sequence, you must supply at least one of these two, depending on the nature of your definition. Note that the labels for these values change, depending on the starting value for N that you choose in the Option field. The third field is for your symbolic definition. The Option field contains the starting value for N. After this field is a check box. If left unchecked (the default) then your symbolic definition is for U(N). If the box is checked, then your symbolic definition is for U(N+1) if you entered a single starting value for your sequence, or U(N+2) if you entered two numerical values to start your sequence. The menu buttons are: • Edit: opens an edit line to edit the chosen definition • ✓: toggles the current item on or off • Choose: select graph color • Show: displays the selected sequence definition in full-screen mode with horizontal and vertical scrolling enabled • Eval: resolves references to other symbolic definitions, e.g. U2(N-1)+U1(N). When the symbolic definition of a sequence is being entered or edited, the additional menu items are: • (N-2), (N-1), N, (N+1), U1: typing aids for entering your sequence definitions • Cancel: cancels the current edit • OK: accepts the current edit Highlight one of the definition fields and begin entering an expression, or tap Edit to open an edit line to edit an existing expression.
Sequence Plot View	Press Plot to enter the Plot view and explore the sequence graphs. Tap Menu to open the menu. The menu buttons are: • Zoom: enters the zoom menu, with options to zoom in or out • Trace: toggles the tracing cursor off and on

Help Topics Tree **13217** **Help Text**

• Go To: takes the tracing cursor to the point on the graph with a given value of n.

• Defn: displays the symbolic definition of the current sequence

• Menu: toggles the menu off and on

Use the rocker wheel left/right or tap to trace along a function. Use the rocker wheel up/down to switch from one function to another. Press + to zoom in on the current cursor location and press - to zoom out. Set the zoom factor under the Zoom menu.
You can also use all the gestures common to the Plot views. See Plot View for more details.

Sequence Plot Setup

Press Shift Plot to enter the Plot Setup. Here you can manually set up the graphing window and the appearance of the sequence graphs. This setup has two pages.
The fields on the first page are:

• Seq Plot: chooses between Stairstep and Cobweb plots of each sequence

• N Rng: the range of terms to plot for each sequence

• X Rng: the horizontal graphing range

• Y Rng: the vertical graphing range

• X Tick: horizontal tick mark spacing

• Y Tick: vertical tick mark spacing

The menu buttons on the first page are:

• Choose: opens the Seq Plot choose box

• Edit: opens an edit line to edit the value of the selected field

• Page 1/2 ▼: enters the second page of the view

Tap Page 1/2 ▼ to view the second page of the setup. Here the fields are:

• Axes: toggles axes on and off

• Labels: toggles axis labels on and off

• Grid Dots: toggles grid dots on and off

• Grid Lines: toggles grid lines on and off

• Cursor: choose between Standard, Inverting, and Blinking cursors

The menu buttons on the second page are:

• ✓: toggles the current setting on or off

• Choose: make a choice from a choose box

• ▲ Page 2/2: returns to the first page of the setup

Sequence Numeric View

Press Num to enter the Numeric View. The Numeric View is designed to create and explore a table of term and sequence values, based on the sequence(s) defined in the Symbolic View.

The menu buttons are:

• Zoom: zooms in or out on a highlighted row of the table. Note that in Numeric view, zooming changes the increment between consecutive x-values. Zooming in decreases the increment; zooming out increases the increment. The values in the row you zoom in or out on remain the same.

• More: opens a menu with editing options

• Go To: jumps to a specified value of the independent variable

• Defn: displays the definition of the selected column

The More menu

The More menu contains the following options:

• Select

    • Row: selects the row that contains the currently selected cell; the row can then be copied to paste elsewhere
    • Swap Ends: this option is available once a multi-cell selection has been made. Swaps the beginning and ending cells of the current selection.
    • Include Headers: the same as Select Row, except that the row headers are selected as well

• Selection: toggles selection mode on and off

• Font size: select from a small, medium, or large font size

Highlight any value in the N-column and enter any counting number greater than or equal to the starting value for N. The table will reconfigure to show your value.
You can also use any of the gestures common to the Numeric views. See Numeric View for more details.

Sequence Numeric Setup

Press Shift Num to enter the Numeric setup. This view enables you to control the appearance of the table in Numeric View, including which N-value is at the top of the table, the step between N-values, and the zoom factor is for zooming in and out on a row of the table.

Num Step must be a positive integer (any other values will be ignored).

The fields are:

• Num Start: the first value of N shown in the table

• Num Step: the positive integer step between consecutive N-values

• Num Zoom: the positive integer zoom factor for zooming

• Num Type: choose between table types

    • Automatic: provides N- and sequence-values
    • BuildYourOwn: you supply N-values; the App provides the corresponding sequence-values

The menu buttons are:

• Edit: opens an edit line to edit the current value in a field

• Choose: select table type

• Plot→: sets Num Start and Num Step so that the Numeric view table independent variable values match the independent variable values while tracing in Plot view

Help Topics Tree 13217	Help Text
Sequence Variables	To display the variables relating to the Sequence app, press Vars, tap App and select Sequence. The Sequence app has variables in the following categories: • Symbolic (see immediately below) • Plot (see Common App Variables) • Numeric (see Common App Variables) • Modes (see Common App Variables) Symbolic View Variables The Sequence app symbolic variables are U1 through U9 and U0. These variables contain lists that define a sequence. The number of members in the list depends on the type of sequence defined. Ux := '{expression'[, Ux(1), Ux(2)[, StartIndex [,Forward definition]]]}, where x is an integer between 0 and 9 inclusive and expression is an algebraic expression in terms of any combination of the following: • N • Un(N-1) • Un(N-2) Examples: U3 := {'N!'} defines U3 to be a sequence of factorials U5 := {'U5(N-1)+1,1'} defines U5 as the sequence of counting numbers U7 := ('U7(N-2)+U7(N-1),1,1'} defines U7 as the Fibonacci sequence
Polar app	The Polar app allows you to explore the graphical representation of equations using polar coordinates. Each equation takes the form R= f(θ) and its graphical representation is the set of points whose polar coordinates satisfies the R= f(θ) relationship. Once you have defined a polar equation, you can view a table of its values or plot its graph. To launch the Polar app, go to the Application Library and tap the Polar app icon. You can also use the rocker wheel to select the Polar app icon, then tap Start or press Enter to launch the app.
Polar Symbolic View	Press Symb to return to this view at any time. The Polar Symbolic view contains fields to define up to ten polar equations, each one defining R in terms of θ. The menu buttons are: • Edit: opens an input line to edit the selected polar definition • ✓: toggles the selected definition on or off for plotting and table-building • θ: a typing aid for entering definitions in θ • Show: displays the selected definition in full-screen mode with horizontal and vertical scrolling enabled • Eval: resolves references to other polar equations, such as R2(θ)=2-R1(θ) • Choose: select a color for the graph Highlight one of the definition fields and begin entering an expression in θ, or tap Edit to open an edit line to edit an existing expression. Example: 6*SIN(6*θ)
Polar Plot View	Press Plot to enter the Polar Plot view. This view displays the graphs of Polar equations defined in the Symbolic view. The functionality here is the same as in the Function Plot view, except that the Fcn functions do not apply here. Tap Menu to open the menu. The menu buttons are: • Zoom: enters the Zoom menu, with options to zoom in or out • Trace: toggles the tracing cursor off and on • Go To: takes the tracing cursor to the point on the graph with a given value of θ. • Defn: displays the symbolic definition of the current graph • Menu: toggles the menu off and on Use the rocker wheel left/right or tap to trace along a function. Use the rocker wheel up/down to switch from one function to another. Press + to zoom in on the current cursor location and press - to zoom out. Set the zoom factor under the Zoom menu. You can also use all the gestures common to the Plot views. See Plot View for more details.
Polar Plot Setup	Press Shift Plot to enter the Polar Plot setup. This view enables you to control the appearance of the graph window, including the appearance of the cursor, whether or not the axes are drawn, etc. The Setup has two pages. On the first page, the fields are: • θ Rng: the range of values for the independent variable θ • θ Step: the step value for the independent variable θ • X Rng: the horizontal graphing range • Y Rng: the vertical graphing range • X Tick: horizontal tick mark spacing • Y Tick: vertical tick mark spacing The menu buttons on the first page are: • Edit: opens an edit line to edit the value of the selected field • Page 1/2 ▼: displays the second page of the setup Tap Page 1/2 ▼ to view the second page of the setup. Here the fields are: • Axes: toggles axes on and off • Labels: toggles axis labels on and off

Help Topics Tree		13217	Help Text
			• Grid Dots: toggles grid dots on and off • Grid Lines: toggles grid lines on and off • Cursor: choose between Standard, Inverting, and Blinking cursors • Method: choose between Adaptive, Fixed-Step Segments, and Fixed-Step Dots The menu buttons on the second page are: • ✓: toggles the current setting on or off • Choose: make a choice from a choose box • ▲ Page 2/2: returns to the first page of the setup The Method field requires an explanation. By default, the HP Prime uses the Adaptive method, an advanced method that gives very accurate results. You can choose the more traditional method, called Fixed-Step Segments, which samples x-values, computes their corresponding y-values, and then plots and connects the points. Or you can choose Fixed-Step Dots, which works like Fixed-Step Segments but does not connect the points.
		Polar Numeric View	Press Num to enter the Polar Numeric View. The Polar Numeric View is designed to create and explore a table of θ/R values, based on the definitions in Symbolic View. Place the highlight bar in any row of the θ-column and enter any real value. The table will reconfigure. You can also zoom in or out on any row in the table. Press + to zoom in on a row of the table and - to zoom out. The menu buttons are: • Zoom: zooms in or out on a highlighted row of the table. Note that in Numeric view, zooming changes the increment between consecutive x-values. Zooming in decreases the increment; zooming out increases the increment. The values in the row you zoom in or out on remain the same. • More: opens a menu with editing options • Go To: jumps to a specified value of the independent variable • Defn: displays the definition of the selected column The More menu The More menu contains the following options: • Select • Row: selects the row that contains the currently selected cell; the row can then be copied to paste elsewhere • Swap Ends: this option is available once a multi-cell selection has been made. Swaps the beginning and ending cells of the current selection. • Include Headers: the same as Select Row, except that the row headers are selected as well • Selection: toggles selection mode on and off • Font size: select from a small, medium, or large font size You can also use any of the gestures common to the Numeric views. See Numeric View for more details.
		Polar Numeric Setup	Press Shift Num to enter the Polar Numeric setup. This view enables you to control the appearance of the table in Numeric View, including which θ-value is at the top of the table, the step between θ-values, and the zoom factor is for zooming in and out on a row of the table. The fields are: • Num Start: the first value of θ shown in the table • Num Step: the common difference between consecutive θ-values • Num Type: choose between table types • Automatic: provides θ- and R-values • BuildYourOwn: you supply θ-values; the app provides the corresponding R-values • Num Zoom: the zoom factor for zooming The menu buttons are: • Edit: opens an edit box to edit the current value in a field • Plot→: sets Num Start and Num Step so that the Numeric view table independent variable values match the independent variable values while tracing in Plot view
		Polar Variables	To display the variables relating to the Polar app, press Vars, tap App and select Polar. The Polar app has variables in the following categories: • Symbolic (see immediately below) • Plot (see Common App Variables) • Numeric (see Common App Variables) • Modes (see Common App Variables) Symbolic View Variables The Polar app symbolic variables are R1 through R9 and R0. These variables contain algebraic expressions in terms of θ. Rn:=f(θ), where n is an integer between 0 and 9 inclusive and f(θ) is an algebraic expression in θ. Example: R1:='2*θ'
The Toolbox Menus			Press the Toolbox key to access menus listing the most common calculator functions and commands. The menus are: • Math - most commonly used math functions • CAS - most commonly used symbolic functions • App - all app-specific functions • User - all the functions and programs you have created yourself • Catlg - Catalog of all the functions and commands

Help Topics Tree 13217	Help Text
	You can select an option by tapping it, or by using the rocker wheel to highlight it and then pressing Enter. In a menu of options, you can also select an entry by its number or by typing in the first letter or two of its name and pressing the Enter key. You can also enter a function or command letter-by-letter. Menu Display Mode: The default menu display mode is to display the descriptive names for the Math and CAS functions. For example, "ifactors" is presented as "Factors List". If you prefer the functions to be presented by their command name, deselect the Menu Display option on Home Settings Page 2.
Math Menu	Toolbox Math Menu The Toolbox Math menu lists the most commonly used math functions.
Numbers	This menu lists the basic real number functions
CEILING	Syntax: CEILING(value) Least integer greater than or equal to value. Examples: CEILING(3.2) → 4 CEILING(-3.2) → -3 CEILING({3.2,-3.2}) → {4,−3}
FLOOR	Syntax: FLOOR(value) Greatest integer less than or equal to value. Examples: FLOOR(3.2) → 3 FLOOR(-3.2) → -4 FLOOR({3.2,-3.2}) → {3,−4}
IP	Integer Part Syntax: IP(value) Returns the Integer part of value. Examples: IP(23.2) → 23 IP(-23.2) → -23 IP({23.2,15+1/4,51/2,10-4/5}) → {23,15,25,9}
FP	Fractional Part Syntax: FP(value) Returns the Fractional part of value. Examples: FP(23.2) → 0.2 FP(-23.2) → -0.2 FP({23.2,15+1/4,51/2,10-4/5}) → {0.2,0.25,0.5,0.2}
ROUND	Syntax: ROUND(value, [places]) Rounds value to system display settings. If optional places is given, rounds value to places decimal places. If places is negative, rounds to significant digits instead. Examples: ROUND(7.8676,2) → 7.87 ROUND(7.8676,-2) → 7.9 ROUND((2-3*i)^(2+3*i),4) → −75.8927+236.0767*i ROUND({22/6,7/6,13/6},{-3,3,4}) → {3.67,1.167,2.1667}
TRUNCATE	Syntax: TRUNCATE(value, [places]) Truncates value to system display settings. If optional places is given, truncates value to places decimal places. If places is negative, truncates to significant digits instead. Examples: TRUNCATE(2.3678,2) → 2.36 TRUNCATE(2.3678,-2) → 2.3 TRUNCATE((2-3*i)^(2+3*i),2) → −75.89+236.07*i TRUNCATE({22/6,7/6,13/6},{-3,3,4}) → {3.66,1.166,2.1666}
MANT	Mantissa Syntax: MANT(Value) Returns the significant digits of Value. Examples: MANT(21.2E34) → 2.12 MANT({2.12ᴇ35,5302.00000123}) → {2.12,5.30200000123}
XPON	Exponent Syntax: XPON(value)

Help Topics Tree		13217	Help Text
			Exponent. Returns the exponent of value. Examples: XPON(123.4) → 2 XPON({0.001234,56789.0123}) → {-3,4}
Arithmetic			Arithmetic Menu This menu lists the basic arithmetic functions
	MAX		Maximum Syntax: MAX(value1,[value2],[..value16]) or MAX(list) Returns the greatest of the values given, or the greatest value of a list. Examples: MAX(210,25) → 210 MAX({1,8,2}) → 8 MAX(8/3,11/4) → 2.75 MAX({1,8,2},{2,4,6}) → {2,8,6}
	MIN		Minimum Syntax: MIN(value1,[value2],[..value16]) or MIN(list) Returns the least of the values given, or the least value of a list. Examples: MIN(210,25) → 25 MIN({1,8,2}) → 1 MIN(8/3,11/4) → 2.6667 MIN({1,8,2},{2,4,6}) → {1,4,2}
	MOD		Modulo Syntax: value1 MOD value2 Returns the remainder of the Euclidean division value1/value2. Examples: 9 MOD 4 → 1 #27o MOD 12 → 11 [[1,3],[13,14]] MOD 4 → [[1,3],[1,2]] {11,12,13,15,17} MOD 4 → {3,0,3,1,3}
	FNROOT		Find Root Syntax: FNROOT(Expr, Var, [guess], [guess2]) Function root-finder (like the Solve app). Finds the value for variable at which an expression most nearly evaluates to zero. Uses guess as initial estimate. Examples: FNROOT(A*9.8/600-1,A,1) → 61.2244897959 FNROOT(X^2-3,X,-2) → −1.73205080757 FNROOT(X^2-3,X,2) → 1.73205080757 FNROOT(X^2-3,X,2,-2) → −1.73205080757 FNROOT({'X^2-3','T^3+4'},{'X','T'},{-2,-1},{2,1}) → {−1.73205080757,−1.58740105197}
	%		Percentage Syntax: %(x, y) x percent of y. Returns (x/100)*y. Examples: %(20,50) → 10 %(1.5,7.5) → 0.1125 %({10,20,30},{75,75,75}) → {7.5,15,22.5}
	Complex		This menu lists the basic complex number functions
		ARG	Argument Syntax: ARG(x+yi) Finds the angle determined by a complex number. Example: ARG(3+3i) → 45 (degrees mode)
		CONJ	Complex Conjugate Syntax: CONJ(x+yi) Reverses the sign of the imaginary part of a complex number. Examples: CONJ(3+4*i) → 3-4*i

Help Topics Tree				13217	Help Text
					(CONJ({3+4*i,6-6*i}) → {3-4*i,6+6*i}
				IM	Imaginary Part Syntax: IM(x+yi) Returns the imaginary part of a complex number. Examples: IM(3+4i) → 4 IM({3+4*i,6-6*i}) → {4,-6}
				RE	Real Part Syntax: RE(x+yi) Returns the real part of a complex number. Examples: RE(3+4i) → 3 RE({3+4*i,6-6*i}) → {3,6}
				SIGN	Sign or Unit Vector Syntax: SIGN(value) SIGN(x+yi) Returns the sign of value. If positive, the result is 1; if negative, -1. If zero, the result is zero. For complex inputs returns the unit vector. Examples: SIGN(2) → 1 SIGN(3+4i) → 0.6+0.8i SIGN({3-√13,6+8*i}) → {−1,0.6+0.8*i}
			Exp and Ln		This menu lists the basic exponential and log functions
				ALOG	Common Antilogarithm Syntax: ALOG(value) Common exponential: 10^x (antilogarithm) Returns the result of raising 10 to the power of value. Examples: ALOG(2) → 100 ALOG(2+3*i) → 81.121465284+58.4748481843*i ALOG({2,4}) → {100,10000}
				EXPM1	Exponent Minus 1 Syntax: EXPM1(value) Exponential minus 1: (e^x)-1 This is more accurate than EXP when x is close to zero. Examples: EXPM1(0.23) → 0.258600009929 EXPM1(0.02+0.03*i) → 1.97422838545ᴇ−2+3.06014495014ᴇ−2*i
				LNP1	Natural Log Plus 1 Syntax: LNP1(value) Natural log plus 1: LN(X+1) This is more accurate than the natural logarithm function for values close to zero. Examples: LNP1(0.23) → 0.207014169384 LNP1(0.02+0.03*i) → 2.02349662769ᴇ−2+0.029403288204*i
		Trigonometry			This menu lists the basic trigonometric functions
			CSC		Cosecant Syntax: CSC(value) Cosecant: 1/SIN(X) Example: CSC(90) → 1 (Degrees mode) CSC(1+i) → 0.621518017169-0.303931001627*i CSC({30,90}) → {2,1} (Degrees mode) CSC((π/6)_rad) → 2
			ACSC		Arc Cosecant Syntax: ACSC(value) Inverse Cosecant: CSC^-1 (X) Example: ACSC(1) → 90 (Degrees mode) ACSC(0.621518017169-0.303931001627*i) → 1+i

Help Topics Tree	13217	Help Text
		ACSC({2,1}) → {30,90} (Degrees mode)
SEC		Secant Syntax: SEC(value) Secant: 1/COS(X). Example: SEC(0) → 1 (Degrees mode) SEC(1+i) → 0.498337030555+0.591083841721*i SEC({60,0}) → {2,1} (Degrees mode) SEC((π/3)_rad) → 2.00000000001
ASEC		Arc Secant Syntax: ASEC(value) Inverse Secant: SEC^-1 (X) Example: ASEC(1) → 0 (Degrees mode) ASEC(0.498337030555+0.591083841721*i) → 1+i ASEC({2,1}) → {60,0} (Degrees mode)
COT		Cotangent Syntax: COT(value) Cotangent: COS(X)/SIN(X) Example: COT(45) → 1 (Degrees mode) COT(1+i) → 0.217621561854-0.868014142896*i COT({45,90}) → {1,0} (Degrees mode) COT((π/4)_rad) → 1
ACOT		Arc Cotangent Syntax: ACOT(value) Inverse Cotangent: COT^-1 (X) Example: ACOT(1) → 45 (Degrees mode) ACOT(0.217621561854-0.868014142896*i) → 1+i ACOT({1,0}) → {45,90} (Degrees mode)
Hyperbolic		This menu lists the basic hyperbolic functions
SINH		Hyperbolic Sine Syntax: SINH(value) Hyperbolic Sine Examples: SINH(1) → 1.17520119364 SINH(1+i) → 0.634963914785+1.29845758142*i SINH({0,1}) → {0,1.17520119364}
ASINH		Inverse Hyperbolic Sine Syntax: ASINH(value) Inverse Hyperbolic Sine: SINH^-1 (X) Examples: ASINH(1.17520119365) → 1 ASINH(0.634963914785+1.29845758142*i) → 1+i ASINH({0,1.17520119365}) → {0,1}
COSH		Hyperbolic Cosine Syntax: COSH(value) Hyperbolic Cosine Examples: COSH(1) → 1.54308063482 COSH(1+i) → 0.833730025131+0.988897705763*i COSH({0,1}) → {1,1.54308063482}
ACOSH		Inverse Hyperbolic Cosine Syntax: ACOSH(value) Inverse Hyperbolic Cosine: COSH^-1 (X) Examples: ACOSH(1.54308063482) → 1 ACOSH(0.833730025131+0.988897705763*i) → 1+i ACOSH({1,1.54308063482}) → {0,1}
TANH		Hyperbolic Tangent

Help Topics Tree		13217	Help Text
			Syntax: TANH(value) Hyperbolic Tangent Examples: TANH(1) → 0.761594155956 TANH(1+i) → 1.08392332734+0.27175258532*i TANH({0,0.5}) → {0,0.46211715726}
	ATANH		Inverse Hyperbolic Tangent Syntax: ATANH(value) Inverse Hyperbolic Tangent: TANH^-1 (X) Examples: ATANH(.761594155956) → 1 ATANH(1.08392332734+0.27175258532*i) → 1+i ATANH({0,0.46211715726}) → {0,0.5}
Probability			This menu lists the basic probability functions
	!		Factorial Syntax: value! For Whole numbers, calculates the Factorial of value. For Negative Integer, Real, or Complex numbers, calculates the Gamma function: x! = Γ(x + 1). Examples: 6! → 720 3.45! → 10.8547765843 (2.+3.*i)! → −0.440113407637-6.36372431263e−2*i ([[2,3,4],[4,5,6]])! → [[2,6,24],[24,120,720]] (3!)_miles → 6_miles {6,5,4}! → {720,120,24}
	COMB		Combinations Syntax: COMB(n, r) Returns the number of combinations (without regard to order) of n things taken r at a time: n!/(r!(n-r)!) Examples: COMB(5,2) → 10 COMB({5,10,15},{1,2,3}) → {5,45,455}
	PERM		Permutations Syntax: PERM(n, r) Returns the number of permutations (with regard to order) of n things taken r at a time: n!/(n-r)! Examples: PERM(5,2) → 20 PERM({5,10,15},{1,2,3}) → {5,90,2730}
	Random		Random Number Functions This menu contains random number functions.
		RANDOM	Random Number Syntax: RANDOM([a],[b],[c]) Returns a pseudo-random number generated using a seed value, and updates the seed value. With no argument, this function returns a random number x with 0 ≤ x < 1. With one argument, it returns a random number x with 0 ≤ x < a. With two arguments, it returns a random number with a ≤ x < b. With three arguments, this returns a list of size a with each element being a random number x with b ≤ x < c. Examples: RANDOM RANDOM(5) RANDOM(3,5) RANDOM(3,0,10)
		RANDINT	Random Integer Syntax: RANDINT([a],[b],[c]) Returns a pseudo-random integer generated using a seed value, and updates the seed value. With no argument, this function returns a random integer x from 0 to 1. With one argument, it returns a random integer x from 0 to a. With two arguments, it returns a random integer x from a to b.

Help Topics Tree	13217	Help Text

		With three arguments, it returns a list of size a with each element being a random integer x from b to c.
		
		Examples:
		RANDINT
		RANDINT(6)
		RANDINT(1,6)
		RANDINT(3,1,6) → { random1, random2, random3 }

	RANDNORM	Random Normal Syntax: RANDNORM([μ],[σ]) or RANDNORM(n,μ,σ) Return a random number from the normal distribution with the specified mean μ and standard deviation σ. Default values are 0 and 1. With three arguments, returns a list of size n with each element being a random number from the normal distribution with the specified mean μ and standard deviation σ. Examples: RANDNORM(1.23) RANDNORM(1.2,2.3) RANDNORM(3,0,1) → { random1, random2, random3 }
	RANDSEED	Random Seed Syntax: RANDSEED([value]) Sets the random number generator seed. With no input, it uses the current time value as seed. Examples: RANDSEED(3.14) RANDSEED(3.14); RANDOM(5) → 3.34681220106 RANDSEED(3.14); RANDINT(3,1,6) → {5,5,3}
Density		This menu lists the density distributions functions
	NORMALD	Normal Density Syntax: NORMALD([μ, σ,] x) Normal probability density function. Computes the probability density at the value x, given the mean, μ, and standard deviation, σ, of a normal distribution. With one argument, x, it returns the probability density for the standard normal distribution at x, assuming a mean of zero and standard deviation of 1. Examples: NORMALD(0.5) → 0.352065326764 NORMALD(0,2,0.5) → 0.193334058401
	STUDENT	Student's t Density Syntax: STUDENT(d, x) Student’s t probability density function Computes the probability density of the Student’s-t distribution at x, given d degrees of freedom. Example: STUDENT(3,5.2) → 0.00366574413491
	CHISQUARE	χ^2 Density Syntax: CHISQUARE(d, x) χ^2 (Chi-squared) probability density function Computes the probability density of the χ^2 distribution at x, given d degrees of freedom. Example: CHISQUARE(2,3.2) → 0.100948258997
	FISHER	Fisher Density Syntax: FISHER(n, d, x) 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. Example: FISHER(5,5,2) → 0.158080231095
	BINOMIAL	Binomial Probability Density Syntax: BINOMIAL(n, p, k) Binomial probability density function. Computes the probability of k successes out of n trials, each with a probability of success of p. Note that n and k are integers with k≤n. Example: BINOMIAL(4,0.5,2) → 0.375

Help Topics Tree	13217	Help Text
	GEOMETRIC	Geometric Density Syntax: GEOMETRIC(p,x) Geometric probability density function Computes the probability density of the geometric distribution at x, given probability p. Example: GEOMETRIC(0.3,4) → 0.1029
	POISSON	Poisson Density Syntax: POISSON(μ, k) Poisson probability mass function Computes the probability of k occurrences of an event in a time interval, given μ expected (or mean) occurrences of the event in that interval. For this function, k is a non-negative integer and μ is a real number. Example: POISSON(4, 2) → 0.14652511111
Cumulative		This menu lists the cumulative distributions functions
	NORMALD_CDF	Cumulative Normal Syntax: NORMALD_CDF([μ, σ,] x, [x2]) Cumulative normal distribution function. With three values (μ, σ, and x), 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. With the optional fourth value x2, returns the area under the normal probability density function between the two x-values. With one argument x, returns the lower-tail probability of the standard normal probability density function for the value x, assuming a mean of zero and standard deviation of 1. Examples: NORMALD_CDF(2)→0.977249868052 NORMALD_CDF(-1,1)→0.682689492138 NORMALD_CDF(0,1,2) → 0.977249868052 NORMALD_CDF(0,1,0,2) → 0.477249868052
	STUDENT_CDF	Cumulative Student's t Syntax: STUDENT_CDF(d, x, [x2]) Cumulative Student’s t distribution function With two values (n and x), returns the lower-tail probability of the Student’s t probability density function at x, given d degrees of freedom. With the optional third argument x2, returns the area under the Student's t probability density function between the two x-values. Examples: STUDENT_CDF(3,-3.2) → 0.0246659214813 STUDENT_CDF(3,-3.2,1) → 0.779832969041
	CHISQUARE_CDF	Cumulative χ² Syntax: CHISQUARE_CDF(d, x, [x2]) Cumulative χ² (Chi-squared) distribution function With two values (n and x) returns the lower-tail probability of the χ² probability density function for the value x, given d degrees of freedom. With the optional third argument x2, returns the area under the χ² probability density function between the two x-values. Examples: CHISQUARE_CDF(2,6.3) → 0.957147873133 CHISQUARE_CDF(2,2,6.3) → 0.325027314304
	FISHER_CDF	Cumulative Fisher Syntax: FISHER_CDF(n, d, x, [x2]) Cumulative F (Fisher or Fisher-Snedecor) distribution function Returns the lower-tail probability of the F probability density function for the value x, given numerator n and denominator d degrees of freedom. With the optional fourth argument x2, returns the area under the F probability density function between the two x-values. Examples: FISHER_CDF(5,5,2) → 0.76748868087 FISHER_CDF(5,5,0.5,2) → 0.53497736174
	BINOMIAL_CDF	Cumulative Binomial Syntax: BINOMIAL_CDF(n, p, k, [k2]) Cumulative binomial distribution function

Help Topics Tree	13217	Help Text
		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≤n. With the optional fourth argument k2, returns the cumulative probability for the two k-values; that is, the probability of between k and k2 successes. Examples: BINOMIAL_CDF(20,0.5,6) → 0.05765914917 BINOMIAL_CDF(20,0.5,6,12) → 0.847717285156
	geometric_cdf	Cumulative Geometric Syntax: geometric_cdf(p,x,[x2]) 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, x1, and x2), returns the area under the geometric probability density function defined by the probability p, between x1 and x2. Examples: geometric_cdf(0.3,4) → 0.7599 geometric_cdf(0.5,1,3) → 0.875
	POISSON_CDF	Cumulative Poisson Syntax: POISSON_CDF(μ, k, [k2]) Cumulative Poisson distribution function Returns the probability of k or fewer occurrences of an event in a given time interval, given μ expected (or mean) occurrences. With the optional third argument k2, returns the probability of between k and k2 occurrences. Examples: POISSON_CDF(4,2) → 0.238103305554 POISSON_CDF(4,2,3) → 0.341891925923
Inverse		This menu lists the inverse cumulative distributions functions
	NORMALD_ICDF	Inverse Cumulative Normal Syntax: NORMALD_ICDF([μ, σ,] p) Inverse cumulative normal distribution function. Returns the cumulative normal distribution x-value associated with the lower-tail probability p, given the mean μ, and standard deviation σ, of a normal distribution. With one argument, p, assumes a mean of 0 and a standard deviation of 1. Examples: NORMALD_ICDF(0.977249868052)→2 NORMALD_ICDF(0,1,0.841344746069) → 1
	STUDENT_ICDF	Inverse Cumulative Student's t Syntax: STUDENT_ICDF(d, p) Inverse cumulative Student’s t distribution function Returns the value x such that the Student’s-t lower-tail probability of x, with d degrees of freedom, is p. Example: STUDENT_ICDF(3,0.0246659214813) → -3.2
	CHISQUARE_ICDF	Inverse Cumulative χ² Syntax: CHISQUARE_ICDF(d, p) Inverse cumulative χ² (Chi-squared) distribution function Returns the value x such that the χ² lower-tail probability of x, with d degrees of freedom, is p. Example: CHISQUARE_ICDF(2,0.957147873133) → 6.3
	FISHER_ICDF	Inverse Cumulative Fisher Syntax: FISHER_ICDF(n, d, p) Inverse cumulative F (Fisher or Fisher-Snedecor) distribution function. Returns the value x such that the F lower-tail probability of x, with numerator n, and denominator d degrees of freedom, is p. Example: FISHER_ICDF(5,5,0.76748868087) → 2
	BINOMIAL_ICDF	Inverse Cumulative Binomial Syntax: BINOMIAL_ICDF(n, p, q) 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. Example: BINOMIAL_ICDF(4,0.5,0.6875) → 2
	geometric_icdf	Inverse Cumulative Geometric Syntax:

Help Topics Tree	13217	Help Text
		geometric_icdf(p,k) Inverse cumulative geometric distribution function Returns the value x that has the lower-tail probability value k, given the probability p. Example: geometric_icdf(0.3,0.95) → 9
	POISSON_ICDF	Inverse Cumulative Poisson Syntax: POISSON_ICDF(μ, p) Inverse cumulative Poisson distribution function. Returns the value k such that the probability of k or fewer occurrences of an event in a time interval, with μ expected (or mean) occurrences of the event in the interval, is p. Example: POISSON_ICDF(4,0.238103305554) → 3
List		This menu lists the basic lists functions
MAKELIST		Make List Syntax: MAKELIST(expression, variable, begin, end, [increment]) Calculates a sequence of elements for a new list. Evaluates expression, incrementing variable from begin to end values, using increment steps (default is 1). Example: MAKELIST(2*X-1,X,1,5,1) → {1,3,5,7,9}
SORT		Sort List Syntax: SORT(list,[sort_by]) Sorts the elements of a list in ascending order. For compound lists (lists of lists or lists containing strings), sort_by identifies the element number within each object in the list to be used for sorting. The list must be in the format {object_1,object_2,...,object_n). Examples: SORT({2,9,5,3}) → {2,3,5,9} SORT({"foo","bar","bra"}) → {"bar","bra","foo"} SORT({"foo","bar","bra"},2) → {"bar","foo","bra"} (sort the list by the 2nd element of each object) SORT({{10,2,7,6,9},{5,9,8,6,9},{4,10,3,1,6},{7,6,1,8,6}},3) → {{7,6,1,8,6},{4,10,3,1,6},{10,2,7,6,9},{5,9,8,6,9}} (sort the list by the 3nd element of each object) SORT({{10,2,"CMABA",6,9},{5,9,"EOGJI",6,9},{4,10,"DOFEB",1,6},{7,6,"IHLCP",8,6},{0,0,"ONAED"}},{3,2}) → {{7,6,"IHLCP",8,6},{10,2,"CMABA",6,9},{0,0,"ONAED"},{4,10,"DOFEB",1,6},{5,9,"EOGJI",6,9}} (sort each list by the 2nd element of the 3rd object) SORT({"DACCI","GCIFA","GBCHA","AJEGE","BDCCA"},{3,1}) → {"BDCCA","DACCI","GBCHA","AJEGE","GCIFA"} (sort the list by the 3rd element followed by the 2nd element of each object)
REVERSE		Reverse List Syntax: REVERSE(list) Creates a list by reversing the order of the elements in list. Example: REVERSE({2,3,4,5}) → {5,4,3,2}
CONCAT		Concatenate Syntax: CONCAT(value1, value2, [..value16]) or CONCAT(List1, List2) or CONCAT(List, Item) Concatenates (joins) items into a list or concatenates two lists. Examples: CONCAT({1,2,3},4) → {1,2,3,4} CONCAT(1,2,3,4) → {1,2,3,4} CONCAT({1,2},3,{{4,5},6,{7,8}}) → {1,2,3,{4,5},6,{7,8}}
SUPPRESS		Remove Items Syntax: SUPPRESS(object, index) SUPPRESS(object, start, end) SUPPRESS(object, {index1, index2, ... indexN}) SUPPRESS(string1, string2) Remove items from object using a single position index, a start and end index range, or a list of indices. object may be a list, vector, or string. In the case of string1 and string2, every instance of each character in string2 will be removed from string1. Examples: SUPPRESS({1,2,3,4},3) → {1,2,4}

Help Topics Tree	13217	Help Text
		SUPPRESS({1,2,3,4},2,3) → {1,4} SUPPRESS({1,2,3,4},{1,3}) → {2,4} SUPPRESS([1,2,3,4],3) → [1,2,4] SUPPRESS([1,2,3,4],2,3) → [1,4] SUPPRESS([1,2,3,4],{1,3}) → [2,4] SUPPRESS("1234",3) → "124" SUPPRESS("1234",2,3) → "14" SUPPRESS("1234",{1,3}) → "24" SUPPRESS("FizzBuzz","zu") → "FiB"
INSERT		Insert Items Syntax: INSERT(object1, index, object2) Insert object2 into object1 immediately prior to position specified by index. If index is one greater than the size of object1, object2 will be appended to object1. object1 may be a list, vector, or string. object2 may be anything if object1 is a list. object2 must be a real or complex number if object1 is a vector. object2 must be a single character string if object1 is a string.
POS		Position Syntax: POS(list, element) Returns the position of element within list. If there is more than one instance of element, the position of the first occurrence is returned. Returns 0 if there is no occurrence of the specified element. Example: POS({0,1,3,5},1) → 2
SIZE		List Size Syntax: SIZE(list) Returns the number of elements in a list. With a matrix, returns the dimensions of the matrix. Example: SIZE({0,1,2,3}) → 4
ΔLIST		Δ List Syntax: ΔLIST(list) Creates a new list composed of the first differences of a given list; that is, the differences between the sequential elements in a list. The new list has one fewer elements than the original list. Example: ΔLIST({1,2,3,5,8}) → {1,1,2,3}
ΣLIST		Σ List Syntax: ΣLIST(list) Calculates the sum of all elements in a list. If the list contains a string, the result will be a single string with all elements concatenated together. Examples: ΣLIST({2,3,4}) → 9 ΣLIST({"A","B","CE"}) → "ABCE" ΣLIST({"A",1,"B",2,"CE",3}) → "A1B2CE3"
ΠLIST		Π List Syntax: ΠLIST(list) Calculates the product of all elements in a list. Example: ΠLIST({2,3,4}) → 24
DIFFERENCE		List Difference Syntax: DIFFERENCE({list1}, ...{listN}) Returns a list of the elements that are not common between two or more lists. Examples: DIFFERENCE({1,2,3},{2,4,8}) → {1,3,4,8} DIFFERENCE({1,2,3},{2,4,8},{1,2},{3,5,8}) → {4,5}

INTERSECT		List Intersect
		Syntax:
		INTERSECT({list1}, ...{listN})
		Returns a list of common elements in two or more lists.
		Examples:
		INTERSECT({1,2,3},{2,4,8}) → {2}
		INTERSECT({1,2,4},{2,4,8}) → {2,4}
		INTERSECT({1,2,3},{2,4,8},{1,3,5,8}) → {}

UNION		List Union Syntax:

Help Topics Tree				13217	Help Text

					UNION(list1 or object1, ... list_n or object_n)
					UNION concatenates the inputs, removing all duplicates.
					Example:
					UNION({1,2,3}, {2,4,8}, 10) → {1, 2, 3, 4, 8, 10}

			EQ		Syntax: EQ(object_1, object2) Returns 1 if the two objects are the same. This function is equivalent to the = and == function with one exception - if the two objects are lists, it returns 1 if the two lists are the same while the = and == functions return a list containing 0 or 1 for each pair of items. Examples: EQ({1,2,3}, {1,2,3}) → 1 EQ({1,2,3}, {0,1,2,3}) → 0
		Matrix			This menu lists the basic matrix functions
			TRN		Transpose Syntax: TRN(matrix) Transposes matrix. If Complex mode is on and the matrix contains complex elements, then TRN finds the conjugate transpose. Examples: TRN([[1,2],[3,4]]) → [[1,3],[2,4]] TRN([[1+2*i,2+4*i],[3+i,4-5*i]]) → [[1+2*i,3+i],[2+4*i,4-5*i]] TRN({[[5,2],[1,3]],[[2,9],[7,8]]}) → {[[5,1],[2,3]],[[2,7],[9,8]]}
			DET		Square Matrix Determinant Syntax: DET(matrix) Determinant of a square matrix. Examples: DET([[1,2],[3,4]]) → -2 DET([[1+2*i,2+4*i],[3+i,4-5*i]]) → 12-11*i DET({[[1,2],[5,6]],[[3,4],[−6,−2]]}) → {−4,18}
			RREF		Reduced-Row Echelon Form Syntax: RREF(matrix) Changes a rectangular matrix to its reduced row-echelon form. Examples: RREF([[1,-2,1],[3,4,-1]]) → [[1,0,0.2],[0,1,-0.4]] RREF([[1+2*i,2+4*i,1+i],[3+i,4-5*i,2-i]]) → [[1,0,0.335849056604-0.275471698113*i],[0,1,0.132075471698+3.77358490566ᴇ−2*i]] RREF({[[−2,2,1],[1,4,0]],[[1,3,1],[3,6,9]]}) → {[[1,0,−0.4],[0,1,0.1]],[[1,0,7],[0,1,−2]]}
			Create		This menu lists the matrix creation functions
				MAKEMAT	Make Matrix Syntax: MAKEMAT(Expr, Rows, Columns) or MAKEMAT(Expr, Elements) Creates a matrix of dimension Rows × Columns, using Expr to calculate each element. If Expr 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 using the number of Elements instead of the number of rows and columns. Examples: MAKEMAT(0,3,3) → [[0,0,0],[0,0,0],[0,0,0]] MAKEMAT(√2,2,3) → [[√2,√2,√2],[√2,√2,√2]] in CAS view MAKEMAT(I+J-1,2,3) → [[1,2,3],[2,3,4]] in Home view
				IDENMAT	Identity Matrix Syntax: IDENMAT(n) Creates a square matrix of dimension n x n whose diagonal elements are 1 and off-diagonal elements are zero. Examples: IDENMAT(2) → [[1,0],[0,1]] IDENMAT({2,3}) → {[[1,0],[0,1]],[[1,0,0],[0,1,0],[0,0,1]]}
				RANDMAT	Random Matrix Syntax: RANDMAT([MatrixName], rows, [columns, [integer or real1, real2 or 'generation_function']]) Creates a random matrix with the specified number of rows and columns. If MatrixName is provided, the result is stored there. If no additional inputs are provided, the entries will be integers ranging from –99 to 99. If one integer is provided, the entries will be integers ranging from 0 to that integer. If two reals are provided, the entries will be reals from real1 to real2.

Help Topics Tree	13217	Help Text
		If one generation_function is provided, it will be used to generate each entry in the matrix. Example: RANDMAT(2,2) → [[n1,n2],[n3,n4]]
	JordanBlock	Jordan Block Syntax: JordanBlock(Expr, n) Returns a square n x n matrix with Expr on the diagonal, 1 above and 0 everywhere else. Examples: JordanBlock(7,3) → [[7,1,0],[0,7,1],[0,0,7]] JordanBlock(x+1,3) → [[x+1,1,0],[0,x+1,1],[0,0,x+1]]
	hilbert	Hilbert Matrix Syntax: hilbert(n) 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. Example: hilbert(3) → [[1,1/2,1/3],[1/2,1/3,1/4],[1/3,1/4,1/5]]
	mkisom	Isometry Syntax: mkisom(Vect,(Sign(1) or -1)) Returns the matrix of an isometry given by its proper elements. Examples: mkisom(π,1) → [[-1,0],[0,-1]] (in radians mode)
	vandermonde	Vandermonde Matrix Syntax: vandermonde(vector) 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. Examples: vandermonde([1,2,3]) → [[1,1,1],[1,2,4],[1,3,9]] vandermonde([a,b,c]) → [[1,a,a^2],[1,b,b^2],[1,b,b^2]]
Basic		This menu lists the basic matrix functions
	ABS	Absolute Value Syntax: ABS(expr) or ABS(matrix) For numerical arguments, returns the absolute value of the expression. For matrix arguments, returns the Frobenius (Euclidean) norm of the array. Examples: ABS(-3.14) → 3.14 ABS([[1,2],[3,4]]) → 5.47722557505 ABS(2-3*i) → 3.60555127546 CAS(ABS([[1,2],[3,4]])) → $\sqrt{30}$
	ROWNORM	Row Norm Syntax: ROWNORM(matrix) Finds the maximum value (over all rows) for the sums of the absolute values of all elements in a row. Example: ROWNORM([[1,2],[3,4]]) → 7
	COLNORM	Column Norm Syntax: COLNORM(matrix) Finds the maximum value (over all columns) of the sums of the absolute values of all elements in a matrix. Example: COLNORM([[1,2],[3,4]]) → 6
	SPECNORM	Spectral Norm Syntax: SPECNORM(matrix) Returns the spectral Norm of a square matrix. Example: SPECNORM([[1,2],[3,4]]) → 5.4650
	SPECRAD	Spectral Radius Syntax: SPECRAD(matrix) Returns the spectral radius of a square matrix. Example:

Help Topics Tree					13217	Help Text
						SPECRAD([[1,2],[3,4]]) → 5.3723
					COND	Condition Number Syntax: COND(matrix) Finds the 1-norm (column norm) of a square matrix. Example: COND([[1,2],[3,4]]) → 21
					RANK	Rank of Rect. Matrix Syntax: RANK(matrix) Returns the rank of a rectangular matrix. Examples: RANK([[1,2],[3,4]]) → 2 RANK([[1,2,3],[3,2,1],[2,1,3]]) → 3 RANK([[1+2*i,2+4*i],[3+i,4-5*i]]) → 2 RANK({[[1,2],[3,4],[5,6]],[[1,2,3],[6,5,4]]}) → {2,2}
					pivot	Syntax: pivot(matrix,n,m) 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. Example: pivot([[1,2],[3,4],[5,6]],1,1) → [[1,2],[0,-2],[0,-4]]
					TRACE	Syntax: TRACE(matrix) 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.) Examples: TRACE([[1,2],[3,4]]) → 5 TRACE([[1+2*i,2+4*i],[3+i,4-5*i]]) → 5-3*i TRACE({[[2,1],[6,3]],[[3,8],[5,7]]}) → {5,10}
				Advanced		This menu lists the advanced matrix functions
					EIGENVAL	Eigenvalues Syntax: EIGENVAL(matrix) Displays the eigenvalues in vector form for matrix. Example: EIGENVAL([[1,2],[3,4]]) → [5.3723, -0.3723]
					EIGENVV	Eigenvectors and Values Syntax: EIGENVV(matrix) Eigenvectors and Eigenvalues for a square matrix Displays a list of two arrays. The first contains the eigenvectors and the second contains the eigenvalues. Example: EIGENVV([[1,2],[3,4]]) → { [[0.4160,-0.8370],[0.9094,0.5743]], [[5.3723,0], [0,-0.3723]]}
					jordan	Syntax: jordan(Matrix) Returns the list made by the matrix of passage and the Jordan form of a matrix. Examples: jordan([[0,2],[1,0]]) → [[√2,-√2],[1,1]],[[√2,0],[0,-√2]] jordan([[-2,-2,1],[-2,1,-2],[1,-2,-2]])
					diag	Diagonal Syntax: diag(list) or diag(matrix) 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. Examples: diag({1,2,3}) → [[1,0,0],[0,2,0],[0,0,3]] diag([[1,2],[3,4]]) → [1,4]
					cholesky	Syntax: cholesky(matrix) For a numerical symmetric matrix A, returns the matrix L such that A=L*tran(L). Example: cholesky([[3,1],[1,4]]) → [[3/√(3),0],[1/√(3),(1/3)*√(33)]]
					ihermite	Hermite Normal Syntax: ihermite(Matrix_A) Given Matrix_A, returns the 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

Help Topics Tree	13217	Help Text
		Example: ihermite([[1,2,3],[4,5,6],[7,8,9]]) → [[-3,1,0],[4,-1,0],[-1,2,-1]],[[1,-1,-3],[0,3,6],[0,0,0]]
	hessenberg	Syntax: hessenberg(Matrix_A) Given Matrix_A, returns the matrix reduction to Hessenberg form. Returns [P,B] such that B=inv(P)*A*P. Example: hessenberg([[1,2,3],[4,5,6],[7,8,1]]) → [[[1,0,0],[0,4/7,1],[0,1,0]],[[1,29/7,2],[7,39/7,8],[0,278/49,3/7]]]
	ismith	Smith Normal Syntax: ismith(Matrix_A) Given Matrix_A, returns the Smith normal form of a matrix with coefficients in Z. Returns [U V B] such that U and V are invertible in Z, B is the diagonal, B[i,i] divides B[i+1,i+1] and B=U*A*V. Example: ismith([[1,2,3],[4,5,6],[7,8,9]]) → [[1,0,0],[4,-1,0],[-1,2,-1]],[[1,0,0],[0,3,0],[0,0,0]],[[1,-2,1],[0,1,-2],[0,0,1]]
Factorize		This menu lists the factorization matrix functions
	LQ	LQ Factorization Syntax: LQ(matrix) Factorizes a m × n matrix into three matrices: L, Q, and P, where L is an m × n lower trapezoidal, Q is an n × n orthogonal, and P is an m × m permutation; and P*A=L*Q. Example: LQ([[1,2],[3,4]]) → {[[2.2360,0],[4.9193,0.8944]],[[0.4472,0.8944],[0.8944,-0.4472]],[[1,0],[0,1]]}
	LSQ	Least Squares Syntax: LSQ(matrix1, matrix2) Returns the minimum norm least squares matrix (or vector) corresponding to the system matrix1*X=matrix2 Examples: LSQ([[1,2],[3,4]],[[5],[11]]) → [[1],[2]] LSQ([[1,2],[3,4]],[[5,-1],[11,-1]]) → [[1,1],[2,-1]]
	LU	LU Decomposition Syntax: LU(matrix) Factorizes a square matrix into three matrices L, U, and P, where L is a lowertriangular, U is an uppertriangular, and P is the permutation; and P*A=L*U. Example: LU([[1,2],[3,4]]) → {[[1,0],[0.3333,1]],[[3,4],[0,0.6666],[0,1],[1,0]]}
	QR	QR Factorization Syntax: QR(matrix) Factors an m x n matrix into three matrices: {[[m x m orthogonal]],[[m x n uppertrapezoidal]],[[n x n permutation]]}. Example: QR([[1,2],[3,4]]) → {[[0.3612,0.9486],[0.9486,-0.3162]],[[3.1622,4.4217],[0,0.6324]],[[1,0],[0,1]]}
	SCHUR	Schur Decomposition Syntax: SCHUR(matrix) Factors a square matrix into two matrices. If matrix is real, then the result is {[[orthogonal]],[[upper-quasi triangular]]}. If Complex mode is on and the matrix is complex, then the result is {[[unitary]],[[upper-triangular]]}. Example: SCHUR([[7,-2],[12,-3]]) → {[[0.4472,0.8944],[0.8944,−0.4472]],[[3,14],[0,1]]}
	SVD	Singular Value Decomposition Syntax: SVD(matrix) Factorizes an m × n matrix into two orthogonal matrices U (m x m) and V (n x n) and a vector S such that matrix = U*S'*trn(V). (S' is the diagonalization of S.) Example: SVD([[1,2],[3,4]]) → {[[0.9145,-0.4046],[-0.4046,0.9145]],[0.3660,5.4650],[[-0.8174,0.5760],[0.5760,0.8174]]}
	SVL	Singular Values Syntax: SVL(matrix) Returns a vector containing the singular values of matrix. Example: SVL([[1,2],[3,4]]) → [0.3660,5.4650]
Vector		This menu lists the vector functions

Help Topics Tree		13217	Help Text
		CROSS	Cross Product Syntax: CROSS(Vector1, Vector2) Returns the cross product two vectors. Examples: CROSS([1,2,3],[4,3,2]) → [-5,10,-5] CROSS([1+2*i,2-4*i,3+i],[−4+i,1-3*i,2+0.5*i]) → [i,−14-5.5*i,11-19*i] CROSS({[1,2,3],[4,3,2]},{[7,2,8],[9,1,6]}) → {[10,13,−12],[16,−6,−23]}
		DOT	Dot Product Syntax: DOT(Vector1, Vector2) Returns the dot product of two vectors. Examples: DOT([1,2],[3,4]) → 11 DOT({[1,2],[5,6]},{[3,4],[−6,−2]}) → {11,−42}
		maxnorm	Max Norm Syntax: maxnorm(Vector) or maxnorm(Matrix) Returns the l∞ norm (the maximum of the absolute values of the coordinates) of a vector or matrix. Examples: maxnorm([1,2]) → 2 maxnorm([[1,2],[3,-4]]) → 4
		l1norm	L^1 Norm Syntax: l1norm(Vector) Returns the L^1 norm (sum of the absolute values of the coordinates) of a vector. Example: l1norm([3,-4,2]) → 9
		l2norm	L^2 Norm Syntax: l2norm(Vector) Returns the L^2 norm (sqrt($x1^2+x2^2+…xn^2$)) of a vector. Example: l2norm([3,4,-2]) → √29
Special			This menu lists the special function linked with distributions functions
	Beta		Syntax: Beta(x, y) Returns the value of the Beta function for two values, x and y, defined as Gamma(x)*Gamma(y)/Gamma(x+y). Example: Beta(3,2) → 1/12
	erf		Error Function Syntax: erf(x) For a real value x, returns the approximate value of 2/√π*int(e^(-t^2),t,0,x) Example: erf(1) → 0.84270079295
	erfc		Complementary Error Function Syntax: erfc(x) For a real value x, returns the approximate value of 2/√π*int(exp(-t^2),t,x,∞). Example: erfc(1) → 0.15729920705
	Gamma		Gamma Function Syntax: Gamma(Real) Returns the value of the gamma function (Γ) for a real number. Gamma(n)=(n-1)! if n is an integer. Examples: Gamma(5) → 24 Gamma(1/2)
	Psi		Syntax: Psi(Real(a),Intg(n)) 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)). Example: Psi(3,1) → π^2/6-5/4
	Zeta		Syntax:

Help Topics Tree 13217	Help Text
	Zeta(x) Returns the value of the zeta function (Z) for a real x. Example: Zeta(2) → π^2/6
Ci	Cosine Integral Syntax: Ci(Expr) Returns the cosine integral of an expression. int(cos(t)/t,t=-∞..x). Example: Ci(1.0) → 0.337403922901
Ei	Exponential Integral Syntax: Ei(x) For a real value x, returns the approximate value of int(e^(t)/t, -∞, x) Example: Ei(1.0) → 1.89511781636
Si	Sine Integral Syntax: Si(Expr) Returns the sine integral of an expression, int(sin(t)/t,t=0..x) Example: Si(1.0) → 0.946083070367
CAS Menu	Toolbox CAS Menu The Toolbox CAS menu lists all the most useful Computer Algebra System (CAS) functions.
Algebra	Algebra Menu The Algebra menu contains common symbolic algebra commands, such as collect, expand, and factor.
simplify	Simplify Expression Syntax: simplify(Expr) Simplifies an expression. Example: simplify(4*atan(1/5)-atan(1/239)) → (1/4)*π
collect	Collect Like Terms Syntax: collect(Poly) or collect(Poly, Var) or collect({Poly1, Poly2,..., Polyn}) Collects like terms in a polynomial expression (or of a list of polynomial expressions). Factorizes the results, depending on the CAS settings. If specified, will collect with respect to Var. Examples: collect(x+2*x+1-4) → 3*x-3 collect(x^2-9*x+5*x+3+1) → (x-2)² collect(a*(b-c)+d*(b-c)) → (-c+b)*(a+d) collect(a*(b-c)+d*(b-c),a) → b*d-c*d+(b-c)*a
expand	Expand Expression Syntax: expand(Expr) Returns an expression expanded. Example: expand((x+y)*(z+1)) → y*z+x*z+y+x
factor	Factorize Polynomial Syntax: factor(Expr) Returns a polynomial factorized. Similar to collect, but will factor using square roots. Examples: factor(x^4+12*x^3+54*x²+108*x+81) → (x+3)^4 factor(x^4-1) → (x-1)*(x+1)*(x^2+1)
partfrac	Partial Fraction Decomposition Syntax: partfrac(RatFrac) Performs partial fraction decomposition on a fraction. Example: partfrac(x/(4-x²)) → (-1/2)/(x-2)-(1/2)/((x+2)
subst	Substitute Syntax: subst(Expr,Var=value)

Help Topics Tree		13217	Help Text
			Substitutes a value for a variable in an expression. Examples: subst(x/(4-x^2),x=3) → -3/5 subst(∫(sin(x^2)*x,x),x=√(t))
	Extract		This menu contains commands that allow you to extract one side of an equation or one part of a fraction.
		denom	Simplified Denominator Syntax: denom(a/b) For integers a and b, returns the denominator of the fraction a/b after simplification. Example: denom(10/12) → 6
		numer	Simplified Numerator Syntax: numer(a/b) For integers a and b, returns the numerator of the fraction a/b after simplification. Example: numer(10/12) → 5
		left	Left Side of Equation Syntax: left(Expr1=Expr2) or left(Real1..Real2) Returns the left side of an equation or the left end of an interval. Example: left(x^2-1=2*x+3) → x^2-1
		right	Right Side of Equation Syntax: right(Expr1=Expr2) or right(Real1..Real2) Right Side Returns the right side of an equation or the right end of an interval. Example: right(x^2-1=2*x+3) → 2*x+3
Calculus			Calculus Menu The Calculus menu contains operations pertaining to limits, differentiation, and integration.
	diff		Differentiate Syntax: diff(Expr,[Var,[Order]]) diff(Expr,[{Var1,Var2,…},[Order]]) Returns the derivative of an expression with respect to a given variable or list of variables. You can use the differentiation template in the Template menu as well. If Var or a list of variables is defined, a final parameter, Order, designates the order of the derivative to be found. Order defaults to 1. Examples: diff(x^3-x) → 3*x^2-1 diff(x^3-x,x,2) → 6*x diff(sin(x)-cos(y),x) → cos(x) diff(sin(x)-cos(y),y) → sin(y) diff(sin(x)-cos(y),{x,y}) → [cos(x) sin(y)] diff(sin(x)-cos(y),{x,y},2) → [[-sin(x),0],[0,cos(y)]]
	limit		Syntax: limit(Expr,Var,Val, [Dir]) Returns the limit (2-sided or 1-sided) of the given expression as the given variable approaches a value. The optional argument Dir indicates a two sided limit if 0, one sided from below if -1, and one sided from above if 1. If the fourth argument is not provided, the limit returned is bidirectional. Examples: limit((n*tan(x)-tan(n*x))/(sin(n*x)-n*sin(x)),x,0) → 2 limit(sin(x)/(x^2-3*x),x,0) → -1/3 limit(exp(1/x),x,0,1) → +∞
	int		Integrate Syntax: int(Expr,[Var],[Real1,Real2]) 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. Examples: int(1/x) → ln(abs(x)) int(sin(x),x,0,π) → 2

Help Topics Tree	13217	Help Text
		int(1/(1-x^4),x,2,3)) → -1/4*(2*atan(2)+ln(3))+1/4*(2*atan(3)-ln(2)+ln(4))
series		Series Expansion Syntax: series(Expr,Var=Val,[Order],[Dir]) Returns the series expansion of an expression in the vicinity of a given variable value. 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. The optional argument Dir is bidirectional if 0, one sided from below if -1, and one sided from above if 1. If no direction is specified, the series is bidirectional. Example: series((x^4+x+2)/(x^2+1),x=0,5) → $2+x-2x^2-x^3+3x^4+x^5+x^6$*order_size(x)
sum		Summation Syntax: sum(Expr,Var,Real1,Real2,[Step]) Returns the discrete sum of Expr with respect to the variable Var from Real1 to Real2. With only the first two arguments, returns the discrete antiderivative of the expression with respect to the variable. Examples: sum(n^2,n,1,5) → 55 sum(cos(n*x),n)
Differential		Differential Menu This sub-menu contains specialized vector operations based on differentiation, such as curl and grad, as well as the Laplace and inverse Laplace transforms.
	curl	Rotational Curl Syntax: curl([Expr1, Expr2, ... ExprN], [Var1, Var2, ... VarN]) 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]. Example: curl([2*x*y,x*z,y*z],[x,y,z]) → [z-x,0,z-2*x]
	divergence	Syntax: divergence([Expr1, Expr2, ... ExprN],[Var1, Var2, ... VarN]) Returns the divergence of a vector field, defined by divergence([A,B,C],[x,y,z])=dA/dx+dB/dy+dC/dz. Example: divergence([x^2+y,x+z+y,z^3+x^2],[x,y,z]) → 2*x+3*z^2+1
	grad	Gradient Syntax: grad(Expr, ListVars) Returns the gradient of an expression. With a list of variables as second argument, returns the vector of partial derivatives. Example: grad(2*x^2*y-x*z^3,[x,y,z]) → [-z^3+4*x*y 2*x^2 -3*x*z^2]
	hessian	Hessian Matrix Syntax: hessian(Expr,ListVar) Returns the Hessian matrix of an expression. Example: hessian(2*x^2*y-x*z,[x,y,z]) → [[4*y,4*x,-1],[4*x,0,0],[-1,0,0]]
Integral		Integral Menu This menu contains specialized operations based on integration, such as integration by parts.
	ibpdv	Integration By Parts v Syntax: ibpdv(f(Var), v(Var), [Var], [Real1], [Real2]) 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. Examples: ibpdv(ln(x),1) → x*ln(x)-x ibpdv(ln(x),x) → [x*ln(x), -1]
	ibpu	Integration By Parts u Syntax: ibpu(f(Var), u(Var), [Var], [Real1], [Real2]) 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, it 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.

Help Topics Tree	13217	Help Text
		Example: ibpu(x*ln(x), x) → [x*(x*ln(x)-x), -x*ln(x)+x]
	preval	Syntax: preval(F(var),Real(a),Real(b),[Var]) Returns F(b) – F(a). Examples: preval(x^2+x,2,3) → 6 preval(y^2-2,2,3,y) → 5
Limits		Limits Menu This sub-menu contains specialized operations involving limits, such as Taylor polynomials.
	taylor	Taylor Series Expansion Syntax: taylor(Expr,[Var=Value],[Order]) Returns the Taylor series expansion of an expression at a point or at infinity (by default, at x=0 and with relative order=5). Examples: taylor(sin(x)/x,x=0) → 1-(1/6)*x^2+(1/120)*x^4+x^6*order_size(x) taylor((x^4+x+2)/(x^2+1),x,5)
	divpc	Taylor of Quotient Syntax: divpc(Poly1, Poly2, Integer) Returns the n-degree Taylor polynomial for the quotient of 2 polynomials. Example: divpc(x^4+x+2,x^2+1,5) → x^5+3*x^4-x^3-2*x^2+x+2, the 5th-degree polynomial
	sum_riemann	Riemann Sum Syntax: sum_riemann(Expr(Xpr),Lst(var1,var2)) Returns in the neighborhood of n=+∞ an equivalent of the sum of Xpr(var1,var2) for var2 from var2=1 to var2=var1 when the sum is looked at as a Riemann sum associated with a continuous function defined on [0,1]. Examples: sum_riemann(1/(n+k),[n,k]) → ln(2) sum_riemann(n/(n^2+k^2),[n,k]) → π/4
Transform		Transform Menu This menu contains Laplace and Fourier Transform commands.
	laplace	Laplace Transform Syntax: laplace(Expr,[Var],[LapVar]) Returns the Laplace transform of an expression. Examples: laplace(e^(x)*sin(x)) → 1/(x^2-2*x+2) laplace(sin(x)^2,x,s) → 2/(s^3+4*s)
	invlaplace	Inverse Laplace Transform Syntax: invlaplace(Expr,[Var],[IlapVar]) Returns the inverse Laplace transform of an expression. Example: invlaplace(1/(x^2+1)^2) → (-x/2)*cos(x)+(1/2)*sin(x)
	fft	Fast Fourier Transform Syntax: fft(Vector) or fft(Vector, a, p) With one argument (a vector), returns the discrete Fourier transform in R. With two additional integer arguments a and p, returns the discrete Fourier transform in the field Z/pZ, with a primitive nth root of 1 (n=size(Vector)). Example: fft([1,2,3,4,0,0,0,0]) → [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]
	ifft	Inverse Fast Fourier Transform Syntax: ifft(Vect) Returns the inverse discrete Fourier transform. 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]) → [0.99999999999,3.99999999999,10.0,20.0,25.0,24.0,16.0,-6.39843733552e-12]
Solve		Solve Menu The Solve menu contains the various commands for solving equations.
solve		CAS solve Syntax:

Help Topics Tree 13217	Help Text
	solve(Expr,[Var]) or solve({Eq1, Eq2,…}, [Var]) or solve(Expr, Var=Guess) or solve(Expr, Var=Val1..Val2) Returns a list of the solutions (real and complex) to a polynomial equation or a set of polynomial equations. The user is advised to supply a guess or define an interval in which to search for a solution to get the best results in cases where the solution is known to be approximate. To supply a guess, use the syntax Var=Guess. To supply an interval, use the syntax Var=Val1..Val2. In the latter case, the search is confined to the closed interval [Val1, Val2]. Examples: solve(x^2-3=1) → {-2,2} solve([x^2-y^2=0,x^2-z^2=0],[x,y,z]) → {[x,x,x],[x,-x,-x],[x,x,-x],[x,-x,x]} solve(x^2-(LN(x)+5)=0, x=2) → 2.42617293082 solve(x^2-(LN(x)+5)=0, x=2..3) → 2.42617293082
zeros	Syntax: zeros(Expr,[Var]) or zeros([Expr1, Expr2, ... Exprn], {Var1, Var2, ... Varn}) Returns the zeros (real or complex according to the CAS settings) of the expression Expr for the variable Var (or the matrix where the lines are the solutions of the system : Expr1=0, Expr2=0…). Examples: zeros([x^2-1,x^2-y^2],[x,y]) → [[1,1],[1,-1],[-1,1],[-1,-1]] zeros(x^2+4) → [-2*i,2*i] if Use i is checked in CAS Settings and [ ] otherwise.
cSolve	Complex Solve Syntax: cSolve(Expr,[Var]) Returns the solutions, including complex solutions, of Expr, for Var. If Expr is an expression, solves the equation Expr=0. Examples: cSolve(x^4=1,x) → {-1,i,1,−i} cSolve(u*v-u=v and v^2=u,[u,v]) → {[0,0],[(1/2*(√5+1))^2,1/2*(√5+1)],[(1/2*(-√5+1))^2,1/2*(-√5+1)]}
cZeros	Complex Zeros Syntax: cZeros(Expr,[Var]) or cZeros({Expr1, Expr2, ... ExprN}, {Vr1, Vr2, ... VarN}) Returns the roots, including complex roots, of Expr (that is, the solution of Expr=0) or the matrix where the lines are the solutions of the system: Expr1=0, Expr2=0…ExprN=0. Examples: cZeros(x^4-1) → [1,-1, i, -i] cZeros([x^2-1,x^2-y^2],[x,y])
fsolve	Numerical Solve Syntax: fsolve(Expr,Var,[Guess or Interval],[Method]) fsolve(ExprVector, [Guess or Interval], [Method}) 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. If you are solving for a single variable, your options are bisection_solver, newton_solver, or newtonj_solver. If solving for 2 variables, your only option is newton_solver. Examples: fsolve(cos(x)=x,x,-1..1) → [0.739085133215] fsolve([x^2+y-2,x+y^2-2],[x,y],[0,0]) → [1.,1.]
desolve	Solve Differential Equation Syntax: desolve(Eq,[TimeVar],Var) Returns the solution to a differential equation. Examples: desolve(y''+y=0,y) → G_0*cos(x)+G_1*sin(x) desolve((y''+y=sin(x)) and (y(0)=1) and (y'(0)=2),y)
odesolve	ODE Solver Syntax: odesolve(Expr, VectVar, VectInit, FinalVal, [tstep=Val, curve]) Ordinary Differential Equation solver Solves an ordinary differential equation given by Expr, with variables declared in VectVar and initial conditions for those variables declared in VectInit. 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. Example: odesolve(sin(t*y),[t,y],[0,1],2) → [1.82241255674]
linsolve	Linear System Solver Syntax:

Help Topics Tree				13217	Help Text
					linsolve([LinEq1, LinEq2,…LinEqn], [Var1,Var2,…Varn]) Given a vector of linear equations and a corresponding vector of variables, returns the solution to the system of linear equations. Example: linsolve([x+y+z=1,x-y=2,2*x-z=3],[x,y,z]) → [3/2,-1/2,0]
		Rewrite			Rewrite Menu The Rewrite menu contains commands for rewriting or simplifying expressions by using a variety of means, including trigonometric identities, etc.
			lncollect		Collect Logarithms Syntax: lncollect(Expr) Rewrites an expression with the logarithms collected. Applies ln(a)+n*ln(b)=ln(a*b^n) where n is an integer. Example: lncollect(ln(x)+2*ln(y)) → ln(x*y^2)
			powexpand		Power Expand Syntax: powexpand(Expr) 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). Example: powexpand(2^(x+y)) → (2^x)*(2^y)
			texpand		Transcendental Expand Syntax: texpand(Expr) Expands a transcendental expression; that is, an expression containing trigonometric, logarithmic, or exponential functions. texpand develops the expression in terms of sin(), cos(), ln(), and exp(). Examples: texpand(sin(2*x)+exp(x+y)) → 2*cos(x)*sin(x)+e^(x)*e^(y) texpand(cos(3*x))
			Exp and Log		This menu contains commands for converting expressions into equivalent expressions using various identities involving powers, logarithms, and exponents.
				exp2pow	Syntax: exp2pow(Expr) Transforms an expression of the form e^(n*ln(x)) rewritten as a power of x. Applies e^(n*ln(x))=x^n. Example: exp2pow(e^(3*ln(x))) → x^3
				pow2exp	Syntax: pow2exp(Expr) Returns an expression with powers rewritten as an exponential. Essentially the inverse of exp2pow. Example: pow2exp(a^b) → e^(b*ln(a))
				exp2trig	Syntax: exp2trig(Expr) Returns an expression with complex exponentials rewritten in terms of sine and cosine. Example: exp2trig(exp(-i*x)) → cos(x)+ i*sin(x)
				expexpand	Expand Exponentials Syntax: expexpand(Expr) Expands exponentials using the identity e^(a*f(x))=e^(f(x))^a. Example: expexpand(e^(3*x)) → $(e^x)^3$
			Sin to ...		This menu contains commands for converting expressions containing the inverse sine function into equivalent expressions containing other inverse trigonometric functions.
				asin2acos	Syntax: asin2acos(Expr) Replaces arcsin(x) by π/2-arccos(x) in Expr. Example: asin2acos(acos(x)+asin(x)) → π/2-acos(x)+acos(x)

				asin2atan	Syntax:
					asin2atan(Expr)
					Replaces arcsin(x) by arctan(x/√(1-x^2)) in Expr.
					Examples:
					asin2atan(2*asin(x)) → 2*atan(x/(√(1-x^2)))
					asin2atan(asin(√(1-x^2))+asin(x))

				sin2costan	Syntax: sin2costan(Expr)

Help Topics Tree					13217	Help Text
						Rewrites Expr so that sin(x) is replaced by cos(x)*tan(x) Example: sin2costan(sin(x)) → tan(x)*cos(x)
				Cos to ...		This menu contains commands for converting expressions containing the inverse cosine function into equivalent expressions containing other inverse trigonometric functions.

					acos2asin	Syntax:
						acos2asin(Expr)
						Replaces arccos(x) by π/2-arcsin(x) in the argument Expr.
						Examples:
						acos2asin(acos(x)+asin(x)) → π/2-asin(x)+asin(x)
						acos2asin(2*acos(x))

					acos2atan	Syntax:
						acos2atan(Expr)
						Replaces arccos(x) by π/2-arctan(x/√(1-x^2)) in the argument.
						Examples:
						acos2atan(2*acos(x)) → 2*(π/2-atan(x/(√(1-x^2))))
						acos2atan(acos(√(1-x^2))+acos(x))

					cos2sintan	Syntax: cos2sintan(Expr) Replaces cos(x) by sin(x)/tan(x) in the argument. Example: cos2sintan(cos(x)) → sin(x)/tan(x)
				Tan to ...		This menu contains commands for converting expressions containing the inverse tangent function into equivalent expressions containing other inverse trigonometric functions.
					atan2asin	Syntax: atan2asin(Expr) Replaces arctan(x) by arcsin(x/√(1+x^2)) in the argument Expr. Example: atan2asin(atan(y/x) → asin((y/x)/√(1+$(y/x)^2$))
					atan2acos	Syntax: atan2acos(Expr) Replaces arctan(x) by π/2-arccos(x/√(1+x^2)) in the argument. Example: atan2acos(atan(2*x) → π/2-acos((2*x)/√(1+$(2*x)^2$))
					tanx → sinx/cosx	Syntax: tan2sincos(Expr) Rewrites Expr with tan(x) using sin(x)/cos(x) Example: tan2sincos(tan(x)) → sin(x)/cos(x)

					halftan	Syntax:
						halftan(Expr)
						Transforms sin(x), cos(x) and tan(x) as a function of tan(x/2).
						Examples:
						halftan(sin(x)) → (2*TAN(x/2))/($(TAN(x/2))^2$+1)
						halftan(tan(x)) → (2*TAN(x/2))/(-$(TAN(x/2))^2$+1)

				Trigonometry to ...		This menu contains commands for converting expressions containing the various transcendental functions into equivalent expressions containing other transcendental functions.
					trigx → sinx	Syntax: trigsin(Expr) 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. Example: trigsin(cos(x)^4+$sin(x)^2$) → $sin(x)^4$-$sin(x)^2$+1
					trigx → cosx	Syntax: trigcos(Expr) 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. Example: trigcos(sin(x)^4+sin(x)^2) → $cos(x)^4$-3*$cos(x)^2$+2
					trigx → tanx	Syntax: trigtan(Expr) 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. Example: trigtan(cos(x)^4+$sin(x)^2$) → ($tan(x)^4$+$tan(x)^2$+1)/($tan(x)^4$+2*$tan(x)^2$+1)
					atrig2ln	Syntax: atrig2ln(Expr) Returns an expression with inverse trigonometric functions rewritten using the natural logarithm function.

Help Topics Tree		13217	Help Text

			Examples:
			atrig2ln(atan(x)) → 0.5*i*ln((x+i)/(-x+i))
			atrig2ln(acos(x)) → −i*ln(x+√(x^2-1))

		tlin	Syntax:
			tlin(Expr)
			Returns a trigonometric expression with the products and integer powers linearized
			Examples:
			tlin(sin(x)^3) → 3/4*sin(x)-1/4*sin(3.*x)
			tlin(cos(x)*cos(y)) → 1/2*cos(x+y)+1/2*cos(x-y)

		tcollect	Syntax: tcollect(Expr) Returns a trigonometric expression linearized and with any sine and cosine terms of the same angle collected together. Example: tcollect(sin(x)+cos(x)) → √2*cos(x-1/4*π)
		trigexpand	Syntax: trigexpand(Expr) Returns a trigonometric expression in expanded form. Example: trigexpand(sin(3*x)) → (4*cos(x)^2-1)*sin(x)
		trig2exp	Syntax: trig2exp(Expr) Returns an expression with trigonometric functions rewritten as complex exponentials (without linearization). Example: trig2exp(sin(x)) → (e^(i*x)-(1/e^(i*x)))/(2*i)
Integer			Integer Menu The Integer menu contains operations on integers.
	idivis		Integer Divisors Syntax: idivis(Integer) or idivis({Intgr1, Intgr2, ... Intgrn}) Returns a list of all the factors of an integer or of a list of integers. Example: idivis(12) → [1, 2, 3, 4, 6, 12]
	ifactor		Integer Factors Syntax: ifactor(Integer) Returns the prime factorization of an integer as a product. Can be used with STO▸. Note: in some cases, factorization may fail. In these cases, the command will return the product of -1 and the opposite of the original input. The -1 indicates that factorization failed. Example: ifactor(150) → 2*3*5^2
	ifactors		Integer Factors List Syntax: ifactors(Integer) Similar to ifactor, but returns a list of the factors of the integer with their multiplicities. Example: ifactors(150) → [2, 1, 3, 1, 5, 2]
	igcd		Integer GCD Syntax: igcd(Intgr1, Intgr2, ... Intgrn)) Returns the integer that is the greatest common divisor of two or more integers. Example: igcd(24,36) → 12
	lcm		Lowest Common Multiple Syntax: lcm(Intgr1, Intgr2, ...) or lcm(Poly1, Poly2, ...) or lcm(Rational1, Rational2, ...) Returns the lowest common multiple of two or more polynomials of several variables, or of two or more integers, or of two or more rationals. Examples: lcm(6,4) → 12 lcm(x^2-2*x+1,x^3-1) → (x-1)*(x^3-1)
	Prime		The Prime sub-menu contains operations related to prime numbers.
		isprime	Primality Test Syntax:

Help Topics Tree				13217	Help Text
					isprime(Integer) Returns true if the integer is prime; otherwise, returns false. Examples: isprime(1999) → 1 isprime(42) → 0
				ithprime	Ith Prime Syntax: ithprime(Integer) Given an integer n, returns the nth prime number, where n is between 1 and 200,000. Example: ithprime(5) → 11
				nextprime	Next Prime Syntax: nextprime(Integer) Returns the smallest prime number greater than the argument. Example: nextprime(12) → 13
				prevprime	Previous Prime Syntax: prevprime(Integer) Returns the greatest prime number less than the argument. Example: prevprime(11) → 7
				euler	Euler's Totient Syntax: euler(Integer); Euler's phi (or totient) function Takes a positive integer and returns the number of positive integers less than or equal to it that are coprime to it. Example: euler(6) → 2
			Division		The Division sub-menu contains operations related to integer division.
				iquo	Integer Euclidian Quotient Syntax: iquo(Intgr1, Intgr2) Returns the integer quotient of the Euclidean division of two integers. Examples: iquo(148,5) → 29 iquo(25+12*i,5+7*i) → 3-2*i
				irem	Integer Euclidian Remainder Syntax: irem(Intgr1, Intgr2) Returns the integer remainder from the Euclidean division of two integers. Examples: irem(148,5) → 3 irem(25+12*i,5+7*i) → -4+i
				powmod	Integer Power and Modulo Syntax: powmod(a, n, p, [Expr, Var]) For the integers a, n, and p, returns a^n mod p. Examples: powmod(5,2,13) → 12 powmod(x+1,452,19,x^4+x+1,x) → 6*x^3+5*x^2-7*x-7
				ichinrem	Integer Chinese Remainder Syntax: ichinrem([a,p],[b,q])) Integer Chinese Remainder Theorem for two equations. Takes two lists [a, p] and [b, q] and returns a list 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. Example: ichinrem([2,7],[3,5]) → [23,35]
		Polynomial			Polynomial Menu The Polynomial sub-menu contains commands related to polynomials.
			proot		Polynomial Roots Syntax: proot(Poly) or proot(Vector) Returns all computed roots of a polynomial given by its coefficients (may not work if roots are not simple). Examples: proot([1,0,-2]) → [−1.41421356237,1.41421356237]

Help Topics Tree	13217	Help Text
		proot([1,2,-25,-26,120]) → [-5.,-3.,2.,4.]
coeff		Coefficients of Polynomial Syntax: coeff(Expr, [Var], [Integer]) Returns the list of coefficients of a polynomial with respect to the second argument or the coefficient of the term whose degree is Integer. Examples: coeff(x^3+2) → [1,0,0,2] coeff(2*y^2-3,y,0) → -3
divis		Polynomial Divisors Syntax: divis(Poly) or divis({Poly1, Poly2,…Polyn}) Given a polynomial or list of polynomials, returns a vector containing the divisors of the polynomial. Example: divis(x^2-1) → [1,x-1,x+1,(x+1)*(x-1)]
factors		Polynomial Factor List Syntax: factors(Poly) or factors({Poly1, Poly2, ..., Polyn}) Returns the list of prime factors of a polynomial; each factor followed by its multiplicity. Examples: factors(x^4-1) → [x-1,1,x+1,1,x^2+1,1] factors([x^2,x^2-1])
gcd		Greatest Common Divisor Syntax: gcd(Poly1, Poly2,…Polyn) or gcd(Intgr1, Intgr2,…Intgrn) Returns the greatest common divisor of two or more polynomials or the greatest common divisor of two or more integers. Examples: gcd(x^2-4,x^2-5*x+6) → x-2 gcd(45,30) → 15
Create		This sub-menu contains commands for creating polynomials, either randomly or with specific properties.
	symb2poly	Polynomial to Coefficients Syntax: symb2poly(Expr,[Var]) or symb2poly(Expr, {Var1, Var2, ... Varn}) 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. Essentially the inverse of poly2symb(). Examples: symb2poly((x+2)*x+3) → [1,2,3] symb2poly(3*x*y+2*y+1,x,y) → [[3,0],[2,1]] symb2poly(3*x*y+2*y+1,y,x) → [[3,2],1] symb2poly(3*x*y+2*y+1,{y,x}) → %%%{3,[1,1]%%%}+%%%{2,[1,0]%%%}+%%%{1,[0,0]%%%}
	poly2symb	Coefficients to Polynomial Syntax: poly2symb(Vector,[Var]) 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. Examples: poly2symb([1,2,3]) → x*(x+2)+3 poly2symb([1,2,-1],y) → y*(y+2)-1 poly2symb([1,2,3],x=2) → x*(x+2)+3=11
	pcoeff	Roots to Coefficients Syntax: pcoeff(Vector) or pcoeff(List) Given a list or vector containing the roots of a polynomial, returns a vector containing the coefficients (in decreasing order) of the univariate polynomial having those roots. Examples: pcoeff({1,0,0,0,1}) → [1,-2,1,0,0,0] pcoeff([1,0,-2]) → [1,1,-2,0]
	fcoeff	Roots to Polynomial Syntax: fcoeff([Root1, Order1, Root2, Order2, ..., Rootn, Ordern])

Help Topics Tree	13217	Help Text
		Returns the polynomial described by a list of roots, each followed by its order. Example: fcoeff([1,2,0,1,3,-1]) → x*(x-1)^2/(x-3)
	randpoly	Random Polynomial Syntax: randpoly([Var], Integer, [Interval, Dist]) Returns a vector of coefficients of a polynomial of variable Var (or x), 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. Example: randpoly(t,8,-1..1) returns a vector of 9 random integers, all of them between -1 and 1.
	pmin	Minimal Polynomial Syntax: pmin(Matrix,[Var]) 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. Example: pmin([[1,0],[0,1]],x) → x-1
Algebra		Among other operations, this menu contains the polynomial equivalents of some of the commands found in the Integer menu.
	quo	Quotient Syntax: quo(List1, List2, [Var]) or quo(Poly1, Poly2, [Var]) 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. Examples: quo([1,2,3,4],[-1,2]) → poly1[-1,-4,-11] quo(t^3+2t^2+3t+4,-t+2,t)
	rem	Remainder Syntax: rem(Poly1, Poly2, [Var]) or rem(List1, List2, [Var]) 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. Examples: rem(x^3+2x^2+3x+4,-x+2) → 26 rem([1,2,3,4],[-1,2]) → [26]
	degree	Degree of Polynomial Syntax: degree(Poly) Returns the degree of a polynomial. Examples: degree(x^3+x) → 3 degree([1,0,1,0]) → 3
	factor_xn	Factor by Degree Syntax: factor_xn(Poly) For a given polynomial in x of degree n, factors out x^n and returns the resulting product. Examples: factor_xn(x^4-1) → x^4*(1-x^-4) factor_xn(x^4+12*x^3+54*x^2+108*x+81)
	content	Coefficient GCD Syntax: content(Poly,[Var]) Returns the greatest common divisor (GCD) of the coefficients of a polynomial. Example: content(2*x^2+10*x+6) → 2
	sturmab	Zero Count Syntax: sturmab(Poly,[Var,Interval) If Interval real, this returns the number of sign changes in the specified polynomial in the interval. If the interval is complex, it returns the number of complex roots in the rectangle bounded by the interval. If Var is omitted, it is assumed to be x. Examples: sturmab(x^3-1,x,-2-i,5+3i) → 3 sturmab(x^3-1,x,-2,5) → 1
	chinrem	Chinese Remainder Syntax:

Help Topics Tree	13217	Help Text
		chinrem(Matrix_2xn) Given a matrix whose 2 rows each contain the coefficients of a polynomial, returns the Chinese remainder of those polynomials, also written as a matrix. Example: chinrem([[1,2,0],[1,0,1]],[[1,1,0],[1,1,1]]) → [[2,2,1] [1,1,2,1,1]]
Special		Syntax: chinrem(Matrix_2xn) Given a matrix whose 2 rows each contain the coefficients of a polynomial, returns the Chinese remainder of those polynomials, also written as a matrix. Example: chinrem([[1,2,0],[1,0,1]],[[1,1,0],[1,1,1]]) → [[2,2,1] [1,1,2,1,1]]
	cyclotomic	Syntax: cyclotomic(Integer) Generates a vector representing the nth cyclotomic polynomial. Example: cyclotomic(20) → [1,0,-1,0,1,0,-1,0,1]
	gbasis	Groebner Basis Syntax: gbasis([Poly1, Poly2,...], [Var1, Var2, ...]) Given a vector of polynomials and a vector of variables, returns the Groebner basis of the ideal spanned by the set of polynomials. Example: gbasis([x²-y^3,x+y²],[x,y]) → [x*y+x^2,y^2+x]
	greduce	Groebner Remainder Syntax: greduce(Poly1, [Poly2, Poly3,...], [Var1, Var2, ...]) 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. Examples: greduce(x*y-1,{x²-y²,2*x*y-y²,y^3},{x,y}) → (1/2)*y²-1 greduce(x1²*x3²,[x3^3-1,-x2²-x2*x3-x3²,x1+x2+x3],[x1,x2,x3]) → x2
	hermite	Hermite Polynomial Syntax: hermite(Integer) Returns the Hermite polynomial of degree n, where n is an integer less than 1556. Example: hermite(3) → 8*x³-12*x
	lagrange	Lagrange Polynomial Syntax: lagrange([X1, X2,... Xn], [Y1, Y2, ... Yn]) or lagrange(Matrix) 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. Returns the polynomial of degree n-1 such that P(xk)=yk, for k=0, 1, …, n-1. Example: lagrange([[1,3],[0,1]]) → (1/2)*(x-1)
	laguerre	Laguerre Polynomial Syntax: laguerre(Integer) Given an integer n, returns the Laguerre polynomial of degree n. Example: laguerre(2) → -a*x+1/2*a^2+1/2*x^2+3/2*a-2*x+1
	legendre	Legendre Polynomial Syntax: legendre(Integer) Given an integer n, returns the Legendre polynomial of degree n. Example: legendre(4) → 35/8*x^4-15/4*x^2+3/8
	lll_reduce	LLL Reduction Syntax: lll_reduce(Matrix) Implementation of the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm. Takes as argument an invertable matrix with integer coefficients. Returns (S, A, L, O) such that: • the rows of S is a short basis of the Z-module generated by the rows of M • A is the change-of-basis matrix from the short basis to the basis defined by the rows of M(A*M=S) • L is a lower triangular matrix and the modulus of it’s non diagonal coefficients are less than 1/2

Help Topics Tree				13217	Help Text
					• O is a matrix with orthogonal rows such that L * O = S Example: lll_reduce([[1234,3452,4521],[3425,2241,1543],[5643,3425,8721]])
				nop	No Operation The no-operation CAS function. On evaluation, no operation will happen. This function can be useful for some advanced use cases in CAS function programming.
				tchebyshev1	Chebyshev Tn Syntax: tchebyshev1(Integer) Returns the nth Tchebyshev polynomial of the first kind. Example: tchebyshev1(3) → $4*x^3-3*x$
				tchebyshev2	Chebyshev Un Syntax: tchebyshev2(Integer) Returns the nth Tchebyshev polynomial of the second kind. Example: tchebyshev2(3) → $8*x^3-4*x$
			gcd		Greatest Common Divisor Syntax: gcd(Poly1, Poly2) or gcd(Integer1, Integer2) Returns the greatest common divisor of 2 polynomials of several variables. Can also be used as integer gcd. Examples: gcd(x^2-4,x^2-5*x+6) → x-2 gcd(45,30) → 15
			is_cycle		is_cycle Function Syntax: is_cycle(list) Tests whether or not list is a cycle. Returns 1 if it is, and 0 otherwise. Examples: is_cycle([2,1,3,5]) → 1 is_cycle([2,0,3,2]) → 0
			is_permu		is_permu Function Syntax: is_permu(list) Tests whether or not list is a permutation. Returns 1 if it is, and 0 otherwise. Examples: is_permu([3,1,5,4,2]) → 1 is_permu([3,1,5,4]) → 0
			groupermu		Syntax: groupermu(permutation1,permutation2) Returns the group of permutations generated by permutation1 and permutation2. Example: groupermu([2,1],[2,3,1])
			lcm		Lowest Common Multiple Syntax: lcm(Intgr1, Intgr2, ...) or lcm(Poly1, Poly2, ...) or lcm(Rational1, Rational2, ...) Returns the lowest common multiple of two or more polynomials of several variables, or of two or more integers, or of two or more rationals. Examples: lcm(6,4) → 12 lcm(x^2-2*x+1,x^3-1) → (x-1)*(x^3-1)
		Plot			Plot Menu The Plot menu contains operations that allow drawing plots in the CAS.
			plotcontour		Plot Contour Syntax: plotcontour(Expr,[LstVar],[LstVal]) Draws contour-lines z=z_min, …z=z_max of the surface z=Expr, where the contour-lines are defined by the 3rd argument. Used in the Geometry app Plot or Symbolic views or CAS view. Example: plotcontour(x^2+2*y^2-2,[x,y],[1.0,2.0,3.0]) draws three contour-lines for the given expression
			plotfunc		Plot Function Syntax: plotfunc(Expr)

Help Topics Tree 13217	Help Text
	Used in the Geometry app Plot or Symbolic views, or in CAS view. Draws the plot of a function, given an expression in the independent variable x. Note the use of lowercase x. Example: plotfunc(3*sin(x)) draws the graph of y=3*sin(x).
plotimplicit	Plot Implicit Syntax: plotimplicit(Expr, [XIntrvl, YIntrvl]) Used in the Geometry app Plot or Symbolic views, or CAS view. 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. Examples: plotimplicit((x+5)²+(y+4)²-1,[x=-6..-4,y=-5..-3]) plotimplicit((x+5)²+(y+4)²-1) plots a circle, centered at the point (-5, -4), with a radius of 1
plotfield	Plot Slopefield Syntax: plotfield(Expr, VectorVar, [xstep=Val, ystep=Val, Option]) Used in the Geometry app or CAS view. 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. 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.
plotode	Plot ODE Syntax: plotode(Expr, [Var1, Var2, ...], [Val1, Val2. ...], [tstep=Value]) Used in the Symbolic or Plot views of the Geometry app or in CAS view. 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. The optional tstep can be used to control the level of detail of the plot. Examples: 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. plotode(5*[-y,x],[t=0..1,x,y],[0,0.3,0.7],tstep=0.5,plan)
plotlist	Plot List Syntax: plotlist(Matrix) Used in the Plot or Symbolic views of the Geometry app, or CAS view, this command plots a set of n points and connects them with segments. The points are defined by a m x 2 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
App Menu	Toolbox App Menu The Toolbox App menu lists the app-specific functions.
User Menu	Toolbox User Menu The Toolbox User menu lists all the functions and programs you have created yourself. These will be grouped together under the name of the source file that contains the exported variables or functions.
Catlg Menu	Toolbox Catalog Menu The Toolbox Catalog menu lists all the functions and commands in the system. On the right side of the Catalog header is a small information icon (i). Tap the icon to see the number of each type of function currently defined on your HP Prime (CAS, App, User, and so on).
A-E	Function Catalog A-E Toolbox function catalog A-E
:=	Assign Syntax: variable := object Assigns object to variable. Examples: A := 3 stores the value 3 in the variable A F1 := 3-X makes F1(X)=3-X M5 := [1, 2] stores a vector in M5
a2q	Syntax: a2q(Matrix, [Var1, Var2….]) Given a symmetric matrix and a vector of variables, returns the quadratic form of the matrix using the variables in the vector. Example: a2q([[1,2],[4,4]],[x,y]) → 6*x*y+x^2+4*y^2

Help Topics Tree	13217	Help Text
abcuv		Syntax: abcuv(Poly_A,Poly_B,Poly_C,[Var]) 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. Example: abcuv(x^2+2*x+1,x^2-1,x+1) → [1/2,-1/2]
about		Syntax: about(Var) Returns the hypothesis made with the assume and additionally commands on the variable Var. Example about(n) returns any conditions imposed on the variable n.
ABS		Absolute Value Syntax: ABS(expr) or ABS(matrix) For numerical arguments, returns the absolute value of the expression. For matrix arguments, returns the Frobenius (Euclidean) norm of the array. Examples: ABS(-3.14) → 3.14 ABS([[1,2],[3,4]]) → 5.47722557505 ABS(2-3*i) → 3.60555127546 CAS(ABS([[1,2],[3,4]])) → √30
abscissa		Syntax: abscissa(Point) or abscissa(Vector) Returns the abscissa of a point or a vector. Examples: abscissa(point(1+2*i)) → 1 abscissa(point(1,2,3))
ACOS		Inverse Cosine Syntax: ACOS(Value) Returns the inverse cosine of Value. The output depends on the Angle Measure setting in Home Settings, CAS Settings, or Symbolic Setup. Example: ACOS(0.5) → 60 (Degrees mode) ACOS(0.833730025131-0.988897705763*i) → 1+i ACOS({0.5,1}) → {60,0} (Degrees mode)

acos2asin		Syntax:
		acos2asin(Expr)
		Replaces arccos(x) by π/2-arcsin(x) in the argument Expr.
		Examples:
		acos2asin(acos(x)+asin(x)) → π/2-asin(x)+asin(x)
		acos2asin(2*acos(x))

acos2atan		Syntax:
		acos2atan(Expr)
		Replaces arccos(x) by π/2-arctan(x/√(1-x^2)) in the argument.
		Examples:
		acos2atan(2*acos(x)) → 2*(π/2-atan(x/(√(1-x^2))))
		acos2atan(acos(√(1-x^2))+acos(x))

ACOSH		Inverse Hyperbolic Cosine Syntax: ACOSH(value) Inverse Hyperbolic Cosine: COSH^-1 (X) Examples: ACOSH(1.54308063482) → 1 ACOSH(0.833730025131+0.988897705763*i) → 1+i ACOSH({1,1.54308063482}) → {0,1}
ACOT		Arc Cotangent Syntax: ACOT(value) Inverse Cotangent: COT^-1 (X) Example: ACOT(1) → 45 (Degrees mode) ACOT(0.217621561854-0.868014142896*i) → 1+i ACOT({1,0}) → {45,90} (Degrees mode)
ACSC		Arc Cosecant

Help Topics Tree	13217	Help Text
		Syntax: ACSC(value) Inverse Cosecant: CSC^-1 (X) Example: ACSC(1) → 90 (Degrees mode) ACSC(0.621518017169-0.303931001627*i) → 1+i ACSC({2,1}) → {30,90} (Degrees mode)
ADDCOL		Add Column Syntax: ADDCOL(matrixname, vector, column_number) Inserts values from vector into a column before column_number in the specified matrix. The size of vector must be the same as the number of rows in the matrix matrixname. Examples: ADDCOL([[1,3],[4,6]],[2,5],2) → [[1,2,3],[4,5,6]] ADDCOL([[1,3],[4,6]],{[2,5],[3,4]},{2,1}) → {[[1,2,3],[4,5,6]],[[3,1,3],[4,4,6]]} ADDCOL({[[1,3],[4,6]],[[1,9],[5,6]]},[2,5],2) → {[[1,2,3],[4,5,6]],[[1,2,9],[5,5,6]]} ADDCOL({[[1,3],[4,6]],[[1,9],[5,6]]},{[2,5],[3,4]},{2,1}) → {[[1,2,3],[4,5,6]],[[3,1,9],[4,5,6]]}
additionally		Syntax: additionally(Expr) Used in programming with assume( ) to state an additional assumption about a variable. Example: assume(n,integer); additionally(n>5); → [DOM_INT, n]
ADDROW		Add Row Syntax: ADDROW(matrixname, vector, row_number) Inserts values from vector into a row before row_number in the specified matrix. The size of vector must be the same as the number of columns in the matrix matrixname. Examples: ADDROW([[1,2],[5,6]],[3,4],2) → [[1,2],[3,4],[5,6]] ADDROW([[1,2],[5,6]],{[3,4],[2,5]},2) → {[[1,2],[3,4],[5,6]],[[1,2],[2,5],[5,6]]} ADDROW({[[1,3],[4,6]],[[1,9],[5,6]]},[2,5],2) → {[[1,3],[2,5],[4,6]],[[1,9],[2,5],[5,6]]} ADDROW({[[1,3],[4,6]],[[1,9],[5,6]]},{[2,5],[3,4]},{2,1}) → {[[1,3],[2,5],[4,6]],[[3,4],[1,9],[5,6]]}
adjoint_matrix		Adjoint Matrix Syntax: adjoint_matrix(matrix) Returns the characteristic polynomial of A and the comatrix of A-xI. Example: adjoint_matrix([[1,i],[2,3]])
affix		Syntax: affix(Point) or affix(Vector) Returns the coordinates of a point or both the x- and y-lengths of a vector as a complex number. Examples: affix(point(3,2)) returns 3+2*i If GA is a point at (1, -2), then affix(GA) returns 1-2*i.
algvar		Syntax: algvar(Expr) Returns a matrix of the symbolic variable names used in an expression. The list is ordered by the algebraic extensions required to build the original expression. Example: algvar(√x+y) → [[y],[x]]
ALOG		Common Antilogarithm Syntax: ALOG(value) Common exponential: 10^x (antilogarithm) Returns the result of raising 10 to the power of value. Examples: ALOG(2) → 100 ALOG(2+3*i) → 81.121465284+58.4748481843*i ALOG({2,4}) → {100,10000}
alog10		Syntax: alog10(Expr) Function x->10^x. Example: alog10(3) → 1000
altitude		Syntax: altitude(point1, point2, point3)

Help Topics Tree 13217	Help Text
	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. Examples: altitude(point(6,6), point(-2,3), point(5,1)) draws a line passing through point (6,6) that is perpendicular to the line passing through both points (-2,3) and (5,1).
AND	Logical AND Syntax: Value1 AND Value2 For Real numbers, returns 1 if both Value1 and Value2 are non-zero; otherwise returns 0. For Integers and Strings, AND is performed bitwise, returning 1 if corresponding bits are both 1, otherwise 0. Examples: 3 AND 2 → 1 0 AND 1 → 0 0 AND 0 → 0 {3,0,0} AND {2,1,0} → {1,0,0} 75_mph > 120_kph AND 180_deg ≠ 3.14159_rad → 1 #CC44h AND #44CCh → #4444h "a" AND "b" → "`" X:=0; 1 AND (X:=3); 0 AND (X:=5); X → 3 7 > 3 AND 5 < 9 AND 3 ≠ 2 → 1
angle	Syntax: angle(Vertex, Point2, Point3) Returns the measure of a directed angle. The first point is taken as the vertex of the angle and the next two points in order give the measure and orientation. Examples: angle(i,1,1+i,"b") returns the measure of∡BAC
angleat	Syntax: angleat(point1, point2, point3, point4) Used in Symbolic view of the Geometry app. Given the three points of an angle and a fourth point as a location, displays the measure of the angle defined by the first three points. The measure is displayed, with a label, at the location in the Plot view given by the fourth point. The first point is the vertex of the angle. Example: (in degree mode) angleat(point(0, 0), point(2*√3, 0), point(2*√3, 3), point(-6, 6)) displays "appoint(0,0)=40.9" at point (–6,6)
Ans	Last Answer Syntax: Ans In Home view, Ans returns the result of the last calculation made in Home view to its full precision. The variable Ans is different from the numbers in Home's history. A value in Ans is stored internally with the full precision of the calculated result, whereas the displayed numbers match the display mode. Ans(n) returns the nth entry in the Home view history. In CAS view, Ans returns the last result in the CAS history and Ans(n) does not recall the nth item in history. Here, Ans(n) will attempt to substitute n for x (or the default variable) in the last item in history and return the result. In CAS view, if Ans is a matrix, Ans(m,n)returns the element in row m and column n.
append	Syntax: append((List, Element) or append(Vector, Element) Append an element to a list or vector. Example: append([1,2,3],4) → [1,2,3,4]
apply	Syntax: apply(Var→f(Var), Vector) or apply(Var→f(Var), Matrix) Returns a vector or matrix containing the results of applying the function f to the elements in the vector or matrix. Examples: apply(x->x^3,[1,2,3]) → [1,8,27] apply(x->x+1,[[1,2,3],[1,2,3]],matrix)
approx	Syntax: approx(Expr, [Int]) Used in the CAS to return the numerical evaluation of the first argument with the number of digits as the second argument. Examples: approx(2/3) → 0.666666666667 approx(1/3,4) → 0.3333
ARC	Draw Arc

Help Topics Tree 13217	Help Text
	Syntax: ARC(G, x, y, r or {rx, ry}, [∡1, ∡2], [border_color, [fill_color]]) Draws a circle on GROB G, centered at (x,y), with radius r (in pixels). If r is replaced by a list {rx, ry} then the Arc becomes an ellipse centered at (x,y) with radius in the x dimension of rx and in the y dimension of ry. If ∡1 and ∡2 are specified, draws an arc from ∡1 to ∡2 using the current angle mode. Example: Demo_ARC
arc	Syntax: arc(Pnt, Pnt, Real,[Var(C)],[Var(r)]) Draws a circle arc given by 2 vertices and the angle at center. Center will be stored in C and the radius in r. Example: arc(0,1,π/4,C,r)
ARC_P	Draw Arc Syntax: ARC_P(G, x, y, r or {rx, ry}, [∡1, ∡2], [border_color, [fill_color]]) Draws a circle on GROB G, centered at (x,y), with radius r (in pixels). If r is replaced by a list {rx, ry} then the Arc becomes an ellipse centered at (x,y) with radius in the x dimension of rx and in the y dimension of ry. If ∡1 and ∡2 are specified, draws an arc from ∡1 to ∡2 using the current angle mode. Example: Demo_ARC_P
arcLen	Arc Length Syntax: arcLen(Expr, Real1, Real2) 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. Examples: arcLen(x^2,x,-2,2) → 9.29… arcLen({sin(t),cos(t)},t,0,π/2) → 1.57…
area	Syntax: area(Circle) or area(Polygon) or area(Function, Value1, Value2) Returns the area of a circle or polygon. Can also return the area under a function between two x-values. Examples: If GA is defined to be the unit circle, then area(GA) returns π. If GA is defined to be plotfunc(4-x^2/4), then area(GA,-4,4) returns 64/3 or 21.333… In CAS view, area(4-x^2/4,x=-4..4) returns 64/3 as well.
areaat	Syntax: areaat(Polygon, Point) or areaat(Circle, Point) Used in the Symbolic view of the Geometry app. Displays the algebraic area of a polygon or circle. The measure is displayed, with a label, at the given point in Plot view. Example: areaat(circle(x^2+y^2=1),point(-4,4)) displays “acircle(x^2+y^2=1)= π” at point (-4,4))
ARG	Argument Syntax: ARG(x+yi) Finds the angle determined by a complex number. Example: ARG(3+3i) → 45 (degrees mode)

ASC	Syntax:
	ASC(String)
	Returns a list containing the numerical Unicode values of String.
	Examples:
	ASC("AB") → {65,66}
	ASC("ʥ♘♕♤♦☎") → {677,9816,9813,9828,9830,9742}
	ASC({"HE","LLO"}) → {{72,69},{76,76,79}}

ASEC	Arc Secant Syntax: ASEC(value) Inverse Secant: SEC^-1 (X) Example: ASEC(1) → 0 (Degrees mode)

Help Topics Tree 13217	Help Text
	ASEC(0.498337030555+0.591083841721*i) → 1+i ASEC({2,1}) → {60,0} (Degrees mode)
ASIN	Inverse Sine Syntax: ASIN(Value) Returns the inverse sine of Value. The output depends on the Angle Measure setting in Home Settings, CAS Settings, or Symbolic Setup. Example: ASIN(1) → 90 (Degrees mode) ASIN(1.29845758142+0.634963914785*i) → 1+i ASIN({0.5,1}) → {30,90} (Degrees mode)
asin2acos	Syntax: asin2acos(Expr) Replaces arcsin(x) by π/2-arccos(x) in Expr. Example: asin2acos(acos(x)+asin(x)) → π/2-acos(x)+acos(x)

asin2atan	Syntax:
	asin2atan(Expr)
	Replaces arcsin(x) by arctan(x/√(1-x^2)) in Expr.
	Examples:
	asin2atan(2*asin(x)) → 2*atan(x/(√(1-x^2)))
	asin2atan(asin(√(1-x^2))+asin(x))

ASINH	Inverse Hyperbolic Sine Syntax: ASINH(value) Inverse Hyperbolic Sine: SINH^-1 (X) Examples: ASINH(1.17520119365) → 1 ASINH(0.634963914785+1.29845758142*i) → 1+i ASINH({0,1.17520119365}) → {0,1}
assume	Syntax: assume(Expr) Make an assumption on a variable. Example: assume(a>0) → a . Now solve(a^2=9,a) will return {3} instead of {-3,3}.
ATAN	Inverse Tangent Syntax: ATAN(Value) Returns the inverse tangent of Value. The output depends on the Angle Measure setting in Home Settings, CAS Settings, or Symbolic Setup. Example: ATAN(1) → 45 (Degrees mode) ATAN(0.27175258532+1.08392332734*i) → 1+i ATAN({1,0}) → {45,0} (Degrees mode)
atan2acos	Syntax: atan2acos(Expr) Replaces arctan(x) by π/2-arccos(x/√(1+x^2)) in the argument. Example: atan2acos(atan(2*x) → π/2-acos((2*x)/√(1+$(2*x)^2$))
ATANH	Inverse Hyperbolic Tangent Syntax: ATANH(value) Inverse Hyperbolic Tangent: TANH^-1 (X) Examples: ATANH(.761594155956) → 1 ATANH(1.08392332734+0.27175258532*i) → 1+i ATANH({0,0.46211715726}) → {0,0.5}

atrig2ln	Syntax:
	atrig2ln(Expr)
	Returns an expression with inverse trigonometric functions rewritten using the natural logarithm function.
	
	Examples:
	atrig2ln(atan(x)) → 0.5*i*ln((x+i)/(-x+i))
	atrig2ln(acos(x)) → −i*ln(x+√(x^2-1))

barycenter	barycenter Function Syntax: barycenter([Point1, Weight1], [Point2, Weight2],…,[Pointn, Weightn])

Help Topics Tree 13217	Help Text

	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.
	
	Examples:
	barycenter([0,1],[1,1],[4,2]) → point(9/4,0)
	barycenter([point(-1),1],[point(1+i),2],[point(1-i),1] → point(1/2,1/4)

basis	Syntax: basis(Lst(vector1,..,vectorn)) Extract a basis from a spanning set of vectors. Example: basis([[1,2,3],[4,5,6],[7,8,9],[10,11,12]]) → [[-3,0,3],[0,-3,-6]]

BEGIN END	BEGIN END Block
	Syntax:
	BEGIN commands; END;
	Defines a set of commands to be executed in a block.
	Example:
	EXPORT SQM1(X)
	BEGIN
	  RETURN X^2-1;
	END;
	This program defines a user function named SQM1(X). Entering SQM1(8) returns 63.

bernoulli	Bernoulli Number Syntax: bernoulli(n,[variable]) Returns Bernoulli number n, or Bernoulli polynomial n using variable. Examples: bernoulli(6) → 1/42 bernoulli(2,x) → x^2-x+1/6
Beta	Syntax: Beta(x, y) Returns the value of the Beta function for two values, x and y, defined as Gamma(x)*Gamma(y)/Gamma(x+y). Example: Beta(3,2) → 1/12
betad	Discrete Beta Syntax: betad(α,β,x) Beta probability density function Computes the probability density of the beta distribution at x given parameters α and β. Example: betad(2.2,1.5,.8) → 1.46143068876
betad_cdf	Cumulative beta Syntax: betad_cdf(a,b,x,[x2]) Returns the lower-tail probability of the beta probability density function for the value x, given parameters a and b. Examples: betad_cdf(2,1,.2) → 0.04 betad_cdf(2,1,.2,.5) → 0.21
betad_icdf	Inverse cumulative beta Syntax: betad_icdf(a,b,p) Returns the value x such that the beta lower-tail probability of x, given parameters a and b, is p. Example: betad_icdf(2,1,0.95) → 0.974679434481
BINOMIAL	Binomial Probability Density Syntax: BINOMIAL(n, p, k) Binomial probability density function. Computes the probability of k successes out of n trials, each with a probability of success of p. Note that n and k are integers with k≤n. Example: BINOMIAL(4,0.5,2) → 0.375
BINOMIAL_CDF	Cumulative Binomial Syntax: BINOMIAL_CDF(n, p, k, [k2]) Cumulative binomial distribution function

Help Topics Tree	13217	Help Text
		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≤n. With the optional fourth argument k2, returns the cumulative probability for the two k-values; that is, the probability of between k and k2 successes. Examples: BINOMIAL_CDF(20,0.5,6) → 0.05765914917 BINOMIAL_CDF(20,0.5,6,12) → 0.847717285156
BINOMIAL_ICDF		Inverse Cumulative Binomial Syntax: BINOMIAL_ICDF(n, p, q) 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. Example: BINOMIAL_ICDF(4,0.5,0.6875) → 2
bisector		Syntax: bisector(Point1, Point2, Point3) 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. Examples: bisector(0,-4*i,4) bisector(0,1,i) bisector(GA,GB,GC) draws the bisector of∡BAC. bisector(0,-4i,4) draws the line given by y=–x
BITAND		Bitwise AND Syntax: BITAND(int1, int2, … intn) Returns the bitwise logical AND of the specified integers. Example: BITAND(20,13) → 4
BITNOT		Bitwise NOT Syntax: BITNOT(int) Returns the bitwise logical NOT of the specified integer. Example: BITNOT(47) → 549755813840
BITOR		Bitwise OR Syntax: BITOR(int1, int2, … intn) Returns the bitwise logical OR of the specified integers. Example: BITOR(9,26) → 27
BITSL		Bitwise Shift Left Syntax: BITSL(int1 [, int2]) 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 of places indicated by the second integer. If there is no second integer, then the bits in the first integer are shifted to the left one place. Examples: BITSL(28,2) → 112 BITSL(5) → 10
BITSR		Bitwise Shift Right Syntax: BITSR(int1 [, int2]) 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 of places indicated by the second integer. If there is no second integer, then the bits in the first integer are shifted to the right one place. Examples: BITSR(112,2) → 28 BITSR(10) → 5
BITXOR		Bitwise XOR Syntax: BITXOR(int1, int2, … intn) Returns the bitwise logical exclusive OR of the specified integers. Example: BITXOR(9,26) → 19
BLIT		Copy GROB Syntax: BLIT([trgtG], [dx1, dy1], [dx2, dy2], srcG, [sx1, sy1], [sx2, sy2], [c], [alpha])

Help Topics Tree 13217	Help Text
	Copies the region of graphic srcG between point (sx1, sy1) and (sx2, sy2) into the region of trgtG between points (dx1, dy1) and (dx2, dy2). Pixels from srcG that are color c are not copied. alpha is a number from 0 (transparent) to 255 (opaque) which represent the transparency (alpha channel) of the source bitmap. The defaults for the optional arguments are: trgtG = G0 srcG = G0 sx1, sy1 = srcGRB top left corner sx2, sy2 = srcGRB bottom right corner dx1, dy1 = trgtGRB top left corner dx2, dy2 = calculated so destination area is the same as source area c = all pixel colors alpha= 255 (fully opaque) Note: when using the c and alpha options, it is highly recommended to specify the source x/y coordinates in order to make sure that the system can distinguish what each parameter is. Example: Demo_BLIT
BLIT_P	Copy GROB Syntax: BLIT_P([trgtG], [dx1, dy1], [dx2, dy2], srcG, [sx1, sy1], [sx2, sy2], [c], [alpha]) Copies the region of graphic srcG between point (sx1, sy1) and (sx2, sy2) into the region of trgtG between points (dx1, dy1) and (dx2, dy2). Pixels from srcG that are color c are not copied. alpha is a number from 0 (transparent) to 255 (opaque) which represent the transparency (alpha channel) of the source bitmap. The defaults for the optional arguments are: trgtG = G0 srcG = G0 sx1, sy1 = srcGRB top left corner sx2, sy2 = srcGRB bottom right corner dx1, dy1 = trgtGRB top left corner dx2, dy2 = calculated so destination area is the same as source area c = all pixel colors alpha= 255 (fully opaque) Note: when using the c and alpha options, it is highly recommended to specify the source x/y coordinates in order to make sure that the system can distinguish what each parameter is. Example: Demo_BLIT_P
blockmatrix	Block Matrix Syntax: blockmatrix(m,n,{matrix_blocks}) Returns the matrix of size m*n created from a vector of matrix_blocks of count m*n. Example: blockmatrix(2,2,{makemat(1,2,2),makemat(2,2,2),makemat(3,2,2),makemat(4,2,2)}) → [[1,1,2,2],[1,1,2,2],[3,3,4,4],[3,3,4,4]]
bounded_function	Returns the argument returned by a limit function thereby indicating that the function is bounded.
BREAK	Break Loop Syntax: BREAK [n]; Exits from expression local loop structure. Example: FOR A FROM 1 TO 10 DO B:= (A+3) MOD 5 IF B==1 THEN BREAK; END; END; If n is specified, allow to exit n loop structures. Example: Demo_BREAK
breakpoint	Syntax: breakpoint(Intg) Adds a breakpoint. Example: breakpoint(1)
B→R	Base to Real Syntax: B→R(#integer[m]) Converts an integer in base m to a decimal integer (base10). The base marker m can be b (for binary), o (for octal), or h (for hexadecimal). If m is omitted, the current system base is assumed.

Help Topics Tree 13217	Help Text
	Examples: B→R(#1101b) → 13 B→R(#1101) → 4353 (If system base is hexadecimal) B→R({#101h,#101o,#101b}) → {257,65,5}
c1oc2	Syntax: c1oc2(cycle1,cycle2) Returns the permutation product of cycle1 and cycle2. Example: c1oc2([4,1,2],[1,3]) → [3,4,2,1]
c1op2	Syntax: c1op2(cycle,permutation) Returns the permutation product of cycle and permutation. Example: c1op2([4,1,2],[3,2,4,1]) → [3,4,1,2]
canonical_form	Canonical Form Syntax: canonical_form(Trinomial,[Var]) Canonical form of a second degree polynomial. Examples: canonical_form(2*x²-12*x+1) → 2*(x-3)²-17 canonical_form(2*a²-12*a+1,a)
CAS	CAS Evaluation Syntax: CAS(expression) or CAS.function(...) or CAS.variable[(...)] Evaluate an expression or variable using the CAS. Note that outputs in numerical mode are transformed into strings or lists of expressions for symbolic matrices.
CASE	Starts a "CASE … END" branch structure. Syntax: CASE IF test1 THEN commands1 END IF test2 THEN commands2 END … IF testN THEN commandsN END [DEFAULT] [commandsD] END; Evaluates test1. If true, executes commands1 and ends the CASE. Otherwise, evaluates test2. If true, executes commands2. Continues evaluating tests until a true is found. If no true test is found, executes commandsD, if provided. Example: Demo_CASE
cat	Concatenate Syntax: cat(Obj1, Obj2, ..., Objn) Evaluates the objects in a sequence, then returns them concatenated as a string. Example: cat("aaa",c,12*3) → "aaac36"
cauchy	Cauchy Density Syntax: cauchy([x0],[a],x) Cauchy probability density function Computes the probability density of the Cauchy distribution at x given parameters x0 and a. By default, x0 is 0 and a is 1. Examples: cauchy(1) → 1/2/π cauchy(0,1,1) → 1/2/π
cauchy_cdf	Cumulative Cauchy Syntax: cauchy_cdf(x0,a,x,[x2]) Returns the lower-tail probability of the Cauchy probability density function for the value x, given parameters x0 and a. Examples: cauchy_cdf(0,2,2.1) → 0.757762116818 cauchy_cdf(0,2,2.1,3.1) → 0.0598570954516
cauchy_icdf	Inverse Cumulative Cauchy Syntax: cauchy_icdf(x0,a,p) Returns the value x such that the Cauchy lower-tail probability of x, given parameters x0 and a, is p.

Help Topics Tree		13217	Help Text
			Example: cauchy_icdf(0,2,.95) → 12.6275030293
		CEILING	Syntax: CEILING(value) Least integer greater than or equal to value. Examples: CEILING(3.2) → 4 CEILING(-3.2) → -3 CEILING({3.2,-3.2}) → {4,−3}

		center	Syntax:
			center(Circle)
			Returns the center of a circle. The circle can be defined by the circle command or by name (e.g., GC).
			
			Examples:
			center(circle(x^2+y^2-x-y)) → point(1/2,1/2)
			center(circumcircle(0,1,1+i)) → point(1/2,1/2)

		cFactor	Complex Factor Syntax: cFactor(Expr) Returns an expression factorized over the complex field (on Gaussian integers if there are more than two). Examples: cFactor(x^2*y+y) → y*(x+i)*(x-i) cFactor(x^2*y^2+y^2+2*x^2+2) → (x+i)*(x-i)*(y+√2*i)*(y+√2*(−i))
		changebase	Equivalent Matrix Syntax: changebase(A,P) changebase takes as argument a matrix A and a change-of-basis matrix P and creates an equivalent matrix, B, by multiplying them.. changebase returns the matrix B such that B=inv(P)*A*P. Example: changebase([[1,1],[0,1]],[[1,2],[3,4]]) → [[-5,-8],[9/2,7]]
		CHAR	Syntax: CHAR(List) or CHAR(Vector) or CHAR(Integer) Returns the string corresponding to the numerical Unicode character codes in List or Vector, or the numerical Unicode character code of Integer. Examples: CHAR(65) → "A" CHAR({82,77,72}) → "RMH" CHAR({#261Eh,#265Eh,#266Ch,#266Dh,#2680h,#2685h}) → "☞♞♬♭⚀⚅"
		charpoly	Characteristic polynomial Syntax: charpoly(Matrix,[Var]) 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. Examples: charpoly([[1,2],[3,4]], z) → z^2-5*z-2 charpoly([[1,2,3],[1,3,6],[2,5,7]],z)
		chinrem	Chinese Remainder Syntax: chinrem(Matrix_2xn) Given a matrix whose 2 rows each contain the coefficients of a polynomial, returns the Chinese remainder of those polynomials, also written as a matrix. Example: chinrem([[1,2,0],[1,0,1]],[[1,1,0],[1,1,1]]) → [[2,2,1] [1,1,2,1,1]]
		CHISQUARE	χ^2 Density Syntax: CHISQUARE(d, x) χ^2 (Chi-squared) probability density function Computes the probability density of the χ^2 distribution at x, given d degrees of freedom. Example: CHISQUARE(2,3.2) → 0.100948258997
		CHISQUARE_CDF	Cumulative χ^2 Syntax: CHISQUARE_CDF(d, x, [x2]) Cumulative χ^2 (Chi-squared) distribution function With two values (n and x) returns the lower-tail probability of the χ^2 probability density function for the value x, given d degrees of freedom. With the optional third argument x2, returns the area under the χ^2 probability density function between the two x-values.

Help Topics Tree 13217	Help Text
	Examples: CHISQUARE_CDF(2,6.3) → 0.957147873133 CHISQUARE_CDF(2,2,6.3) → 0.325027314304
CHISQUARE_ICDF	Inverse Cumulative χ² Syntax: CHISQUARE_ICDF(d, p) Inverse cumulative χ² (Chi-squared) distribution function Returns the value x such that the χ² lower-tail probability of x, with d degrees of freedom, is p. Example: CHISQUARE_ICDF(2,0.957147873133) → 6.3

chisquaret	χ² Test of Equality
	Syntax:
	chisquaret(Data,[distribution_law],[distribution_paramaters])
	χ² test of equality between 2 (or n) samples, or between 1 sample and distribution_law.
	
	Examples:
	chisquaret([57,54])
	chisquaret([1,1,1,1,1,0,0,1,0,0,1,1,1,0,1,1,0,1,1,0,0,0,0],[.4,.6])
	chisquaret([57,30],[.6,.4])
	chisquaret([17,15,12,15],[15,13,13,14])
	chisquaret(randvector(1000,binomial,10,.5),binomial)
	chisquaret(randvector(1000,binomial,10,.5),binomial,11,.5)
	chisquaret(randvector(1000,normald,0,.2),normald)
	chisquaret([11,16,17,22,14,10],[1/6,1/6,1/6,1/6,1/6,1/6])

cholesky	Syntax: cholesky(matrix) For a numerical symmetric matrix A, returns the matrix L such that A=L*tran(L). Example: cholesky([[3,1],[1,4]]) → [[3/√(3),0],[1/√(3),(1/3)*√(33)]]
CHOOSE	Choose Box Syntax: CHOOSE(var, “title”, “item1”, “item2”,[…"item14"]) or CHOOSE(var,"title",{"item1"..."itemN"}) Displays a choose box with the given "title" and containing items with the strings "item1", etc. If the user chooses an object, var is updated to contain the number of the selected object (an integer, 1, 2, 3, …) and CHOOSE returns true (non zero). If the user exits without choosing, var is not changed and CHOOSE returns false (0). Examples: CHOOSE(A, "Pick a Number",1,2,3,4) CHOOSE(B, "Direction", {"Up","Left","Right","Down"})

CHOOSEDATE	Date Chooser
	Syntax:
	CHOOSEDATE(var, [“title”], [min_date], [max_date])
	Displays a calendar date chooser with the optional title displayed. An optional date range may be specified between min_date and max_date. If min_date or max_date is not given, any valid date for the system is allowed to be chosen. The existing date stored in var will be selected if valid and within min_date and max_date, else the first date within the specified range, or the current system date will be selected.
	If the user chooses a date, var is updated to contain the selected date in form YYYY.MMDD and CHOOSEDATE returns true (non zero).
	If the user exits without choosing, var is not changed and CHOOSEDATE returns false (0).
	
	Examples:
	CHOOSEDATE(A)
	CHOOSEDATE(A,"My Date")
	CHOOSEDATE(A,"My Date",2017.1201)
	CHOOSEDATE(A,"My Date",2017.1201,2017.1231)
	CHOOSEDATE(A,2017.1201,2017.1231)

chrem	Chinese Remainders Syntax: chrem(List1, List2) or chrem(Vector1, Vector2) Returns a vector containing the Chinese remainders for two sets of integers, contained in either two vectors or two lists. Examples: chrem([2,3],[7,5]) → [-12,35] chrem([2*x+1,4*x+2,6*x-1,x+1],[3,5,7,11])
circle	Syntax: circle(Point1, Point2) or circle(Point1, Point2-Point1) or

Help Topics Tree 13217	Help Text
	circle(equation) Draws a circle, given the endpoints of the diameter, or a center and radius, or an equation in x and y. 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 a clockwise arc. circle(GA,GB,0,π/2) draws a quarter-circle with diameter AB.
circumcircle	Syntax: circumcircle(Point1, Point2, Point3) Draws the circumcircle of a triangle; that is, the circle circumscribed about a triangle. Example: circumcircle(GA,GB,GC) draws the circle circumscribed about ΔABC
coeff	Coefficients of Polynomial Syntax: coeff(Expr, [Var], [Integer]) Returns the list of coefficients of a polynomial with respect to the second argument or the coefficient of the term whose degree is Integer. Examples: coeff(x^3+2) → [1,0,0,2] coeff(2*y^2-3,y,0) → -3
col	Column of Matrix Syntax: col(Matrix, Integer) or col(Matrix, Interval) Returns the column n or the sequence of the columns n1 … n2 of the matrix A. Example: col([[1,2,3],[4,5,6],[7,8,9]],2) → [2,5,8]
colDim	Column Dimension Syntax: colDim(Matrix) Returns the number of columns of a matrix. Examples: colDim([[1,2,3],[4,5,6]]) → 3 colDim([[1,2],[3,4],[5,6]]) → 2
collect	Collect Like Terms Syntax: collect(Poly) or collect(Poly, Var) or collect({Poly1, Poly2,..., Polyn}) Collects like terms in a polynomial expression (or of a list of polynomial expressions). Factorizes the results, depending on the CAS settings. If specified, will collect with respect to Var. Examples: collect(x+2*x+1-4) → 3*x-3 collect(x^2-9*x+5*x+3+1) → $(x-2)^2$ collect(a*(b-c)+d*(b-c)) → (-c+b)*(a+d) collect(a*(b-c)+d*(b-c),a) → b*d-c*d+(b-c)*a
COLNORM	Column Norm Syntax: COLNORM(matrix) Finds the maximum value (over all columns) of the sums of the absolute values of all elements in a matrix. Example: COLNORM([[1,2],[3,4]]) → 6
colNorm	Column Norm Syntax: COLNORM(Matrix) Finds the maximum value (over all columns) of the sums of the absolute values of all elements in a column. Examples: COLNORM([[1,2],[3,-4]]) → 6 COLNORM([[1,2,3,-4],[-5,3,2,1]])
colspace	Column Subspace Syntax: colspace(matrix,[variable]) Returns a matrix where the columns are a basis of the vector space generated by the columns of the matrix A. If given, the dimension of this space will be stored into variable. Examples:

Help Topics Tree 13217	Help Text

	colspace([[1,2,3],[1,2,3],[1,2,4],[1,2,5]])
	colspace([[1,2,3],[1,3,6],[2,5,9]],d)

COMB	Combinations Syntax: COMB(n, r) Returns the number of combinations (without regard to order) of n things taken r at a time: n!/(r!(n-r)!) Examples: COMB(5,2) → 10 COMB({5,10,15},{1,2,3}) → {5,45,455}
COMB	Combinations Syntax: COMB(n, r) The number of combinations (without regard to order) of n things taken r at a time. Example: Suppose you want to know how many ways five things can be combined two at a time. COMB(5,2) → 10
comDenom	Common Denominator Syntax: comDenom(Expr,[Var]) 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. Example: comDenom(1/x+1/y^2+1) → (x*y^2+x+y^2)/(x*y^2)
common_perpendicular	Common Perpendicular Syntax: common_perpendicular(Line(D1),Line(D2)) Draws the common perpendicular of the lines D1 and D2. Example: common_perpendicular(line([0,0,0],[0,5,5]),line([5,0,0],[0,0,5]))
companion	Companion Matrix Syntax: companion(Poly,Var) Companion matrix of a polynomial (an=1). Example: companion(x^2+5x-7,x) → [[0,7],[1,-5]]
compare	Compare Objects Syntax: compare(Obj1, Obj2) Compares two objects and returns 1 if type(Obj1)<type(Obj2) or if type(Obj1)==type(Obj2) and Obj1<Obj2; otherwise returns 0. Examples: compare(1,2) → 1 compare("ab","cd") → 1
complexroot	Complex Root Syntax: complexroot(Poly, Real, [Complx1, Complx2]) 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. Examples: complexroot(x^3+1,0.1) → [[-1,1],[[(262144-454047*i)/524288,(524288-908093*i)/1048576],1],[[(524288+908093*i)/1048576,(262144+454047*i)/524288],1]] complexroot(x^3+1,0.1,-1,1+2*i) → [[-1,1],[[(524288+908093*i)/1048576,(262144+454047*i)/524288],1]]
CONCAT	Concatenate Syntax: CONCAT(value1, value2, [..value16]) or CONCAT(List1, List2) or CONCAT(List, Item) Concatenates (joins) items into a list or concatenates two lists. Examples: CONCAT({1,2,3},4) → {1,2,3,4} CONCAT(1,2,3,4) → {1,2,3,4} CONCAT({1,2},3,{{4,5},6,{7,8}}) → {1,2,3,{4,5},6,{7,8}}
CONCAT	Concatenate Objects

Help Topics Tree 13217	Help Text
	Syntax: CONCAT(Obj1, Obj2) Concatenates two lists or two strings or two sequences or 2 matrices. Examples: CONCAT({1,2,3},{4,5,6}) → {1,2,3,4,5,6} CONCAT([[1,2],[3,4]],[[1,2],[3,4]]) → [[1,2,1,2],[3,4,3,4]]
COND	Condition Number Syntax: COND(matrix) Finds the 1-norm (column norm) of a square matrix. Example: COND([[1,2],[3,4]]) → 21
conic	Syntax: conic(Expr) Plots the graph of a conic section defined by an expression in x and y. Example: conic(x^2+y^2-81) draws a circle with center at (0,0) and radius of 9
CONJ	Complex Conjugate Syntax: CONJ(x+yi) Reverses the sign of the imaginary part of a complex number. Examples: CONJ(3+4*i) → 3-4*i (CONJ({3+4*i,6-6*i}) → {3-4*i,6+6*i}
CONJ	Complex Conjugate Syntax: CONJ(Complex) or CONJ(List) or CONJ(Matrix) For a list or matrix, returns a list or matrix containing the complex conjugates of all complex elements. Conjugation is the negation (sign reversal) of the imaginary part of a complex number. Examples: CONJ(3+4*i) → 3-4*i CONJ([[1+i,2,3],[1,3,6],[2,5,9-i]]))
contains	Syntax: contains(List, Element) or contains(Vector, Element) 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. Example: contains({0,1,2,3},2) → 3
content	Coefficient GCD Syntax: content(Poly,[Var]) Returns the greatest common divisor (GCD) of the coefficients of a polynomial. Example: content(2*x^2+10*x+6) → 2
CONTINUE	Syntax: CONTINUE [n]; Transfers execution in a loop to the start of the next iteration of the nth upper loop (default current loop). Example: Demo_CONTINUE
CONVERT	Syntax: CONVERT(Value Unit1, 1_Unit2) Converts Value Unit1 to the corresponding value in compatible Unit2. Example: CONVERT(20_m,1_ft) → 65.6167979003_ft Alternative: 20_m ►_ft
convexhull	Convex Hull Syntax: convexhull(Point1, Point2, ..., PointN) Returns a vector containing the points that serve as the convex hull for a given set of points. Example: convexhull([0,1,1+i,1+2i,-1-i,1-3i,-2+i]) → [1-3*i 1+2*i -2+ i -1- i]
coordinates	Syntax: coordinates(Point) or

Help Topics Tree	Help Text
	coordinates(Vector) Given a point or 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. Examples: coordinates(point(1+2*i)) → [1,2] GA:=point(3,-4); coordinates(GA) → [3, -4]
CopyVar	Copy Variable Syntax: CopyVar(Var1, Var2) Copies the first variable into the second variable without evaluation. Example: CopyVar(A,B)
correlation	Syntax: correlation(List) or correlation(Matrix) Returns the correlation of the elements of a list or matrix. Example: correlation([[1,2],[1,1],[4,7]]) → 33/(6*√31)
COS	Cosine Function Syntax: COS(Value) Returns the cosine of Value. Value is interpreted as radians, degrees or gradians, depending on the setting of Angle Measure in Home Settings, CAS Settings, or Symbolic Setup. Example: COS(60) → 0.5 (Degrees mode) COS(1+i) → 0.833730025131-0.988897705763*i COS({60,0}) → {0.5,1} (Degrees mode) COS((π/3)_rad) → 0.499999999997
cos2sintan	Syntax: cos2sintan(Expr) Replaces cos(x) by sin(x)/tan(x) in the argument. Example: cos2sintan(cos(x)) → sin(x)/tan(x)
COSH	Hyperbolic Cosine Syntax: COSH(value) Hyperbolic Cosine Examples: COSH(1) → 1.54308063482 COSH(1+i) → 0.833730025131+0.988897705763*i COSH({0,1}) → {1,1.54308063482}
COT	Cotangent Syntax: COT(value) Cotangent: COS(X)/SIN(X) Example: COT(45) → 1 (Degrees mode) COT(1+i) → 0.217621561854-0.868014142896*i COT({45,90}) → {1,0} (Degrees mode) COT((π/4)_rad) → 1
count	Syntax: count(Var→Function, List) or count(Var→Function, Matrix) or count(Var→Test, List) or count(Var→Test, Matrix) There are two uses for this function, whose first argument is always a mapping of a variable onto an expression. If the expression is a function of the variable, then the function is applied to each element in a list or matrix (the second argument) and the sum of the results is returned. If the expression is a Boolean test, then each element in a list or matrix is tested and the number of elements that pass the test is returned. Examples: count(x->x²,{1,2,3}) → 14 count(x->x>6,{2,4,6,8,10,12}) → 3

count_eq	Count Items Equal
	Syntax:
	count_eq(item, list)
	count_eq(item, matrix)

Help Topics Tree 13217	Help Text
	Returns the count of items from list or matrix that are equal to item. This is equivalent to count(x→x==item,list or matrix) Examples: count_eq(2,{1,2,3,2,3,2}) → 3 count_eq(x,[x,x+1,x,x^2]) → 2
count_inf	Count Items Less Than Syntax: count_inf(item, list) count_inf(item, matrix) Returns the count of items from list or matrix that are less than item. This is equivalent to count(x→x<item,list or matrix) Examples: count_inf(2,{1,2,3,2,3,2}) → 1 count_inf(3,[1,2,3,2,3,2]) → 4
count_sup	Count Items Greater Than Syntax: count_sup(item, list) count_sup(item, matrix) Returns the count of items from list or matrix that are greater than item. This is equivalent to count(x→x>item,list or matrix) Examples: count_sup(2,{1,2,3,2,3,2}) → 2 count_sup(1,[1,2,3,2,3,2]) → 5
covariance	Covariance of Elements Syntax: covariance(List) or covariance(Matrix) Returns the covariance of the elements in a list or matrix. Example: covariance([[1,2],[1,1],[4,7]]) → 11/3
covariance_correlation	Covariance and Correlation Syntax: covariance_correlation(List) or covariance_correlation(Matrix) Returns a vector containing both the covariance and the correlation of the elements of a list or vector. Example: covariance_correlation([[1,2],[1,1],[4,7]]) → [11/3 33/(6*√31)]
cpartfrac	Complex Partial Fraction Syntax: cpartfrac(RatFrac) Returns the result of partial fraction decomposition of a rational fraction in the Complex field. Examples: cpartfrac(x/(4-x²)) → 1/((x-2)*-2)+1/((x+2)*-2) cpartfrac(a/(z*(z-b)),z)
crationalroot	Complex Rational Roots Syntax: crationalroot(Poly) Returns the list of complex rational roots of a polynomial without indicating the multiplicity. Example: crationalroot(2*x^3+(-5-7*i)*x²+(-4+14*i)*x+8-4*i) → [(3+i)/2 2*i 1+i]
CROSS	Cross Product Syntax: CROSS(Vector1, Vector2) Returns the cross product two vectors. Examples: CROSS([1,2,3],[4,3,2]) → [-5,10,-5] CROSS([1+2*i,2-4*i,3+i],[−4+i,1-3*i,2+0.5*i]) → [i,−14-5.5*i,11-19*i] CROSS({[1,2,3],[4,3,2]},{[7,2,8],[9,1,6]}) → {[10,13,−12],[16,−6,−23]}
CSC	Cosecant Syntax: CSC(value) Cosecant: 1/SIN(X) Example: CSC(90) → 1 (Degrees mode) CSC(1+i) → 0.621518017169-0.303931001627*i CSC({30,90}) → {2,1} (Degrees mode) CSC((π/6)_rad) → 2
cSolve	Complex Solve

Help Topics Tree 13217	Help Text
	Syntax: cSolve(Expr,[Var]) Returns the solutions, including complex solutions, of Expr, for Var. If Expr is an expression, solves the equation Expr=0. Examples: cSolve(x^4=1,x) → {-1,i,1,−i} cSolve(u*v-u=v and v²=u,[u,v]) → {[0,0],[(1/2*(√5+1))^2,1/2*(√5+1)],[(1/2*(-√5+1))^2,1/2*(-√5+1)]}
cumSum	Cumulative Sums Syntax: cumSum(List) or cumSum(Vector) Accepts as argument either a list or a vector and returns a list or vector whose elements are the cumulative sums of the original argument. Examples: cumSum([0,1,2,3,4]) → [0,1,3,6,10] cumSum("a","b","c","d")
curve	Syntax: curve(Expr) Reserved word. Do not use for anything.
cycle2perm	Syntax: cycle2perm(cycle) Converts cycle to a permutation. Example: cycle2perm([1,3,5]) → [3,2,5,4,1]
cycleinv	Syntax: cycleinv(cycle) Returns the inverse cycle of cycle. Example: cycleinv([1,3,2]) → [2,3,1]
cycles2permu	Syntax: cycles2permu(list) Convert a product of cycles into a permutation. Example: cycles2permu({[1,3,5],[3,4]}) → [3,2,4,5,1]
cZeros	Complex Zeros Syntax: cZeros(Expr,[Var]) or cZeros({Expr1, Expr2, ... ExprN}, {Vr1, Var2, ... VarN}) Returns the roots, including complex roots, of Expr (that is, the solution of Expr=0) or the matrix where the lines are the solutions of the system: Expr1=0, Expr2=0…ExprN=0. Examples: cZeros(x^4-1) → [1,-1, i, -i] cZeros([x²-1,x²-y²],[x,y])
C→PX	Syntax: C→PX(x, y) or C→PX({x, y}) Converts from Cartesian coordinates to screen coordinates. Examples: C→PX(0,0) → {160,110} (assuming current app Plot Settings are set to default) C→PX({15.9,10.9}) → {319,0} (assuming current app Plot Settings are set to default)
DATEADD	Date Addition Syntax: DATEADD(Date, NbDays) Adds NbDays to Date, returning the resulting date in YYYY.MMDD format. Examples: DATEADD(2008.1228, 559) → 2010.0710 DATEADD({2008.1228,2017.012},{559,1383}) → {2010.071,2020.1103}

DAYOFWEEK	Syntax:
	DAYOFWEEK(Date)
	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.
	Examples:
	DAYOFWEEK(2006.1228) → 4 (Thursday)
	DAYOFWEEK({2008.1228,2017.012,2020.1103}) → {7,5,2}

DDAYS	Date Difference Syntax: DDAYS(Date1, Date2) Calculates the numbers of days between 2 dates expressed in YYYY.MMDD format.

Help Topics Tree 13217	Help Text
	Examples: DDAYS(2008.1228,2010.0710) → 559 DDAYS({2008.1228,2017.012},{2010.071,2020.1103}) → {559,1383}
DEBUG	Debug Command Syntax: DEBUG(ProgramName(arguments)) DEBUG Inserts a breakpoint in a program, calling the Debugging Environment. When the DEBUG; line in a program is found, the Debugger opens at the following line of code. You can also use this command in Home view to debug a program. DEBUG(name) opens the Debugger with the program name.
degree	Degree of Polynomial Syntax: degree(Poly) Returns the degree of a polynomial. Examples: degree(x^3+x) → 3 degree([1,0,1,0]) → 3
DELCOL	Delete Column Syntax: DELCOL(name, column_number) Deletes column column_number from matrix name. Example: DELCOL([[1,2,3],[4,5,6]],2) → [[1,3],[4,6]]
delcols	Delete Columns Syntax: delcols(Matrix, Integer) or delcols(Matrix, Intg1..Intg2) 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. Example: delcols([[1,2,3],[4,5,6],[7,8,9]],1) → [[2,3],[5,6],[8,9]]
DELROW	Delete Row Syntax: DELROW(name, row_number) Deletes row row_number from matrix name. Example: DELROW([[1,2][3,4][5,6]],2) → [[1,3],[4,6]]
delrows	Delete Rows Syntax: delrows(Matrix, Integer) or delrows(Matrix, Intg1..Intg2) 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. Example: delrows([[1,2,3],[4,5,6],[7,8,9]],2) → [[1,2,3],[7,8,9]]
deltalist	Delta List Syntax: deltalist(List) deltalist(Vector) Creates a new list or vector composed of the first differences of a list or vector; that is, the differences between consecutive elements in the list. The new list has one less element than the original list. Example: deltalist([1,4,8,9]) → [3,4,1]
denom	Simplified Denominator Syntax: denom(a/b) For integers a and b, returns the denominator of the fraction a/b after simplification. Example: denom(10/12) → 6
desolve	Solve Differential Equation Syntax: desolve(Eq,[TimeVar],Var) Returns the solution to a differential equation. Examples: desolve(y''+y=0,y) → G_0*cos(x)+G_1*sin(x) desolve((y''+y=sin(x)) and (y(0)=1) and (y'(0)=2),y)
DET	Square Matrix Determinant

Help Topics Tree 13217	Help Text
	Syntax: DET(matrix) Determinant of a square matrix. Examples: DET([[1,2],[3,4]]) → -2 DET([[1+2*i,2+4*i],[3+i,4-5*i]]) → 12-11*i DET({[[1,2],[5,6]],[[3,4],[−6,−2]]}) → {−4,18}
dfc	Number to Continued Fraction Syntax: dfc(real,[integer]) dfc(real,[epsilon]) Returns the continued fraction representation of real with order integer using the CAS setting epsilon value, or with specified epsilon. Examples: dfc(sqrt(2)) → [1,2,2,2,2,2,2,2,2,2,2,2,2,2,2] dfc(sqrt(2),5) → [1,2,[2]] dfc(π,1e-7) → [3,7,15,1,292]
dfc2f	Continued Fraction to Number Syntax: dfc2c(continued_fraction) Transforms continued_fraction vector back into a real number, fraction, or value. Examples: dfc2f([1,1,1]) → 3/2 normal(dfc2f([1,2,[2]])) → sqrt(2)
diag	Diagonal Syntax: diag(list) or diag(matrix) 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. Examples: diag({1,2,3}) → [[1,0,0],[0,2,0],[0,0,3]] diag([[1,2],[3,4]]) → [1,4]
diff	Differentiate Syntax: diff(Expr,[Var,[Order]]) diff(Expr,[{Var1,Var2,…},[Order]]) Returns the derivative of an expression with respect to a given variable or list of variables. You can use the differentiation template in the Template menu as well. If Var or a list of variables is defined, a final parameter, Order, designates the order of the derivative to be found. Order defaults to 1. Examples: diff(x^3-x) → 3*x²-1 diff(x^3-x,x,2) → 6*x diff(sin(x)-cos(y),x) → cos(x) diff(sin(x)-cos(y),y) → sin(y) diff(sin(x)-cos(y),{x,y}) → [cos(x) sin(y)] diff(sin(x)-cos(y),{x,y},2) → [[-sin(x),0],[0,cos(y)]]
DIM	String Dimensions Syntax: DIM(String) or DIM(Matrix) Returns the number of characters in String or the dimensions of Matrix. Examples: DIM("12345") → 5 DIM([[1,2],[4,5],[7,8]]) → {3,2} DIM({"12345","HP Prime"}) → {5,8}
DIMGROB	Size GROB Syntax: DIMGROB(G, w, h, [color]) or DIMGROB(G, w, h, list) Sets the dimensions of GROB G to w*h. Initializes the graphic G with color or with the graphic data provided in list. 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 (1 bit for alpha channel and 5 bits for R, G and B). Example: Demo_DIMGROB
DIMGROB_P	Size GROB Syntax: DIMGROB_P(G, w, h, [color]) or

Help Topics Tree	13217	Help Text
		DIMGROB_P(G, w, h, list) Sets the dimensions of GROB G to w*h. Initializes the graphic G with color or with the graphic data provided in list. 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 (1 bit for alpha channel and 5 bits for R, G and B). Example: Demo_DIMGROB_P
Dirac		Dirac Delta Function Syntax: Dirac(Real) Returns the value of the Dirac delta function for a real number. Example: Dirac(-1) → 0
distance		Syntax: distance(Point1, Point2) or distance(Point, Curve) Returns the distance between two points or between a point and a curve. Example: distance(0,1+i) → √2
distance2		Distance Squared Syntax: distance2(Point1, Point2) or distance2(Point, Curve) Returns the square of the distance between two points or between a point and a curve. Examples: distance2(1+i,3+3i) → 8 If GA is the point at (0,0) and GB is defined as plotfunc(4-x^2/4), then distance (GA, GB) returns 12.
distanceat		Distance At Syntax: distanceat(Point1, Point2, Point3) or distanceat(Point1, Curve, Point3) Similar to the distance command, but used in Symbolic view of the Geometry app. Displays the distance between two points or between a point and a curve and places that measurement at the location of Point3 in the Plot view. The distance is labeled. Examples: distanceat(1+i,3+3i,point(0,0)) returns 2.828… or 2√2 and places that measure, with a label, at the origin in Plot view. If GA is the point at (0,0) and GB is defined as plotfunc(4-x^2/4), then distanceat(GA,GB,GA) returns 3.464… or 2√3 and places this measure in Plot view at (0,0). Define A:=point(0) and B:=point(1+i); then distanceat(A,B,(1+i)/2)) returns √2 and places this measurement at (1/2, 1/2) with a label.
divergence		Syntax: divergence([Expr1, Expr2, ... ExprN],[Var1, Var2, ... VarN]) Returns the divergence of a vector field, defined by divergence([A,B,C],[x,y,z])=dA/dx+dB/dy+dC/dz. Example: divergence([x^2+y,x+z+y,z^3+x^2],[x,y,z]) → 2*x+3*z^2+1
divis		Polynomial Divisors Syntax: divis(Poly) or divis({Poly1, Poly2,…Polyn}) Given a polynomial or list of polynomials, returns a vector containing the divisors of the polynomial. Example: divis(x^2-1) → [1,x-1,x+1,(x+1)*(x-1)]
division_point		Division Point Syntax: division_point(PointA, PointB, Realk) or division_point(CplxA, CplxB, Cplxk) For two points A and B, and a numerical factor k, returns a point C such that C - B = k*(C - A). The two points may be referenced by name or represented by complex numbers. Examples: division_point(0,6+6*i,4) → point (8,8) division_point(i,2+i,3)
divpc		Taylor of Quotient Syntax: divpc(Poly1, Poly2, Integer)

Help Topics Tree	13217	Help Text
		Returns the n-degree Taylor polynomial for the quotient of 2 polynomials. Example: divpc(x^4+x+2,x^2+1,5) → $x^5+3*x^4-x^3-2*x^2+x+2$, the 5th-degree polynomial
domain		Function Domain Syntax: domain(function,[variable]) Returns the domain of variable in function. If not given, variable is assumed to be x. Examples: domain((1/x)) → x≠0 domain(ln(x),x) → x>0
DOT		Dot Product Syntax: DOT(Vector1, Vector2) Returns the dot product of two vectors. Examples: DOT([1,2],[3,4]) → 11 DOT({[1,2],[5,6]},{[3,4],[−6,−2]}) → {11,−42}
DRAWMENU		Draw Button Menu Syntax: DRAWMENU(string1 or graphic, string2 or graphic,… string6 or graphic) Draws a six-button menu at the bottom of the display, with labels string1, string2, …, string6, or using the provided graphic (G0-G9 or "icon name"). Example: DRAWMENU("ABC","","DEF"); FREEZE creates a menu with the first and third buttons labeled ABC and DEF, respectively. The other four menu keys are blank.
DrawSlp		Draw Slope Syntax: DrawSlp(a, b, m) Given three real numbers a, b, and m, draws a line with slope m that passes through the point (a, b). Examples: DrawSlp(2,1,3) → line(y=3*x–5) DrawSlp(2,1,-1) → line(y=-x+3)
e		Natural Algorithm Base Syntax: e The mathematical constant e (Euler's number), internally represented as 2.71828182846 Example: e → 2.71828182846
EDITLIST		Edit List Syntax: EDITLIST(listvar or list, [title], [read only]) Allows the user to edit the specified list. If a list variable is used (e.g., L0-L9), updates the variable if OK is clicked. The title can be either "title" or { "title", {"row names"…}, {"column names"…}} "title" will be displayed above the editor as a "title" or "name". if "row names" and "column names" are specified, they will be used as row and column headers. If read only is non 0, the user will not be able to modify the object. Returns the edited list upon completion. Example: L1:={"123","456"};EDITLIST(L1) edits list L1 EDITLIST({1,2,3},"My List",1) displays a list but does not allow editing
EDITMAT		Edit Matrix Syntax: EDITMAT(matrixvar, [title], [read only]) EDITMAT(matrix, [title], [read only] Allows the user to edit or view a specified matrix. If a matrix variable is used (e.g., M0-M9), updates the variable when the user taps the OK menu key. The optional title can be either "title" or { "title", {"row names"…}, {"column names"…}} If supplied, "title" will be displayed at the top of the editor. If "row names" and "column names" are specified, they will be used as row and column headers in the editor. If read only is not 0, the user will not be able to modify the matrix, but can only view it. EDITMAT returns the edited matrix upon completion. If used in programming, returns to the program when the user taps the OK menu key. Example: EDITMAT(M1) edits matrix M1.
EDITMAT		Edit Matrix Syntax:

Help Topics Tree 13217	Help Text
	EDITMAT(matrixvar, [title], [read only]) EDITMAT(matrix, [title], [read only] Allows the user to edit or view a specified matrix. If a matrix variable is used (e.g., M0-M9), updates the variable when the user taps the OK menu key. The optional title can be either "title" or { "title", {"row names"…}, {"column names"…}} If supplied, "title" will be displayed at the top of the editor. If "row names" and "column names" are specified, they will be used as row and column headers in the editor. If read only is not 0, the user will not be able to modify the matrix, but can only view it. EDITMAT returns the edited matrix upon completion. If used in programming, returns to the program when the user taps the OK menu key. Example: EDITMAT(M1) edits matrix M1.
egcd	Syntax: egcd((PolyA, PolyB, [Var]) or egcd(ListA, ListB, [Var]) 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 that variable. Examples: egcd((x-1)^2,x^3-1) → [-x-2 1 3*x-3] egcd([1,-2,1],[1,-1,2])
Ei	Exponential Integral Syntax: Ei(x) For a real value x, returns the approximate value of int(e^(t)/t, -∞, x) Example: Ei(1.0) → 1.89511781636
EIGENVAL	Eigenvalues Syntax: EIGENVAL(matrix) Displays the eigenvalues in vector form for matrix. Example: EIGENVAL([[1,2],[3,4]]) → [5.3723, -0.3723]
eigenvals	Matrix Eigenvalues Syntax: eigenvals(Matrix) Returns the sequence of the eigenvalues of a matrix. Example: eigenvals([[-2,-2,1],[-2,1,-2],[1,-2,-2]]) → [3 -3 -3]
eigenvects	Matrix Eigenvectors Syntax: eigenvects(Matrix) Computes the eigenvectors of a diagonalizable matrix. Example: eigenvects([[-2,-2,1],[-2,1,-2],[1,-2,-2]]) → [[1 -3 -3],[-2 0 -3],[1 3 -3]]
EIGENVV	Eigenvectors and Values Syntax: EIGENVV(matrix) Eigenvectors and Eigenvalues for a square matrix Displays a list of two arrays. The first contains the eigenvectors and the second contains the eigenvalues. Example: EIGENVV([[1,2],[3,4]]) → { [[0.4160,-0.8370],[0.9094,0.5743]], [[5.3723,0], [0,-0.3723]]}
eigVc	Syntax: eigVc(Matrix) Computes the eigenvectors of a diagonalizable matrix. Example: eigVc([[-2,-2,1],[-2,1,-2],[1,-2,-2]]) → [[1 -3 -3],[-2 0 -3],[1 3 -3]]
eigVl	Syntax: eigVl(Matrix) Returns the Jordan matrix associated with a matrix when the eigenvalues are calculable. Example: eigVl([[4,1],[-4,0]]) → [[2, 1],[0,2]]
element	Point On Syntax:

Help Topics Tree	13217	Help Text
		element(Object, Real) or element(Real1..Real2) Creates a point on a geometric object whose abscissa is a given value or creates a real value on a given interval as a slider bar. The value you set using element(Real1..Real2) can be used as a coefficient in a function you subsequently define in Symbolic view and plot in Plot view. In addition, it can be used in a measurement or calculation in Numeric view. 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 on the slider name to open the slider bar and manipulate it. There is an Edit menu key that you can tap to define the slider more accurately, create animations, and so forth. Press Esc to close the slider bar at the new value or tap anywhere else on the screen.
ellipse		Syntax: ellipse(Point1, Point2, Point3) or ellipse(Point1, Point2, Realk) 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. 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.
END		Ends a structure, either a block, a test, a loop or a branch.
equilateral_triangle		Equilateral Triangle Syntax: equilateral_triangle(Point1, Point2, [Var]) 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 third point is labeled with the variable name and the coordinates of the third point are stored in that variable. The orientation of the triangle is counterclockwise from the first point. Example: equilateral_triangle(point(0,0),point(1,0)) draws the equilateral triangle through the points at (0,0), (1,0), and (1/2, √3/2).
erf		Error Function Syntax: erf(x) For a real value x, returns the approximate value of 2/√π*int(e^(-t^2),t,0,x) Example: erf(1) → 0.84270079295
erfc		Complementary Error Function Syntax: erfc(x) For a real value x, returns the approximate value of 2/√π*int(exp(-t^2),t,x,∞). Example: erfc(1) → 0.15729920705
error		Syntax: error(String) Generates the display of an error message containing String in a CAS program. Example: error("Error")
euler		Euler's Totient Syntax: euler(Integer); Euler's phi (or totient) function Takes a positive integer and returns the number of positive integers less than or equal to it that are coprime to it. Example: euler(6) → 2
EVAL		Evaluate Expression Syntax: EVAL(Expr) Useful in programs where parameters are passed unevaluated with the command QUOTE. Example: EVAL('2+3') → 5
eval		Syntax: eval(Expr) Evaluates an expression.

Help Topics Tree	13217	Help Text
		Example: eval(2*sin(π)) → 0
evalb		Evaluate Boolean Syntax: evalb(expression) This function will evaluate expression down to a true or false value if possible. Examples: evalb(5=5) → true evalb(5=-5) → false
evalc		Evaluate Complex Expression Syntax: evalb(expression) Returns a complex expression written in the form real + i*imag. Example: evalc(1/(x+y*i)) → (x/(x²+y²))-(i)*y/(x²+y²)
evalf		Evaluate Expression Numerically Syntax: evalf(Expr,[Integer]) 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. Examples: evalf(1/3,4) → 0.3333 evalf(2/3) → 0.666666666667
EVALLIST		Evaluate List Syntax: EVALLIST({list}) Evaluates the content of each element in the list and returns the resulting list. Example: EVALLIST({'1+1','4/2*(6-3)'}) → {2,6}
even		Evenness Test Syntax: even(Integer) Tests whether or not an integer is even. Returns 1 if it is and 0 if it is not. Examples: even(1251) → 0 even(8) → 1
exact		Exact Conversion Syntax: exact(Expr) Converts a decimal expression to a rational or real expression within the Epsilon tolerance specified in CAS settings. Examples: exact(1.4141) → 14141/10000 exact(0.156381102937) → 2572/16447
exbisector		exbisector Function Syntax: exbisector(Point1, Point2, Point3) 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. Examples: exbisector(GA,GB,GC) 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
excircle		Syntax: excircle(Point1, Point2, Point3) Given three points that define a triangle, draws the excircles 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 whose 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.
EXECON		Execute On Element 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}) → {2,3,4} In the example above, &1 indicates an element in the list. &1+1 means to add 1 to each element of the list.

Help Topics Tree	13217	Help Text
		Where the & is followed directly by a number, the relative position in the list is indicated. For example: EXECON("&2-&1",{1,4,3,5}) → {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}) → {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. These element pairs are added 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 refers to the element in the list; 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}) → {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. EXECON can also operate on matrices in the same way as lists: EXECON("&1+&2",[[1,2],[3,4]],[[5,6],[6,7]]) → [[6,8],[9,11]] In the example above, the result is the sum of the two matrices.
EXP		Natural Exponential Syntax: EXP(value) Natural exponential: e^x (natural antilogarithm) Returns the result of raising e (Euler's number) to the power value. Examples: EXP(5) → 148.413159103 EXP(2+3*i) → −7.3151100949+1.04274365624*i EXP({-2.3,0}) → {0.100258843723,1}
exp2pow		Syntax: exp2pow(Expr) Transforms an expression of the form e^(n*ln(x)) rewritten as a power of x. Applies e^(n*ln(x))=x^n. Example: exp2pow(e^(3*ln(x))) → x^3
exp2trig		Syntax: exp2trig(Expr) Returns an expression with complex exponentials rewritten in terms of sine and cosine. Example: exp2trig(exp(-i*x)) → cos(x)+ i*sin(x)
expand		Expand Expression Syntax: expand(Expr) Returns an expression expanded. Example: expand((x+y)*(z+1)) → y*z+x*z+y+x
expexpand		Expand Exponentials Syntax: expexpand(Expr) Expands exponentials using the identity e^(a*f(x))=e^(f(x))^a. Example: expexpand(e^(3*x)) → $(e^x)^3$
EXPM1		Exponent Minus 1 Syntax: EXPM1(value) Exponential minus 1: (e^x)-1 This is more accurate than EXP when x is close to zero. Examples: EXPM1(0.23) → 0.258600009929 EXPM1(0.02+0.03*i) → 1.97422838545ᴇ−2+3.06014495014ᴇ−2*i
exponential		Discrete Exponential Syntax: exponential(k,x) Exponential probability density function Computes the probability density of the exponential distribution at x given parameter k. Example:

Help Topics Tree 13217	Help Text
	exponential(2.1,0.5) → 0.734869273133
exponential_cdf	Cumulative Exponential Syntax: exponential_cdf(k,x,[x2]) Cumulative exponential distribution function Returns the lower-tail probability of the exponential probability density function for the value x, given parameter k. Examples: exponential_cdf(4.2,.5) → 0.877543571747 exponential_cdf(4.2,.5,3) → 0.122453056238
exponential_icdf	Inverse Cumulative Exponential Syntax: exponential_icdf(k,p) Inverse cumulative exponential distribution function Returns the value x such that the exponential lower-tail probability of x, given parameter k, is p. Example: exponential_icdf(4.2,0.95) → 0.713269588941
exponential_regression	Exponential Regression Syntax: exponential_regression(Matrix) or exponential_regression(List1, List2) or exponential_regression(Vector1, Vector2) 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 2 x n matrix. Example: exponential_regression([1.0,0.0,4.0],[2.0,1.0,7.0]) → [1.60092225473 1.10008339351]
EXPORT	EXPORT function or variables Syntax: EXPORT FunctionName(Parameters) EXPORT Var1[,Var2, ... ,Var8]; EXPORT Var1[:=Val1, Var2:=Val2, ... Var8:=Val8]; In a program, declares functions or variables to export globally. Exported functions appear in the Toolbox User menu; exported variables appear in the Vars CAS, App, or User menus. For an exported function: Forward function declaration: EXPORT function(params); Normal function declaration: EXPORT function[(params)] BEGIN //Function definition goes here END; Examples: EXPORT X2m1(X); EXPORT ratio:=0.15; EXPORT X2M1(X) BEGIN RETURN X^2-1; END; Examples: Demo_EXPORT
EXPR	Evaluate String Syntax: EXPR(String) Parses a string into a number or expression and returns the result evaluated. Examples: EXPR("2+3") → 5 X:=90; EXPR("X+10") → 100 X:=90; Y:=3; EXPR({"X/2","2^Y"}) → {45,8}
extract_measure	Extract Measure Syntax: extract_measure(Var) 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. Examples: extract_measure(angleatraw(0,1,1+i,1) → π/4 extract_measure(distanceatraw(0,1+i,(1+i)/2)) → √2
ezgcd	Syntax:

Help Topics Tree 13217	Help Text
	ezgcd(Poly1, Poly2) Uses the EZ GCD algorithm to return the greatest common divisor of two polynomials with at least two variables. Example: ezgcd(x^2-2*x-x*y+2*y,x^2-y^2) → x-y
F-J	Function Catalog F-J Toolbox function catalog F-J
f2nd	Fraction → Numerator/Denominator Syntax: f2nd(Frac) or f2nd(RatFrac) Returns a vector consisting of the numerator and denominator of an irreducible form of a rational fraction. Examples: f2nd(42/12) → [7,2] f2nd((x^2+2*x+1)/(x^2-1)) → [x^2+2.*x+1.,x^2-1.]
factor	Factorize Polynomial Syntax: factor(Expr) Returns a polynomial factorized. Similar to collect, but will factor using square roots. Examples: factor(x^4+12*x^3+54*x^2+108*x+81) → (x+3)^4 factor(x^4-1) → (x-1)*(x+1)*(x^2+1)
factor_xn	Factor by Degree Syntax: factor_xn(Poly) For a given polynomial in x of degree n, factors out x^n and returns the resulting product. Examples: factor_xn(x^4-1) → x^4*(1-x^-4) factor_xn(x^4+12*x^3+54*x^2+108*x+81)
factorial	Syntax: factorial(Integer) or factorial(Real) 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). Examples: factorial(4) → 24 factorial(1.2) → 1.10180249088
factors	Polynomial Factor List Syntax: factors(Poly) or factors({Poly1, Poly2, ..., Polyn}) Returns the list of prime factors of a polynomial; each factor followed by its multiplicity. Examples: factors(x^4-1) → [x-1,1,x+1,1,x^2+1,1] factors([x^2,x^2-1])
fcoeff	Roots to Polynomial Syntax: fcoeff([Root1, Order1, Root2, Order2, ..., Rootn, Ordern]) Returns the polynomial described by a list of roots, each followed by its order. Example: fcoeff([1,2,0,1,3,-1]) → x*$(x-1)^2$/(x-3)
fft	Fast Fourier Transform Syntax: fft(Vector) or fft(Vector, a, p) With one argument (a vector), returns the discrete Fourier transform in R. With two additional integer arguments a and p, returns the discrete Fourier transform in the field Z/pZ, with a primitive nth root of 1 (n=size(Vector)). Example: fft([1,2,3,4,0,0,0,0]) → [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]
FILLPOLY	Draw Filled Polygon Syntax: FILLPOLY([G], {Coordinates}, Color, [Alpha]) FILLPOLY([G], [Coordinates], Color, [Alpha]) Fills the polygon specified by the provided Cartesian coordinates using the color provided. If Alpha (0 to 255) is provided, the polygon is drawn with transparency. Examples:

Help Topics Tree	13217	Help Text
		FILLPOLY([(0,0),(1,1),(2,0),(3,-1),(2,-2)],#FF,128) Demo_FILLPOLY
FILLPOLY_P		Draw Filled Polygon Syntax: FILLPOLY_P([G], {Coordinates}, Color, [Alpha]) FILLPOLY_P([G], [Coordinates], Color, [Alpha]) Fills the polygon specified by the provided pixel coordinates using the color provided. If Alpha (0 to 255) is provided, the polygon is drawn with transparency. Examples: FILLPOLY_P([(20,20),(120,120),(150,20),(180,150),(50,100)],#FF,128) Demo_FILLPOLY_P
FISHER		Fisher Density Syntax: FISHER(n, d, x) 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. Example: FISHER(5,5,2) → 0.158080231095
FISHER_CDF		Cumulative Fisher Syntax: FISHER_CDF(n, d, x, [x2]) Cumulative F (Fisher or Fisher-Snedecor) distribution function Returns the lower-tail probability of the F probability density function for the value x, given numerator n and denominator d degrees of freedom. With the optional fourth argument x2, returns the area under the F probability density function between the two x-values. Examples: FISHER_CDF(5,5,2) → 0.76748868087 FISHER_CDF(5,5,0.5,2) → 0.53497736174
FISHER_ICDF		Inverse Cumulative Fisher Syntax: FISHER_ICDF(n, d, p) Inverse cumulative F (Fisher or Fisher-Snedecor) distribution function. Returns the value x such that the F lower-tail probability of x, with numerator n, and denominator d degrees of freedom, is p. Example: FISHER_ICDF(5,5,0.76748868087) → 2
FLOOR		Syntax: FLOOR(value) Greatest integer less than or equal to value. Examples: FLOOR(3.2) → 3 FLOOR(-3.2) → -4 FLOOR({3.2,-3.2}) → {3,−4}
fMax		Function Maximum Syntax: fMax(Expr,[Var]) 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. Example: fMax(-x²+2*x+1,x) → 1
fMin		Function Minimum Syntax: fMin(Expr,[Var]) 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. Example: fMin(x²-2*x+1,x) → 1
FNROOT		Find Root Syntax: FNROOT(Expr, Var, [guess], [guess2]) Function root-finder (like the Solve app). Finds the value for variable at which an expression most nearly evaluates to zero. Uses guess as initial estimate. Examples: FNROOT(A*9.8/600-1,A,1) → 61.2244897959 FNROOT(X²-3,X,-2) → −1.73205080757 FNROOT(X²-3,X,2) → 1.73205080757

Help Topics Tree	13217	Help Text
		FNROOT(X^2-3,X,2,-2) → −1.73205080757 FNROOT({'X^2-3','T^3+4'},{'X','T'},{-2,-1},{2,1}) → {−1.73205080757,−1.58740105197}

FOR FROM TO DO END		For Loop Structure
		Syntax:
		FOR var FROM start TO (or DOWNTO) finish [STEP increment] DO commands END;
		Sets variable var to start; then, for as long as this variable's value is less than or equal to (or more than for a DOWNTO) finish, executes commands and adds (or subtracts for DOWNTO) 1 (or increment) to var.
		
		Examples:
		//print 1 3 5 7 9
		FOR A FROM 1 TO 10 STEP 2
		  DO
		  PRINT(A);
		END;
		//print 10 8 6 4 2
		FOR A FROM 10 DOWNTO 1 STEP 2
		  DO
		  PRINT(A);
		END;
		Example:
		Demo_FOR

format		Format Number Syntax: format(real, format)) Returns real number as a string in the indicated format (f=float, s=scientific, e=engineering, a=hexadecimal). Examples: format(9.3456,"s3") → "9.35" format(pi,"a12") → "0x1.921fb54442d0p+1"
fourier_an		Syntax: fourier_an(Expr,Var,T,n,a) Returns the nth Fourier coefficient an=2/T*∫(f(x)*cos(2*pi*n*x/T),a,a+T). Example: fourier_an(x^2,x,2,0,-1) → 1/3
fourier_bn		Syntax: fourier_bn(Expr,Var,T,n,a) Returns the nth Fourier coefficient bn=2/T*∫(f(x)*sin(2*pi*n*x/T),a,a+T). Example: fourier_bn(x^2,x,2,0,-1) → 0
fourier_cn		Syntax: fourier_cn(Expr,Var,T,n,a) Returns the nth Fourier coefficient cn=1/T*∫(f(x)*exp(-2*i*pi*n*x/T),a,a+T). Example: fourier_cn(x^2,x,2,0,-1) → 1/3
FP		Fractional Part Syntax: FP(value) Returns the Fractional part of value. Examples: FP(23.2) → 0.2 FP(-23.2) → -0.2 FP({23.2,15+1/4,51/2,10-4/5}) → {0.2,0.25,0.5,0.2}
fPart		Fractional Part Syntax: fPart(Real) or fPart(List) Returns the fractional part of a real number or the fractional parts of a list of real numbers. Examples: fPart(1.2) → 0.2 fPart([3.4,√2])
fracmod		Syntax: fracmod(Expr,Integer)) For a given expression and an integer n, returns the fraction a/b such that a/b=Expr mod n, where -√n/2<a≤√n/2 and 0≤b<√n/2 Example: fracmod(41,121) → 2/3
FREEZE		Freeze Screen Syntax: FREEZE

Help Topics Tree 13217	Help Text
	Prevents the screen from being redrawn after the program ends. Leaves the modified display on the screen for the user to see. This command does not pause and wait for input. Rather, it prevents a redraw until any other operation (key press, screen touch, or data communication, or command) triggers the screen to be drawn. Example: FREEZE
froot	Numerical Roots & Poles Syntax: froot(RatPoly) Returns the list of roots and poles of a rational polynomial with their multiplicities. Example: froot((x^5-2*x^4+x^3)/(x-1)) → [0,3,1,2,3,-1]
fsolve	Numerical Solve Syntax: fsolve(Expr,Var,[Guess or Interval],[Method]) fsolve(ExprVector, [Guess or Interval], [Method}) 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. If you are solving for a single variable, your options are bisection_solver, newton_solver, or newtonj_solver. If solving for 2 variables, your only option is newton_solver. Examples: fsolve(cos(x)=x,x,-1..1) → [0.739085133215] fsolve([x²+y-2,x+y²-2],[x,y],[0,0]) → [1.,1.]
function_diff	Functional Derivative Syntax: function_diff(FunctionName) Returns the derivative function as a mapping of x onto the derivative of the given function. Examples: function_diff(sin) → (x)->cos(x) function_diff(sin²+id) → (x)->2*cos(x)*sin(x)+1
Gamma	Gamma Function Syntax: Gamma(Real) Returns the value of the gamma function (Γ) for a real number. Gamma(n)=(n-1)! if n is an integer. Examples: Gamma(5) → 24 Gamma(1/2)
gammad	Gamma Density Syntax: gammad(a,t,x) Gamma probability density function Computes the probability density of the gamma distribution at x given parameters a and t. Example: gammad(2.2,1.5,.8) → 0.510330619114
gammad_cdf	Cumulative Gamma Syntax: gammad_cdf(a,t,x,[x2]) 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 x2, returns the area between the two x-values. Examples: gammad_cdf(2,1,2.96) → 0.794797087996 gammad_cdf(2,1,2.96,4) → 0.11362471756
gammad_icdf	Inverse Cumulative Gamma Syntax: gammad_icdf(a,t,p) 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. Example: gammad_icdf(2,1,0.95) → 4.74386451839
gauss	Syntax: gauss(Expr,Vector)

Help Topics Tree 13217	Help Text
	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. Example: gauss(x^2+2*a*x*y,[x,y]) → -a^2*y^2+(a*y+x)^2
gbasis	Groebner Basis Syntax: gbasis([Poly1, Poly2,...], [Var1, Var2, ...]) Given a vector of polynomials and a vector of variables, returns the Groebner basis of the ideal spanned by the set of polynomials. Example: gbasis([x^2-y^3,x+y^2],[x,y]) → [x*y+x^2,y^2+x]
gcd	Greatest Common Divisor Syntax: gcd(Poly1, Poly2) or gcd(Integer1, Integer2) Returns the greatest common divisor of 2 polynomials of several variables. Can also be used as integer gcd. Examples: gcd(x^2-4,x^2-5*x+6) → x-2 gcd(45,30) → 15
GETBASE	Get Base Syntax: GETBASE(#integer[m]) Returns the base number for integer with base marker m. The base number is used by the SETBASE function. 0 = System 1 = Binary 2 = Octal 3 = Decimal 4 = Hexadecimal The base marker m can be b (for binary), o (for octal), d (for decimal), or h (for hexadecimal). If m is omitted, the current system base is assumed. Examples: GETBASE(#1101b) → #1h GETBASE(#1101) → #0h (if default base is hexadecimal) GETBASE({#100h,#100d,#100o,#100b}) → {#4h,#3h,#2h,#1h}
GETBITS	Get Bits Syntax: GETBITS(#integer) Returns the number of bits used for encoding an integer. If not specified, then the value in the Integers field of Page 1 of Home Settings is used. Examples: GETBITS(#22122) → 32 (If Home Settings Integers is set to 32 bits) GETBITS(#1:45h) → 45 GETBITS(#153:-16o) → -16 GETBITS({#FFFF:16h,#777:-23o}) → {16,−23}
GETKEY	Get Key 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). 0 = Apps 1 = Symb 2 = Up 3 = Help 4 = Esc 5 = Home 6 = Plot 7 = Left 8 = Right 9 = View 10 = CAS 11 = Num 12 = Down 13 = Menu After that, the keys are numbered from top left (14 = Vars) to bottom right (50 = +)
GETPIX	Get Pixel Color Syntax: GETPIX([G], x, y) Returns the color of the pixel of G with Cartesian coordinates (x, y).

Help Topics Tree 13217	Help Text
	Examples: Demo_GETPIX
GETPIX_P	Get Pixel Color Syntax: GETPIX_P([G], x, y) Returns the color of the pixel of G with pixel coordinates (x, y). Examples: Demo_GETPIX_P
GF	Create Galois Field Syntax: GF(Integerp, Integern) Creates a Galois Field of characteristic p with p^n elements. Example: GF(5,9) → GF(5,k^9-k^8+2*k^7+2*k^5-k^2+2*k-2,[k,K,g],undef)
grad	Gradient Syntax: grad(Expr, ListVars) Returns the gradient of an expression. With a list of variables as second argument, returns the vector of partial derivatives. Example: grad(2*x²*y-x*z^3,[x,y,z]) → [-z³+4*x*y 2*x² -3*x*z²]
gramschmidt	Gramschmidt Orthonormalization Syntax: gramschmidt(Vector, Function) 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. Example: gramschmidt([1,1+x],(p,q)->integrate(p*q,x,-1,1)) → [1/(√2),(1+x-1)/(√6)/3]
greduce	Groebner Remainder Syntax: greduce(Poly1, [Poly2, Poly3,...], [Var1, Var2, ...]) 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. Examples: greduce(x*y-1,{x²-y²,2*x*y-y²,y^3},{x,y}) → (1/2)*y²-1 greduce(x1²*x3²,[x3^3-1,-x2²-x2*x3-x3²,x1+x2+x3],[x1,x2,x3]) → x2
GROBH	GROB Height Syntax: GROBH(G) Returns the height of the graphic object G. Example: GROBH(G0) → 24
GROBH_P	GROB Height Syntax: GROBH_P(G) Returns the height of the graphic object G in pixels. Example: GROBH(G0) → 240
GROBW	GROB Width Syntax: GROBW(G) Returns the width of the graphic object G. Example: GROBW(G0) → 32
GROBW_P	GROB Width Syntax: GROBW_P(G) Returns the width of the graphic object G in pixels. Example: GROBW_P(G0) → 320
groupermu	Syntax: groupermu(permutation1,permutation2) Returns the group of permutations generated by permutation1 and permutation2. Example: groupermu([2,1],[2,3,1])
hadamard	Syntax: hadamard(Matrix,[Matrix]) Hadamard bound of a matrix or element by element multiplication of 2 matrices. Examples:

Help Topics Tree 13217	Help Text
	hadamard([[1,2],[3,4]]) → 5*√5 hadamard([[1,2],[3,4]],[[3,4],[5,6]]) → [[3,8],[15,24]]
half_line	Ray Syntax: half_line(Point1, Point2) Given 2 points, draws a ray from the first point through the second point. Example: half_line(0,1+i) draws a ray starting at the origin and passing through the point at (1,1)

halftan	Syntax:
	halftan(Expr)
	Transforms sin(x), cos(x) and tan(x) as a function of tan(x/2).
	Examples:
	halftan(sin(x)) → (2*TAN(x/2))/((TAN(x/2))²+1)
	halftan(tan(x)) → (2*TAN(x/2))/(-(TAN(x/2))²+1)

halftan_hyp2exp	Syntax: halftan_hyp2exp(Expr) Transforms the trigonometric functions in tan(x/2) and hyperbolic functions into exponentials. Example: halftan_hyp2exp(sin(x)+sinh(x)) → (1/2)*((-1/e^x)+e^x)+2*tan((1/2)*x)/((tan((1/2)*x))²+1)
halt	Syntax: halt Puts a program in step-by-step debug mode. Example: halt()
hamdist	Hamming Distance Syntax: hamdist(Intgr1, Intgr2) Returns the Hamming distance between two integers. Example: hamdist(0x12,0x38) → 3
harmonic_conjugate	Harmonic Conjugate Syntax: harmonic_conjugate(Point1, Point2, Point3) or harmonic_conjugate(Line1, Line2, Line3) 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. Examples: harmonic_conjugate(point(0,0),point(3,0),point(4,0)) harmonic_conjugate(line(0,1+i),line(0,3+i),point(3/2+i)) harmonic_conjugate(point(0, 0), point(3, 0), point(4, 0)) → point(12/5, 0)
harmonic_division	Harmonic Division Syntax: harmonic_division(point1, point2, point3, var) or harmonic_division(line1, line2, line3, var) Returns the 4 points (resp lines) and affects the last argument, such as the 4 points (resp lines) are in a harmonic division. Examples: harmonic_division(point(0, 0),point(3, 0),point(4, 0), p) → point(12/5,0) harmonic_division(line(0,1+i),line(0,3+i),line(0,i),D)
has	Has Variable Syntax: has(Expr,Var) Checks if a variable is in an expression. Returns 1 if the variable is in the expression, and returns 0 otherwise. Examples: has(x+y,x) → 1 has(x+y,n) → 0
head	Head Element Syntax: head(Vector) or head(String) or head(Obj1, Obj2,...) Returns the first element of a vector or a string or a set of objects. Examples: head(1,2,3) → 1 head("bonjour") → "b"
Heaviside	Syntax:

Help Topics Tree	13217	Help Text
		Heaviside(Real) Returns the value of the Heaviside function for a given real number. The Heaviside function is equal to 0 if x<0 and 1 if x≥0. Example: Heaviside(1) → 1
hermite		Hermite Polynomial Syntax: hermite(Integer) Returns the Hermite polynomial of degree n, where n is an integer less than 1556. Example: hermite(3) → 8*x³-12*x
hessenberg		Syntax: hessenberg(Matrix_A) Given Matrix_A, returns the matrix reduction to Hessenberg form. Returns [P,B] such that B=inv(P)*A*P. Example: hessenberg([[1,2,3],[4,5,6],[7,8,1]]) → [[[1,0,0],[0,4/7,1],[0,1,0]],[[1,29/7,2],[7,39/7,8],[0,278/49,3/7]]]
hessian		Hessian Matrix Syntax: hessian(Expr,ListVar) Returns the Hessian matrix of an expression. Example: hessian(2*x²*y-x*z,[x,y,z]) → [[4*y,4*x,-1],[4*x,0,0],[-1,0,0]]
hexagon		Syntax: hexagon(Point1, Point2, [Var1, Var2, Var3, Var4]) Draws a regular hexagon defined by one of its sides; that is, by two consecutive vertices. The remaining points are calculated automatically, but are not defined symbolically. The orientation of the hexagon is counterclockwise from the first point. Examples: hexagon(0,6) draws a regular hexagon whose first two vertices are at (0, 0) and (6, 0). hexagon(0,6,a,b,c,d) draws a regular hexagon whose first two vertices are at (0, 0) and (6, 0)l labels the other four vertices a, b, c, and d, and stores the coordinates into the CAS variables a, b, c, and d. You do not have to define variables for all four remaining points, but the coordinates are stored in order. For example, hexagon(0,6, a) stores just the third point into the CAS variable a.
hilbert		Hilbert Matrix Syntax: hilbert(n) 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. Example: hilbert(3) → [[1,1/2,1/3],[1/2,1/3,1/4],[1/3,1/4,1/5]]
HMS→		Syntax: HMS→(value) Displays a sexagesimal value in decimal format. Examples: HMS→(8°30) → 8.5 HMS→({8°30’00”,286°15’00”}) → {8.5,286.25}
homothety		Dilation Syntax: homothety(Point, Realk, Object) Dilates a geometric object, with respect to a center point, by a scale factor. Examples: 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. homothety(point(0,0),1/3,point(9,9)) creates an image point at (3,3).
horner		Syntax: horner(Polynomial,Real) 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. Examples: horner(x^2+1,2) → 5 horner([1,0,1],2) → 5
hyp2exp		Syntax: hyp2exp(Expr) Returns an expression with hyperbolic terms rewritten as exponentials. Example: hyp2exp(cosh(x)) → (exp(x)+1/exp(x))/2
hyperbola		Syntax: hyperbola(Point1, Point2, Point3) or

Help Topics Tree 13217	Help Text
	hyperbola(Point1, Point2, Realk) 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. 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.
iabcuv	Syntax: iabcuv(Intg(a),Intg(b),Intg(c)) Returns [u,v] such as au+bv=c for 3 integers a,b,c Example: iabcuv(21,28,7) → [-1,1]
ibasis	Intersection Basis Syntax: ibasis(Matrix1, Matrix2)) Given two matrices, interprets them as two vector spaces and returns the vector basis of their intersection. Example: ibasis([[1,0,0],[0,1,0]],[[1,1,1],[0,0,1]]) → [-1 -1 0]
ibpdv	Integration By Parts v Syntax: ibpdv(f(Var), v(Var), [Var], [Real1], [Real2]) 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. Examples: ibpdv(ln(x),1) → x*ln(x)-x ibpdv(ln(x),x) → [x*ln(x), -1]
ibpu	Integration By Parts u Syntax: ibpu(f(Var), u(Var), [Var], [Real1], [Real2]) 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, it 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. Example: ibpu(x*ln(x), x) → [x*(x*ln(x)-x), -x*ln(x)+x]
ichinrem	Integer Chinese Remainder Syntax: ichinrem([a,p],[b,q])) Integer Chinese Remainder Theorem for two equations. Takes two lists [a, p] and [b, q] and returns a list 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. Example: ichinrem([2,7],[3,5]) → [23,35]
icontent	GCD of Integer Coefficients Syntax: icontent(Poly,[Var]) Returns the greatest common divisor of the integer coefficients of a polynomial. Example: icontent(24x^3+6x²-12x+18) → 6
id	Identity Function Syntax: id(Expr) The id entity function: x→x. Returns a set containing the original argument. Example: id(1,2,3) → [1 2 3]
IDENMAT	Identity Matrix Syntax: IDENMAT(n) Creates a square matrix of dimension n x n whose diagonal elements are 1 and off-diagonal elements are zero. Examples: IDENMAT(2) → [[1,0],[0,1]] IDENMAT({2,3}) → {[[1,0],[0,1]],[[1,0,0],[0,1,0],[0,0,1]]}
identity	Identity Matrix Syntax:

Help Topics Tree 13217	Help Text
	identity(Integer) Given an integer n, returns the identity matrix of dimension n. Example: identity(3) → [[1,0,0],[0,1,0],[0,0,1]]
idivis	Integer Divisors Syntax: idivis(Integer) or idivis({Intgr1, Intgr2, ... Intgrn}) Returns a list of all the factors of an integer or of a list of integers. Example: idivis(12) → [1, 2, 3, 4, 6, 12]
iegcd	Integer Extended GCD Syntax: iegcd(Integer1, Integer2) Given two integers a and b, returns the extended greatest common divisor for two integers. Returns [u,v,igcd(a,b)] such that a*u+b*v=igcd(a,b). Example: iegcd(14, 21) → [-1, 1, 7]

IF THEN ELSE END	IF Branch Structure
	Syntax:
	IF test THEN commands1 [ELSE commands2] END;
	Starts an "IF … THEN … END" or "IF … THEN … ELSE … END" branch structure.
	Evaluate test. If test is true (non 0), executes commands1, otherwise, executes commands2
	
	Example:
	IF A<1
	  THEN PRINT("A<1");
	  ELSE PRINT("A>1");
	END;
	Examples:
	Demo_IF

ifactor	Integer Factors Syntax: ifactor(Integer) Returns the prime factorization of an integer as a product. Can be used with STO▸. Note: in some cases, factorization may fail. In these cases, the command will return the product of -1 and the opposite of the original input. The -1 indicates that factorization failed. Example: ifactor(150) → $2*3*5^2$
ifactors	Integer Factors List Syntax: ifactors(Integer) Similar to ifactor, but returns a list of the factors of the integer with their multiplicities. Example: ifactors(150) → [2, 1, 3, 1, 5, 2]
IFERR	Error Trapping Structure Syntax: 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. Many conditions are automatically recognized by the HP Prime as error conditions and are automatically treated as errors in programs. This command facilitates error-trapping of such errors. Note: the error number will be stored in the Ans variable. So you can access it and use it in the THEN clause of the IFERR. Example: IFERR 1/0 THEN PRINT("1/0 Error"); END; Example: Demo_IFERR
ifft	Inverse Fast Fourier Transform Syntax: ifft(Vect) Returns the inverse discrete Fourier transform. 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]) → [0.99999999999,3.99999999999,10.0,20.0,25.0,24.0,16.0,-6.39843733552e-12]

Help Topics Tree	13217	Help Text
IFTE		If Then Else structure
		Syntax:
		IFTE(Expr, TrueClause, FalseClause)
		If Expr evaluates true (1), evaluates TrueClause; if not, evaluates FalseClause.
		If Expr returns a list, then TrueClause and FalseClause each have to be either a single object, or a list of the same size as the result of Expr. The result will be a list of that size with elements picked from TrueClause and FalseClause according to the Boolean value of each element of the result of Expr.
		Examples:
		IFTE(2<3, 5-1, 2+7) → 4
		IFTE({2<3,√(6*π)≤3},5-1,{2+7,7*6}) → {4,7*6}
igcd		Integer GCD
		Syntax:
		igcd(Intgr1, Intgr2, ... Intgrn))
		Returns the integer that is the greatest common divisor of two or more integers.
		Example:
		igcd(24,36) → 12
ihermite		Hermite Normal
		Syntax:
		ihermite(Matrix_A)
		Given Matrix_A, returns the 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
		Example:
		ihermite([[1,2,3],[4,5,6],[7,8,9]]) → [[-3,1,0],[4,-1,0],[-1,2,-1]],[[1,-1,-3],[0,3,6],[0,0,0]]
ilaplace		Inverse Laplace Transform
		Syntax:
		ilaplace(Expr,[Var],[IlapVar])
		Returns the inverse Laplace transform of a rational fraction.
		Examples:
		ilaplace(1/(x^2+1)^2) → (1/2)*sin(x)-(1/2)*x*cos(x)
		ilaplace(s/(s^4-1),s,x)
IM		Imaginary Part
		Syntax:
		IM(x+yi)
		Returns the imaginary part of a complex number.
		Examples:
		IM(3+4i) → 4
		IM({3+4*i,6-6*i}) → {4,-6}
image		Syntax:
		image(Matrix)
		Image of a linear application of a matrix.
		Example:
		image([[1,2],[3,6]]) → [1,3]
implicit_diff		Implicit derivative
		Syntax:
		implicit_diff(expression, var1, var2, [Order])
		Returns the implicit derivative of expression with respect to var1, var2. The result is an expression that defines d(var2)/d(var1), so the order of the variables is important. expression is usually an equation; if there is no equal sign, expression=0 is assumed. The optional parameter, Order, designates the order of the derivative to be found. Order defaults to 1.
		Examples:
		implicit_diff(y^5=x,x,y) → 1/(5*y^4)
		implicit_diff(y^5=x,y,x) → 5*y^4
		implicit_diff(x^2+y^2-5,x,y) → –x/y
		implicit_diff(x^2+y^2-5,x,y,2) → (-x^2-y^2)/y^3
incircle		Syntax:
		incircle(Point1, Point2, Point3)
		Draws the incircle of a triangle, the circle tangent to all three sides of the triangle.
		Examples:
		incircle(0,4,4+4*i)
		incircle(GA,GB,GC) draws the incircle of ΔABC.
INPUT		Input Form
		Syntax:
		INPUT(var,["title"], ["label"], ["help"], [reset_value], [initial_value])
		INPUT({vars},["titles"], [{"labels"}], [{"helps"}], [{reset_values}], [{initial_values}])
		var -> {var_name, real, [{pos}]}
		var -> {var_name, [allowed_types_matrix], [{pos}]}
		var -> {var_name, {choose_items}, [{pos}]}

Help Topics Tree 13217	Help Text
	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 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, var 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 format: {var_name, real, [{pos}]} to create a checkbox control. If real is >1, then this checkbox gets pooled with the next n -1 checkboxes in a radio group (i.e., only one of the n checkboxes 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, then the edition will hide 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 percentage, field width in screen percentage, line (starts at 0) }. This allows you to control precisely the 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 7 lines of controls per page. Controls with more than 7 lines will be placed in subsequent pages. If more than one page is created, titles can be a list of titles.
INSTRING	In String Syntax: INSTRING(String1, String2) Returns the index of the first occurrence of String2 in String1. Returns 0 if String2 is not present in String1. Note that the first character in a string is position 1. Examples: INSTRING("vanilla", "van") → 1 INSTRING("banana","na") → 3 INSTRING("ab","abc") → 0 INSTRING({"vanilla","banana","ab"},{"van","ana","abc"}) → {1,2,0}
int	Integrate Syntax: int(Expr,[Var],[Real1,Real2]) 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. Examples: int(1/x) → ln(abs(x)) int(sin(x),x,0,π) → 2 int(1/(1-x^4),x,2,3)) → -1/4*(2*atan(2)+ln(3))+1/4*(2*atan(3)-ln(2)+ln(4))
inter	Intersections Syntax: inter(Curve1, Curve2) Returns the intersections of two curves as a vector. Example: inter(8-x²/6,x/2-1) → [[6, 2] [-9, -11/2]], indicating that there are two intersections-one at (6,2) and the other at (-9,-11/2).
interval2center	Syntax: interval2center(Interval) or interval2center(Object) Returns the center of an interval or object. Example: interval2center(2..5) → 7/2
inv	Inverse Syntax: inv(Expr) or inv(Matrix) Returns the inverse of an expression or matrix. Examples: inv(9/5) → 5/9 inv([[1,2],[3,4]]) → [[-2 1], [3/2 -1/2]]
INVERSE	Square matrix Inverse Syntax: Matrix^(-1) Inverts a square matrix. If Complex mode is on, the matrix may contain complex elements.
inversion	Syntax: inversion(Point1, Realk, Point2) Draws the inversion of a point, with respect to another point, by a scale factor. Example:

Help Topics Tree 13217	Help Text
	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.
INVERT	Invert GROB Syntax: INVERT([G, x1, y1, x2, y2]) Inverts the rectangle on G defined by the diagonal points (x1, y1) and (x2, y2). The effect is reverse video. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G If only one (x,y) pair is specified, it refers to the top left corner of G. Example: Demo_INVERT
INVERT_P	Invert GROB Syntax: INVERT_P([G, x1, y1, x2, y2]) Inverts the rectangle on G defined by the diagonal points (x1, y1) and (x2, y2). The effect is reverse video. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G If only one (x,y) pair is specified, it refers to the top left corner of G. Example: Demo_INVERT_P
invlaplace	Inverse Laplace Transform Syntax: invlaplace(Expr,[Var],[IlapVar]) Returns the inverse Laplace transform of an expression. Example: invlaplace(1/(x^2+1)^2) → (-x/2)*cos(x)+(1/2)*sin(x)
invztrans	Inverse Z Transform Syntax: invztrans(Expr,[Var],[InvZtransVar]) Returns the inverse z transform of a rational fraction. Examples: invztrans(1/(x^2+1)^2) → (1/4)*(x*e^((i/2)*π*x)+x*e^((−i/2)*π*x)-2*e^((i/2)*π*x)-2*e^((−i/2)*π*x))*Heaviside(x-1) invztrans(z/(z^4-1),z,n)
IP	Integer Part Syntax: IP(value) Returns the Integer part of value. Examples: IP(23.2) → 23 IP(-23.2) → -23 IP({23.2,15+1/4,51/2,10-4/5}) → {23,15,25,9}
iPart	Integer Part Syntax: iPart(Real) or iPart(List) Returns a real number without its fractional part or a list of real numbers each without its fractional part. Examples: iPart(4.3) → 4 iPart(4.3,√2)
iquo	Integer Euclidian Quotient Syntax: iquo(Intgr1, Intgr2) Returns the integer quotient of the Euclidean division of two integers. Examples: iquo(148,5) → 29 iquo(25+12*i,5+7*i) → 3-2*i
iquorem	Integer Quotient and Remainder Syntax: iquorem(Integer1, Integer2)) Returns the Euclidean quotient and remainder of two integers. Example:

Help Topics Tree 13217	Help Text
	iquorem(63,23) → [2,17]
irem	Integer Euclidian Remainder Syntax: irem(Intgr1, Intgr2) Returns the integer remainder from the Euclidean division of two integers. Examples: irem(148,5) → 3 irem(25+12*i,5+7*i) → -4+i
is_collinear	is_collinear Function Syntax: is_collinear(Point1, Point2, ..., Pointn) 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. Example: is_collinear(point(0,0), point(5,0), point(6,1)) → 0
is_concyclic	is_concyclic Function Syntax: is_concyclic(Point1, Point2, Point3, Point4)) Takes a set of 4 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. Example: is_concyclic(point(-4,-2), point(-4,2), point(4,-2), point(4,2)) → 1
is_conjugate	is_conjugate Function Syntax: is_conjugate(Circle, Point1, Point2, [Point3]) or is_conjugate(Line1, Line2, Line3, [Line4]) Tests whether or not two points or two lines are conjugates for the given circle. Returns 1 if they are and 0 otherwise.
is_coplanar	is_coplanar Function Syntax: is_coplanar(Point1, Point2, Point3, Point4) Tests if four points are in the same plane. Returns 1 if true or 0 if false.
is_cycle	is_cycle Function Syntax: is_cycle(list) Tests whether or not list is a cycle. Returns 1 if it is, and 0 otherwise. Examples: is_cycle([2,1,3,5]) → 1 is_cycle([2,0,3,2]) → 0
is_element	is_element Function Syntax: is_element(Point, Object) Tests if a point is on a geometric object. Returns a number (1 to number of sides) representing the segment containing the point and 0 otherwise. Examples: is_element(point((√(2)/2),(√(2)/2)),circle(0,1)) → 1 is_element(point(0,0.5),square(0,1)) → 4
is_equilateral	is_equilateral Function Syntax: is_equilateral(Point1, Point2, Point3) Takes three points and tests whether or not they are vertices of a single equilateral triangle. Returns 1 if they are and 0 otherwise. Example: is_equilateral(triangle(0,2,1+i*√3)) → 1
is_harmonic	Syntax: is_harmonic(Point1, Point2, Point3, Point4) Tests whether or not four points are in a harmonic division or range. Returns 1 if they are or 0 otherwise. Example: is_harmonic(0,1+i,1,i) → 0
is_harmonic_circle_bundle	Syntax: is_harmonic_circle_bundle(Circle1, Circle2, ..., Circlen) Returns 1 if the circles have a common external tangent 2 if they have the same center 3 if they are all the same circle 0 if none of the above Example: is_harmonic_circle_bundle([circle(0,1+i),circle(2,1+i),circle(1+i,point(1-i))]) → 1
is_harmonic_line_bundle	Syntax:

Help Topics Tree	13217	Help Text
		is_harmonic_line_bundle(Line1, Line2, ..., Linen) Returns: 1 if the lines have a common point 2 if they are all parallel 3 if they are all the same line 0 otherwise Example: is_harmonic_line_bundle(line(x+2*y=3), line(2*x+4*y=6)) → 3
is_isosceles		is_isosceles Function Syntax: is_isosceles(Point1, Point2, Point3) 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. Examples: is_isosceles(point(0,0), point(4,0), point(2,4)) → 3 is_isosceles(triangle(0,i,1+i)) → 2
is_orthogonal		is_orthogonal Function Syntax: is_orthogonal(Line1, Line2) or is_orthogonal(Circle1, Circle2) 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. Example: is_orthogonal(line(y=x),line(y=-x)) → 1
is_parallel		is_parallel Function Syntax: is_parallel(Line1, Line2) Tests whether or not two lines are parallel. Returns 1 if they are and 0 otherwise. Example: is_parallel(line(2x+3y=7),line(2x+3y=9) → 1
is_parallelogram		is_parallelogram Function Syntax: is_parallelogram(Point1, Point2, Point3, Point4) 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. Example: is_parallelogram(point(0,0), point(2,4), point(0,8), point(-2,4)) → 2
is_permu		is_permu Function Syntax: is_permu(list) Tests whether or not list is a permutation. Returns 1 if it is, and 0 otherwise. Examples: is_permu([3,1,5,4,2]) → 1 is_permu([3,1,5,4]) → 0
is_perpendicular		is_perpendicular Function Syntax: is_perpendicular(Line1, Line2) Similar to is_orthogonal. Tests whether or not two lines are perpendicular. Returns 1 if they are or 0 if they are not. Example: is_perpendicular(line(y=x),line(y=-x)) → 1
is_rectangle		is_rectangle Function Syntax: is_rectangle(Point1, Point2, Point3, Point4) 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. Example: is_rectangle(point(0,0), point(4,2), point(2,6), point(-2,4)) → 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_rhombus		Syntax: is_rhombus(Point1, Point2, Point3, Point4) Tests whether or not a set of four points are vertices of a rhombus. Returns 0 if they are not 1 if they are 2 if they are vertices of a square

Help Topics Tree 13217	Help Text
	Example: is_rhombus(point(0,3), point(3,0), point(0,-3), point(-3,0)) → 2
is_square	is_square Function Syntax: is_square(Point1, Point2, Point3, Point4) Tests whether or not a set of four points are vertices of a square. Returns 1 if they are and 0 otherwise. Example: is_square(point(0,0),point(4,2), point(2,6),point(-2,4)) → 1
ISKEYDOWN	Is Key Pressed Syntax: ISKEYDOWN(KeyIdentifier) Returns true (non-zero) if the key whose KeyIdentifier is provided is currently pressed, and false (0) if it is not.
ismith	Smith Normal Syntax: ismith(Matrix_A) Given Matrix_A, returns the Smith normal form of a matrix with coefficients in Z. Returns [U V B] such that U and V are invertible in Z, B is the diagonal, B[i,i] divides B[i+1,i+1] and B=U*A*V. Example: ismith([[1,2,3],[4,5,6],[7,8,9]]) → [[1,0,0],[4,-1,0],[-1,2,-1]],[[1,0,0],[0,3,0],[0,0,0]],[[1,-2,1],[0,1,-2],[0,0,1]]
isobarycenter	isobarycenter Function Syntax: isobarycenter(Point1, Point2, …, Pointn) Returns the hypothetical center of mass of a set of points. Works like barycenter but assumes all points have equal weight. Example: isobarycenter(-1,1-i,i) → point(0,0)
isopolygon	Regular Polygon Syntax: isopolygon(Point1, Point2, Realn) 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. Examples: isopolygon(point(0,0,0),point(3,3,3),point(0,0,3),-5) isopolygon(GA,GB,6) draws a regular hexagon whose first two vertices are the points A and B.
isosceles_triangle	Isosceles Triangle Syntax: isosceles_triangle(Point1, Point2, Angle, [Var]) 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. 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.
isprime	Primality Test Syntax: isprime(Integer) Returns true if the integer is prime; otherwise, returns false. Examples: isprime(1999) → 1 isprime(42) → 0
ITERATE	Iterate Expression Syntax: ITERATE(expr, var, ivalue, times) For times, recursively evaluates expr in terms of var, beginning with var = ivalue. Examples: ITERATE(X^2, X, 2, 3) → 256 ITERATE({'X^2','Y^3','Z+1'},{'X','Y','Z'},{2,3,4},{3,2,3}) → {256,19683,7}
ithprime	Ith Prime Syntax: ithprime(Integer) Given an integer n, returns the nth prime number, where n is between 1 and 200,000. Example:

Help Topics Tree	13217	Help Text
		ithprime(5) → 11
	jacobi_symbol	Jacobi Symbol Syntax: jacobi_symbol(Integer1, Integer2) Returns the Jacobi symbol of the two given integers. Example: jacobi_symbol(132,5) → -1
	jordan	Syntax: jordan(Matrix) Returns the list made by the matrix of passage and the Jordan form of a matrix. Examples: jordan([[0,2],[1,0]]) → [[√2,-√2],[1,1]],[[√2,0],[0,-√2]] jordan([[-2,-2,1],[-2,1,-2],[1,-2,-2]])
	JordanBlock	Jordan Block Syntax: JordanBlock(Expr, n) Returns a square n x n matrix with Expr on the diagonal, 1 above and 0 everywhere else. Examples: JordanBlock(7,3) → [[7,1,0],[0,7,1],[0,0,7]] JordanBlock(x+1,3) → [[x+1,1,0],[0,x+1,1],[0,0,x+1]]
	limit	Syntax: limit(Expr,Var,Val, [Dir]) Returns the limit (2-sided or 1-sided) of the given expression as the given variable approaches a value. The optional argument Dir indicates a two sided limit if 0, one sided from below if -1, and one sided from above if 1. If the fourth argument is not provided, the limit returned is bidirectional. Examples: limit((n*tan(x)-tan(n*x))/(sin(n*x)-n*sin(x)),x,0) → 2 limit(sin(x)/(x^2-3*x),x,0) → -1/3 limit(exp(1/x),x,0,1) → +∞
	→HMS	Syntax: →HMS(value) Displays a decimal value in sexagesimal format; that is, in units subdivided into groups of 60. This includes degrees, minutes, and seconds as well as hours, minutes, and seconds. Examples: →HMS(8.5) → 8°30' →HMS({8.5,37.7539}) → {8°30'00",37°45'14.04"}
K-O		Function Catalog K-O Toolbox function catalog K-O
	+/-	Negative Syntax: - Value or - Expression Unary minus. Changes the sign of Value or Expression. Used to enter negative numbers.
	ker	Kernel of Matrix Syntax: ker(Mtrx(M)) Returns the kernel of a linear application of a matrix. Example: ker([[1,2],[3,6]]) → [2,-1]
	KILL	Stop Execution Syntax: KILL; Stops the execution of a program. Example: Demo_KILL
	kolmogorovd	Kolmogorov-Smirnov distribution Returns the Kolmogorov-Smirnov distribution value. 1-2*sum((-1)^(k-1)*exp(-k^2*x^2),k,1,∞) Example: kolmogorovd(1.36) → 0.950514123245
	kolmogorovt	Kolmogorov-Smirnov test Syntax: kolmogorovt(list1,list2) kolmogorovt(list1,distribution_law) Kolmogorov-Smirnov equality test to a continuous distribution law - either between two samples list1 and list2 from an unknown distribution, or between a sample list1 and a specific distribution_law.

Help Topics Tree 13217	Help Text

	Examples:
	kolmogorovt(randvector(100,normald,0,1),randvector(100,normald,0,1))
	kolmogorovt(randvector(100,normald,0,1),randvector(100,normald,3,1))
	kolmogorovt(randvector(100,normald,0,1),normald(0,1))
	kolmogorovt(randvector(100,normald,0,1),student(2))

l1norm	L^1 Norm Syntax: l1norm(Vector) Returns the L^1 norm (sum of the absolute values of the coordinates) of a vector. Example: l1norm([3,-4,2]) → 9
l2norm	L^2 Norm Syntax: l2norm(Vector) Returns the L^2 norm (sqrt(x1²+x2²+…xn²)) of a vector. Example: l2norm([3,4,-2]) → √29
lagrange	Lagrange Polynomial Syntax: lagrange([X1, X2,... Xn], [Y1, Y2, ... Yn]) or lagrange(Matrix) 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. Returns the polynomial of degree n-1 such that P(xk)=yk, for k=0, 1, …, n-1. Example: lagrange([[1,3],[0,1]]) → (1/2)*(x-1)
laguerre	Laguerre Polynomial Syntax: laguerre(Integer) Given an integer n, returns the Laguerre polynomial of degree n. Example: laguerre(2) → -a*x+1/2*a^2+1/2*x^2+3/2*a-2*x+1
laplace	Laplace Transform Syntax: laplace(Expr,[Var],[LapVar]) Returns the Laplace transform of an expression. Examples: laplace(e^(x)*sin(x)) → 1/(x²-2*x+2) laplace(sin(x)^2,x,s) → 2/(s³+4*s)
laplacian	Syntax: laplacian(Expr, Vector) Returns the Laplacian of an expression with respect to a vector of variables. Example: laplacian(e^(z)*cos(x*y),[x,y,z]) → e^(z)*cos(x*y)-x²*e^(z)*cos(x*y)-y²*e^(z)*cos(x*y)
latex	Generate Latex Text Syntax: latex(Expr) Returns the evaluated CAS expression written in Latex format. Examples: latex(1/2) → "\frac{1}{2}" latex((x^4-1)/(x^2+3)) → "\frac{(x^{4}-1)}{(x^{2}+3)}"
lcm	Lowest Common Multiple Syntax: lcm(Intgr1, Intgr2, ...) or lcm(Poly1, Poly2, ...) or lcm(Rational1, Rational2, ...) Returns the lowest common multiple of two or more polynomials of several variables, or of two or more integers, or of two or more rationals. Examples: lcm(6,4) → 12 lcm(x²-2*x+1,x^3-1) → (x-1)*(x³-1)

lcoeff	Syntax:
	lcoeff(Poly)
	lcoeff(Vector)
	lcoeff(List)
	Returns the coefficient of the term of highest degree of a polynomial. The polynomial can be expressed in symbolic form or as a vector or list of coefficients.
	Examples:

Help Topics Tree		13217	Help Text
			lcoeff(1-2*x^3+x^2+7*x) → -2 lcoeff([-2,1,7,0]) → -2
		left	Left Side of Equation Syntax: left(Expr1=Expr2) or left(Real1..Real2) Returns the left side of an equation or the left end of an interval. Example: left(x^2-1=2*x+3) → x^2-1
		LEFT	Left Part Syntax: LEFT(String, Integer) Given a string and an integer n, return the first n characters of the string. If n ≥ DIM(str) or n ≤ 0, returns the entire string. Example: LEFT("MOMOGUMBO",3) → "MOM"
		legendre	Legendre Polynomial Syntax: legendre(Integer) Given an integer n, returns the Legendre polynomial of degree n. Example: legendre(4) → 35/8*x^4-15/4*x^2+3/8
		legendre_symbol	Legendre symbol Syntax: legendre_symbol(Integer1, [Integer2]) Given two integers, returns the Legendre symbol of the second integer, using the Legendre polynomial whose degree is the first integer. Example: legendre_symbol(132,5) → -1
		length	Syntax: length(List) or length(String) or length(Object1) Returns the length of a list, a string or a set of objects. Examples: length([1,2,3]) → 3 length("12345") → 5 length(x^2+5*x-1) → 3
		lgcd	Greatest Common Divisor Syntax: lgcd(List) or lgcd(Vector) or lgcd(Integer1, Integer2, …) or lgcd(Poly1, Poly2, …) Returns the greatest common divisor of a set of integers or polynomials, contained in a list, a vector, or just entered directly as arguments. Examples: lgcd({45,75,20,15}) → 5 lgcd(x^2-2*x+1,x^3-1,x-1) → x-1
		lin	Syntax: lin(Expr) Linearization of exponentials. Returns an expression with the exponentials linearized. Example: lin((e^(x)^3+e^(x))^2) → e^(2*x^3)+2*e^(x^3+x)+e^(2*x)
		LINE	Line Drawing Syntax: LINE([G], x1, y1, x2, y2, [color]) LINE([G],points_definition, lines_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) LINE([G],pre_rotated_points, line_definitions, [zstring]) The basic form of LINE draws one line between specified coordinates in the graphic using the specified color. The advanced form of LINE allows the rendering of multiple lines at a time with a potential 3D transformation of the points that define the line. 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 2 to 4 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).

Help Topics Tree 13217	Help Text
	lines_definitions is either a list or a matrix of line definitions. Each line is defined by 2 to 4 numbers. p1, p2, color and alpha. p1 and p2 are the index in the points_definition of the 2 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 re-specifying the same color for each line. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the points using the usual 3D or 4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed objects. Each point is rotated and translated through a multiplication by rotation_matrix. It is then projected on the view plane using the eye position according to 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 provided easier zClipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). LINE 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 by this string and omitting the transformation definition in subsequent calls to TRIANGLE and LINE. About ZString TRIANGLE([G]) returns a string adapted for z clipping. To use Z clipping, call TRIANGLE to create a Z clipping string (initialized at 255 for each pixels). You can then call LINE with appropriate z (0-255) values for each of the triangle vertexes and LINE will not draw pixels further than the already drawn pixels. ZString is automatically updated as appropriate. Example: Demo_LINE
line	Syntax: line(Point1, Point2) or line(a*x+b*y+c) or line(point1, slope=realm) Draws a line in the Plot view of the Geometry app or returns the equation of a line in CAS view. The arguments can be two points, a linear expression of the form a*x+b*y+c, or a point and a slope. Examples: line(2+i,3+2*i) 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
LINE_P	Line Drawing Syntax: LINE_P([G], x1, y1, x2, y2, [color]) LINE_P([G],points_definition, lines_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) LINE_P([G],pre_rotated_points, line_definitions, [zstring]) The basic form of LINE_P draws one line between specified coordinates in the graphic using the specified color. The advanced form of LINE_P allows the rendering of multiple lines at a time with a potential 3D transformation of the points that define the line. 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 2 to 4 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). lines_definitions is either a list or a matrix of line definitions. Each line is defined by 2 to 4 numbers. p1, p2, color and alpha. p1 and p2 are the index in the points_definition of the 2 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 re-specifying the same color for each line. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the points using the usual 3D or 4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed objects. Each point is rotated and translated through a multiplication by rotation_matrix. It is then projected on the view plane using the eye position using 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 provided easier zClipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below).

Help Topics Tree 13217	Help Text
	LINE_P returns a string which contains all the transformed points. If you plan to call TRIANGLE_P or LINE_P multiple times in a row using the same points and transformation, you can do so by replacing the points_definition by this string and omitting the transformation definition in subsequent calls to TRIANGLE_P and LINE_P. 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 vertexes and LINE_P will not draw pixels further than the already drawn pixels. ZString is automatically updated as appropriate. Example: Demo_LINE_P
linear_interpolate	Syntax: linear_interpolate(Matrix,Xmin,Xmax,Xstep) Makes a regular sample from a polygonal line defined by a two row matrix. Example: linear_interpolate([[1,2,6,9],[3,4,6,7]],1,9,1) → [[1. 2. 3. 4. 5. 6. 7. 8. 9.],[3 4 4.5 5 5.5 6 6.33333333333 6.66666666667 7]]
linear_regression	Linear Regression Syntax: linear_regression(Matrix) or linear_regression(List1, List2) Given a set of points, returns a vector containing the coefficients a and b of y=a*x+b, the line which best fits the set of points. The points may be the elements in two lists or the rows of a matrix. Example: linear_regression([0.0,1.0,2.0,3.0,4.0],[0.0,1.0,4.0,9.0,16.0]) → [4. -2]
LineHorz	Horizontal Line Syntax: LineHorz(Exp) or LineHorz(Real) 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. Example: LineHorz(-1) draws the line whose equation is y=-1
LineTan	LineTan Function Syntax: LineTan(f(x), [Var], Value) Draws the tangent to y=f(Var) at Var=Value. Example: LineTan(x^2-x,1) returns line(y=x-1); that is, the line tangent to the graph of y=x^2-x at x=1
LineVert	Vertical Line Syntax: LineVert(Expr) or LineVert(Real) 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. Example: LineVert(2) draws the line whose equation is x=2
linsolve	Linear System Solver Syntax: linsolve([LinEq1, LinEq2,…LinEqn], [Var1,Var2,…Varn]) Given a vector of linear equations and a corresponding vector of variables, returns the solution to the system of linear equations. Example: linsolve([x+y+z=1,x-y=2,2*x-z=3],[x,y,z]) → [3/2,-1/2,0]
list2mat	Syntax: list2mat(List, Integer) Given a list and an integer n, 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. Example: list2mat([1,8,4,9],2) → [[1,8],[4,9]]
lll_reduce	LLL Reduction Syntax: lll_reduce(Matrix) Implementation of the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm. Takes as argument an invertable matrix with integer coefficients. Returns (S, A, L, O) such that: • the rows of S is a short basis of the Z-module generated by the rows of M • A is the change-of-basis matrix from the short basis to the basis defined by the rows of M(A*M=S)

Help Topics Tree 13217	Help Text
	• L is a lower triangular matrix and the modulus of it’s non diagonal coefficients are less than 1/2 • O is a matrix with orthogonal rows such that L*O = S Example: lll_reduce([[1234,3452,4521],[3425,2241,1543],[5643,3425,8721]])
LN	Natural Logarithm Syntax: LN(value) Natural Logarithmic function Returns the logarithm of value in base e, Euler's number. Examples: LN(1) → 0 LN(2+3*i) → 1.28247467873+0.982793723247*i LN({0.1,1}) → {−2.30258509299,0}
lname	List Variable Names Syntax: lname('Expr') Returns a list of the variables in an expression, which must be contained in single quotation marks ('). In CAS, this command and all variables must be in lower case. In Home, the command and variables must be in upper case. Examples: lname('e^(x)*2*sin(y)') → [x y] lname({'e^x*2*sin(y)','x^(x-3)*z-4*q*t'}) → [x,y,z,q,t] LNAME('e^X*2*SIN(Y)') → {X,Y}
lncollect	Collect Logarithms Syntax: lncollect(Expr) Rewrites an expression with the logarithms collected. Applies ln(a)+n*ln(b)=ln(a*b^n) where n is an integer. Example: lncollect(ln(x)+2*ln(y)) → ln(x*y²)
lnexpand	Expand Logarithm Syntax: lnexpand(Expr) Returns the expanded form of a logarithmic expression. Example: lnexpand(ln(3*x)) → ln(3)+ln(x)
LNP1	Natural Log Plus 1 Syntax: LNP1(value) Natural log plus 1: LN(X+1) This is more accurate than the natural logarithm function for values close to zero. Examples: LNP1(0.23) → 0.207014169384 LNP1(0.02+0.03*i) → 2.02349662769ᴇ−2+0.029403288204*i
LOCAL	LOCAL keyword Syntax: LOCAL Var1[:=Val1, Var2:=Val2, ... Var8:=Val8]; Declares one or more local variables. Each variable can be assigned an optional initial value as well. If the declaration is in a function block, these variables will be local to the function. If the declaration is in the main program body, the variables are local to the program. There can only be 8 variables per LOCAL keyword. To create more variables, you must add another LOCAL keyword. Examples: Demo_LOCAL
locus	Syntax: locus(Point,Element, [tstep=Value])) 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. The optional tstep statement can be used to control the default level of detail.
LOG	General Logarithm Syntax: LOG(value, [base]) General logarithmic function Returns the logarithm of value in base. By default, base=10. Examples: LOG(8) → 0.903089986992 LOG(8,2) → 3 LOG(2+3*i) → 0.556971676153+0.426821890855*i LOG(2+3*i,2) → 1.85021985907+1.41787163075*i

Help Topics Tree 13217	Help Text

	LOG({100,10}) → {2,1}
	LOG({8,27,10000},{2,3,10}) → {3,3,4}

log10	Common Logarithm Syntax: log10(Expr) Common logarithm (base 10). Returns the common logarithm of an expression. Example: log10(10) → 1
logarithmic_regression	Logarithmic Regression Syntax: logarithmic_regression(Matrix) or logarithmic_regression(List1, List2) 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. Example: logarithmic_regression([[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]]) → 10.1506450002,-0.564824055818
logb	Syntax: logb(a,b) Given a real number a and an integer b, returns the logarithm of a in the base b. Example: logb(5, 2) → ln(5)/ln(2), ~2.32192809489
logistic_regression	Logistic Regression Syntax: logistic_regression(Lst(L),Real(x0),Real(y0)) Returns [y,y',C,y'max,xmax,R] where y is a logistic function (solution of y'/y=a*y+b), such that y(x0)=y0 and where [y'(x0),y'(x0+1)...] is the best approximation of L. Example: logistic_regression([0.0,1.0,2.0,3.0,4.0],0.0,1.0) → [-17.77/(1+exp(-0.496893925384*x+2.82232341488+3.14159265359*i)),-2.48542227469/(1+cosh(-0.496893925384*x+2.82232341488+3.14159265359*i))]
LOWER	Lowercase Syntax: LOWER(string) Returns string with uppercase characters converted to lowercase. Examples: LOWER("ABC") → "abc" LOWER("ΑΒΓ") → "αβγ"
LQ	LQ Factorization Syntax: LQ(matrix) Factorizes a m × n matrix into three matrices: L, Q, and P, where L is an m × n lower trapezoidal, Q is an n × n orthogonal, and P is an m × m permutation; and P*A=L*Q. Example: LQ([[1,2],[3,4]]) → {[[2.2360,0],[4.9193,0.8944]],[[0.4472,0.8944],[0.8944,-0.4472]],[[1,0],[0,1]]}
LSQ	Least Squares Syntax: LSQ(matrix1, matrix2) Returns the minimum norm least squares matrix (or vector) corresponding to the system matrix1*X=matrix2 Examples: LSQ([[1,2],[3,4]],[[5],[11]]) → [[1],[2]] LSQ([[1,2],[3,4]],[[5,-1],[11,-1]]) → [[1,1],[2,-1]]
LU	LU Decomposition Syntax: LU(matrix) Factorizes a square matrix into three matrices L, U, and P, where L is a lowertriangular, U is an uppertriangular, and P is the permutation; and P*A=L*U. Example: LU([[1,2],[3,4]]) → {[[1,0],[0.3333,1]],[[3,4],[0,0.6666],[0,1],[1,0]]}
lvar	List of Variables & Expressions Syntax: lvar(Expr) Given an expression, returns a list of the functions of the expression which utilize variables, including occurrences of the variables themselves. Example: lvar(e^(x)*2*sin(y)+ln(x)) → [e^(x) sin(y) ln(x)]
MAKELIST	Make List Syntax: MAKELIST(expression, variable, begin, end, [increment])

Help Topics Tree 13217	Help Text
	Calculates a sequence of elements for a new list. Evaluates expression, incrementing variable from begin to end values, using increment steps (default is 1). Example: MAKELIST(2*X-1,X,1,5,1) → {1,3,5,7,9}
MAKEMAT	Make Matrix Syntax: MAKEMAT(Expr, Rows, Columns) or MAKEMAT(Expr, Elements) Creates a matrix of dimension Rows × Columns, using Expr to calculate each element. If Expr 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 using the number of Elements instead of the number of rows and columns. Examples: MAKEMAT(0,3,3) → [[0,0,0],[0,0,0],[0,0,0]] MAKEMAT(√2,2,3) → [[√2,√2,√2],[√2,√2,√2]] in CAS view MAKEMAT(I+J-1,2,3) → [[1,2,3],[2,3,4]] in Home view
MANT	Mantissa Syntax: MANT(Value) Returns the significant digits of Value. Examples: MANT(21.2E34) → 2.12 MANT({2.12ᴇ35,5302.00000123}) → {2.12,5.30200000123}
map	map Function Syntax: map(Matrix, Var→Function) or map(Matrix, Var→Test) There are two uses for this function, whose second argument is always a mapping of a variable onto an expression. Examples: map([1,2,3], x→x^3) → [1,8,27] If the expression is a function of the variable, then the function is applied to each element in the vector or matrix (the first argument) and the resulting vector or matrix is returned. map([1,2,3], x→x>1) → [0,1,1] If the expression is a Boolean test, then 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).
mat2list	Syntax: mat2list(Matrix) Returns a list containing the elements of the given matrix. Example: mat2list([[1,8],[4,9]]) → [1,8,4,9]
matpow	Syntax: matpow(Matrix,Int(n)) Calculates the n power of a matrix by use of the Jordan normal form. Example: matpow([[1,2],[3,4]],n) → [[(√33-3)*((√33+5)/2)^n*-6/(-12*√33)+(-(√33)-3)*((-(√33)+5)/2)^n*6/(-12*√33),(√33-3)*((√33+5)/2)^n*(-(√33)-3)/(-12*√33)+(-(√33)-3)*((-(√33)+5)/2)^n*(-(√33)+3)/(-12*√33)],[6*((√33+5)/2)^n*-6/(-12*√33)+6*((-(√33)+5)/2)^n*6/(-12*√33),6*((√33+5)/2)^n*(-(√33)-3)/(-12*√33)+6*((-(√33)+5)/2)^n*(-(√33)+3)/(-12*√33)]]
MAX	Maximum Syntax: MAX(value1,[value2],[..value16]) or MAX(list) Returns the greatest of the values given, or the greatest value of a list. Examples: MAX(210,25) → 210 MAX({1,8,2}) → 8 MAX(8/3,11/4) → 2.75 MAX({1,8,2},{2,4,6}) → {2,8,6}
maxnorm	Max Norm Syntax: maxnorm(Vector) or maxnorm(Matrix) Returns the l∞ norm (the maximum of the absolute values of the coordinates) of a vector or matrix. Examples: maxnorm([1,2]) → 2 maxnorm([[1,2],[3,-4]]) → 4

Help Topics Tree 13217	Help Text
MAXREAL	Maximum Real Syntax: MAXREAL Returns the maximum real number that the HP Prime is capable of representing in Home and CAS views. In CAS view, MAXREAL=1.79769313486ᴇ308 In Home view, MAXREAL=9.99999999999ᴇ499 Example: MAXREAL
mean	Syntax: mean(List1, [List2]) or mean(Matrix) 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. Examples: mean([1,2,3],[1,2,3]) → 7/3 mean([[1,2,3],[4,5,6]]) → [5/2,7/2,9/2]
median	Syntax: median(List1, [List2]) or median(Matrix) Returns the median of a list or vector (with an optional list as a list of weights). With a matrix as argument, returns the medians of the columns. Example: median([0,1,3,4,2,5,6]) → 3
median_line	Median Syntax: median_line(Point1, Point2, Point3) Given three points that define a triangle, draws the median of the triangle that passes through the first point and contains the midpoint of the segment defined by the other two points. In CAS view, returns the equation of the median line. Example: median_line(0,8*i,4) draws the line whose equation is y=2x; that is, the line through the first vertex of the triangle at (0,0) and the point at (2,4), the midpoint of the segment with endpoints (0, 8) and (4, 0).
member	Syntax: member(Element, List) or member(Element, Vector) 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. Example: member(1,[4,3,1,2]) → 3
MEMORY	System Memory Syntax: MEMORY MEMORY(n) Returns a list containing integers representing memory and storage space, or the individual integer at position n. Examples: MEMORY() MEMORY(1)
MID	Middle Syntax: MID(String, Position, [n]) Extracts n characters from String starting at Position. If n is not specified, then MID extracts the remainder of String from Position. Examples: MID("MOMOGUMBO",3,5) → "MOGUM" MID("PUDGE",4) → "GE"
midpoint	Syntax: midpoint(Segment) or midpoint(Point1, Point2) 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. Example: midpoint(0,6+6i) → point(3,3)
MIN	Minimum Syntax: MIN(value1,[value2],[..value16]) or MIN(list) Returns the least of the values given, or the least value of a list.

Help Topics Tree 13217	Help Text
	Examples: MIN(210,25) → 25 MIN({1,8,2}) → 1 MIN(8/3,11/4) → 2.6667 MIN({1,8,2},{2,4,6}) → {1,4,2}
MINREAL	Minimum Real Syntax: MINREAL Returns the minimum real number (closest to zero) that the HP Prime is capable of representing in Home and CAS views. In CAS view, MINREAL=2.22507385851ᴇ-308 In Home view, MINREAL=1ᴇ-499 Example: MINREAL
mkisom	Isometry Syntax: mkisom(Vect,(Sign(1) or -1)) Returns the matrix of an isometry given by its proper elements. Examples: mkisom(π,1) → [[-1,0],[0,-1]] (in radians mode)
MKSA	Convert to Metric System Syntax: MKSA(Value_Unit) Converts the measurement Value_Unit to its corresponding value and unit in Unit's MKSA equivalent. MKSA stands for the Meter-Kilogram-Second-Ampere system. Examples: MKSA(32_yd) → 29.2608_m MKSA(75_mph) →33.528_m/s MKSA({33_(cm),4_(yd^3)}) → {0.33_m,3.05821943194_(m^3)}
MOD	Modulo Syntax: value1 MOD value2 Returns the remainder of the Euclidean division value1/value2. Examples: 9 MOD 4 → 1 #27o MOD 12 → 11 [[1,3],[13,14]] MOD 4 → [[1,3],[1,2]] {11,12,13,15,17} MOD 4 → {3,0,3,1,3}
modgcd	Syntax: modgcd(Poly1, Poly2) Uses the modular algorithm to return the greatest common divisor of two polynomials. Example: modgcd(x^4-1,(x-1)^2) → x-1
MOUSE	Get Touch Event 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 z, 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).
mRow	Multiply Row Syntax: mRow(Expr, Matrix, Integer) Given an expression, a matrix, and an integer n, multiplies row n of the matrix by the expression. Example: mRow(12,[[1,2],[3,4],[5,6]],1) → [[12,24],[3,4],[5,6]]
MSGBOX	Message Box Syntax: MSGBOX(expr,[OK_Cancel]) or MSGBOX(string,[OK_Cancel]) Displays a message box with either the value of expr or string. If OK_Cancel is true, displays OK and CANCEL menu keys, otherwise only displays the OK menu key. Default value for OK_Cancel is false. Returns true (non-zero) if the user presses OK, false (0) if the user presses CANCEL. Example: MSGBOX("Click OK to continue")

Help Topics Tree 13217	Help Text
mult_c_conjugate	Syntax: mult_c_conjugate(Expr) 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. Example: mult_c_conjugate(1/(3+i*2)) → (3-i*2)/((3+i*2)*(3-i*2))
mult_conjugate	Syntax: mult_conjugate(Expr) 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. Example: mult_conjugate(1/(√3-√2)) → ((√3+√2)/((√3-(√2)*(√3+√2))

multinomial	Multinomial Distribution
	Syntax:
	multinomial(Integern, VectorP, VectorK)
	Computes the probability of VectorK successes out of Integern trials, each with a probability of success of VectorP.
	The multinomial distribution is a more general form of the Binomial Distribution where each outcome can have two or more possible outcomes.
	Examples:
	multinomial(10,[0.5,0.5],[3,7])
	multinomial(10,[0.2,0.3,0.5],[1,3,6])
	randvector(3,multinomial,[1/2,1/3,1/6])
	ranm(4,3,multinomial,[1/2,1/3,1/6])

nDeriv	Numerical Derivative Syntax: nDeriv(Expr, Var, Real) or nDeriv(Expr, Var1, Var2) 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. Examples: nDeriv(f(x),x,h) → (f(x+h)-(f(x-h)))*0.5/h nDeriv(x^2,x)
negbinomial	Discrete Negative Binomial Syntax: negbinomial(n,k,x) Negative binomial probability density function Computes the probability density of the negative binomial distribution at x given parameters n and k. Example: negbinomial(4,2,.6) → 0.20736
negbinomial_cdf	Cumulative Negative Binomial Syntax: negbinomial_cdf(n,k,x,[x2]) Cumulative negative binomial distribution function Returns the lower-tail probability of the negative binomial probability density function for the value x, given parameters n and k. Examples: negbinomial_cdf(4,.5,2) → 0.34375 negbinomial_cdf(4,.5,2,3) → 0.15625
negbinomial_icdf	Inverse Cumulative Negative Binomial Syntax: negbinomial_icdf(n,k,p) Inverse cumulative negative binomial distribution function Returns the value x such that the negative binomial lower-tail probability of x, given parameters n and k, is p. Example: negbinomial_icdf(4,0.5,0.7) → 5
newton	Newton's Method Syntax: newton(Expr,Var, [Guess],[Integer]) Uses Newton's method to estimate the root of a function, beginning with Guess and calculating Integer iterations. By default, Integer is 20.

Help Topics Tree 13217	Help Text
	Example: newton(3-x^2,x,2) → 1.73205080757
nextprime	Next Prime Syntax: nextprime(Integer) Returns the smallest prime number greater than the argument. Example: nextprime(12) → 13
nop	No Operation The no-operation CAS function. On evaluation, no operation will happen. This function can be useful for some advanced use cases in CAS function programming.
normal	Syntax: normal(Expr) Returns the expanded irreducible form of an expression. Examples: normal(2*x*2) → 4*x normal((2*x+1)^2) → 4*x^2+4*x+1
NORMALD	Normal Density Syntax: NORMALD([μ, σ,] x) Normal probability density function. Computes the probability density at the value x, given the mean, μ, and standard deviation, σ, of a normal distribution. With one argument, x, it returns the probability density for the standard normal distribution at x, assuming a mean of zero and standard deviation of 1. Examples: NORMALD(0.5) → 0.352065326764 NORMALD(0,2,0.5) → 0.193334058401
NORMALD_CDF	Cumulative Normal Syntax: NORMALD_CDF([μ, σ,] x, [x2]) Cumulative normal distribution function. With three values (μ, σ, and x), 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. With the optional fourth value x2, returns the area under the normal probability density function between the two x-values. With one argument x, returns the lower-tail probability of the standard normal probability density function for the value x, assuming a mean of zero and standard deviation of 1. Examples: NORMALD_CDF(2)→0.977249868052 NORMALD_CDF(-1,1)→0.682689492138 NORMALD_CDF(0,1,2) → 0.977249868052 NORMALD_CDF(0,1,0,2) → 0.477249868052
NORMALD_ICDF	Inverse Cumulative Normal Syntax: NORMALD_ICDF([μ, σ,] p) Inverse cumulative normal distribution function. Returns the cumulative normal distribution x-value associated with the lower-tail probability p, given the mean μ, and standard deviation σ, of a normal distribution. With one argument, p, assumes a mean of 0 and a standard deviation of 1. Examples: NORMALD_ICDF(0.977249868052)→2 NORMALD_ICDF(0,1,0.841344746069) → 1
normalize	Syntax: normalize(Vector) or normalize(Complex) Given a vector, returns it divided by its l2 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. It is also an option for plotfield. In this case, the term comes last in the set of arguments and the result is the slopefield segments are given equal length. Examples: normalize(3+4*i) → (3+4*i)/5 normalize([3,4]) → [3/5,4/5]
NOT	Logical NOT Syntax: NOT Value For Real numbers, returns 1 if Value is zero; otherwise returns 0. For Integers and Strings, NOT is performed bitwise, flipping all 1's to 0's and all 0's to 1's. Examples:

Help Topics Tree 13217	Help Text
	NOT 3 → 0 NOT 0 → 1 A:=32; B:=2^5; NOT (A=B) → 0 NOT #DFCA:16h → #2035:16h NOT #1776:16o → #176001:16o NOT "abcdefg" → "ˋﾝﾜﾛﾚﾙﾘ" NOT {"ab","cd"} → {"ˋﾝ","ﾜﾛ"}
numer	Simplified Numerator Syntax: numer(a/b) For integers a and b, returns the numerator of the fraction a/b after simplification. Example: numer(10/12) → 5
odd	Oddness Test Syntax: odd(Integer) Returns 1 if the given integer is odd, otherwise returns 0. Examples: odd(6) → 0 odd(1251) → 1
odesolve	ODE Solver Syntax: odesolve(Expr, VectVar, VectInit, FinalVal, [tstep=Val, curve]) Ordinary Differential Equation solver Solves an ordinary differential equation given by Expr, with variables declared in VectVar and initial conditions for those variables declared in VectInit. 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. Example: odesolve(sin(t*y),[t,y],[0,1],2) → [1.82241255674]
open_polygon	Syntax: open_polygon(point1, point2, …, point1) or open_polygon(point1, point2, …, pointn) 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. Example: open_polygon(point(0,0),point(3,3),point(0,3),point(0,0)) draws a right triangle
OR	Logical OR Syntax: Value1 OR Value2 For Real numbers, returns 1 if either Value1 or Value2 is non-zero; otherwise returns 0. For Integers and Strings, OR is performed bitwise, returning 1 if either corresponding bit is 1, otherwise 0. Examples: 3 OR 2 → 1 0 OR 2 → 1 0 OR 0 → 0 {3,0,0} OR {2,1,0} → {1,1,0} 3_inch==7.62_cm OR 9_(inch²)==58.0644_(cm²) → 1 #CC44h OR #44CCh → #CCCCh "c" OR "d" → "g" X:=0; 0 OR (X:=7); 1 OR (X:=9); X → 7 7 ≤ 3 OR 5 < 9 OR 3 ≠ 2.9 + 0.1 → 1
order_size	Syntax: order_size(Expr) Returns the remainder (O term) of a series expansion: limit(x^a*order_size(x),x=0)=0 if a>0
ordinate	Syntax: ordinate(Point) or ordinate(Vector) Returns the ordinate of a point or the y-length of a vector. Example: ordinate(point(1+2*i)) → 2
orthocenter	orthocenter Function Syntax: orthocenter(Triangle) or orthocenter(Point1, Point2, Point3)

Help Topics Tree	13217	Help Text

		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.
		
		Examples:
		orthocenter(0,4*i,4) → point(0,0)
		orthocenter(triangle(0,1,1+i)) → point(1,0)

	orthogonal	Syntax: orthogonal(Point, Line) or orthogonal(Point, Plane) orthogonal(A,line(B,C)) draws the orthogonal plane of line BC through point A. Example: orthogonal(point(0,0,0),plane(point(1,0,0),point(0,1,0),point(0,0,1)))
	ΔLIST	Δ List Syntax: ΔLIST(list) Creates a new list composed of the first differences of a given list; that is, the differences between the sequential elements in a list. The new list has one fewer elements than the original list. Example: ΔLIST({1,2,3,5,8}) → {1,1,2,3}
	ΠLIST	Π List Syntax: ΠLIST(list) Calculates the product of all elements in a list. Example: ΠLIST({2,3,4}) → 24
	ΣLIST	Σ List Syntax: ΣLIST(list) Calculates the sum of all elements in a list. If the list contains a string, the result will be a single string with all elements concatenated together. Examples: ΣLIST({2,3,4}) → 9 ΣLIST({"A","B","CE"}) → "ABCE" ΣLIST({"A",1,"B",2,"CE",3}) → "A1B2CE3"
	$^{n}\sqrt{}$	Nth Root Key Syntax: Value1 √ Value2 NTHROOT: the nth root function This Shift-key combination brings up a template for the NTHROOT function. It returns the primary Value1 root of Value2. On the keyboard, NTHROOT is represented by $^{n}\sqrt{}$. Examples: 3 NTHROOT 8 → 2 3 NTHROOT 79.507 → 4.3 2.3 NTHROOT 5413.44050218 → 42 2.1 NTHROOT 3+2*i → 1.76999848019+0.508973095403*i (1.2-0.5*i) NTHROOT (0.2+4*i) → 0.137162958212+1.70241905473*i 3 NTHROOT {27,8,64} → {3,2,4}
P-T		Function Catalog P-T Toolbox function catalog P-T
	p1oc2	Syntax: p1oc2(permutation,cycle) Returns the permutation product of permutation and cycle. Example: p1oc2([3,2,4,1],[4,1,2]) → [2,1,4,3]
	p1op2	Syntax: p1op2(permutation1,permutation2) Returns the permutation product of permutation1 and permutation2. Example: p1op2([1,3,2],[2,1,4,3]) → [3,1,4,2]
	pa2b2	Syntax: pa2b2(Integer) Takes a prime integer n congruent to 1 modulo 4 and returns [a,b] such that $a^2+b^2=n$. Examples: pa2b2(17) → [4,1] pa2b2(97) → [9,4]
	pade	Pade Approximation Syntax: pade(Expr, Var, Integern, Integerp)

Help Topics Tree 13217	Help Text
	Returns the Pade approximation of an expression; that is, a rational fraction P/Q such that P/Q=Expr mod x^(n+1) or mod n with degree<p. Example: pade(e^(x),x,5,3) → $(–3*x^2–24*x–60)/(x^3–9*x^2+36*x–60)$
pade	Pade Approximation Syntax: pade(Expr, Var, Integern, Integerp) Returns the Pade approximation of an expression; that is, a rational fraction P/Q such that P/Q=Expr mod x^(n+1) or mod n with degree<p. Example: pade(e^(x),x,5,3) → $(–3*x^2–24*x–60)/(x^3–9*x^2+36*x–60)$
parabola	Syntax: parabola(Point, Line) or parabola(Point, Realk) or parabola(Expr) 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 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. parabola($x-y^2+y-2$) draws the graph of the parabolic equation $x=y^2-y+2$
parallel	Syntax: parallel(Point, Line) Given a point and a line, returns the equation of the line through the point that is parallel to the given line. Examples: parallel(GA,GB) draws the line through point A that is parallel to line B. parallel(point(3,-2),line(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.
parallelogram	Syntax: parallelogram(Point1, Point2, Point3) 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. 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.
parameq	Parametric Syntax: parameq(Obj) Returns a parametric equation for a geometric object. The parametric equation is true for all complex numbers that represent points on the object. Examples: parameq(circle(0,1)) → e^(i*t) parameq(line(i,1-i))

part	Part of Expression
	Syntax:
	part(Expr, Integer)
	Returns the nth sub expression of an expression.
	Examples:
	part(sin(x)+cos(x),1) → sin(x)
	part(sin(x)+cos(x),2) → cos(x)
	part(part(exp(x)*sin(x)+cos(x),1),2) → sin(x)

partfrac	Partial Fraction Decomposition Syntax: partfrac(RatFrac) Performs partial fraction decomposition on a fraction. Example: partfrac($x/(4-x^2)$) → (-1/2)/(x-2)-(1/2)/((x+2)
pcoeff	Roots to Coefficients Syntax: pcoeff(Vector) or pcoeff(List) Given a list or vector containing the roots of a polynomial, returns a vector containing the coefficients (in decreasing order) of the univariate polynomial having those roots. Examples: pcoeff({1,0,0,0,1}) → [1,-2,1,0,0,0] pcoeff([1,0,-2]) → [1,1,-2,0]
perimeterat	Syntax: perimeterat(polygon, point) or

Help Topics Tree 13217	Help Text
	perimeterat(circle, point) Used in Symbolic view of the Geometry app. Displays the perimeter of a polygon or the circumference of a circle. The measure is displayed, with a label, at the given point in Plot view. Example: perimeterat(circle(x^2+y^2=1), point(-4,4)) displays "pcircle(x^2+y^2=1)= 2*π" at point (-4, 4)
perminv	Permutation Inverse Syntax: perminv(permutation) Returns the inverse permutation of permutation. Example: perminv([2,4,3,1]) → [4,1,3,2]
permu2cycles	Syntax: permu2cycles(permutation) Converts permutation to a product of disjoined cycles. Example: permu2cycles([1,3,2,4,6,5]) → [[2,3],[5,6]]
permu2mat	Syntax: permu2mat(permutation) Returns the matrix where the rows of the identity matrix are permuted with permutation. Example: permu2mat([2,3,1]) → [[0,1,0],[0,0,1],[1,0,0]]
permuorder	Permutation Order Syntax: permuorder(permutation) Returns the order of permutation. Example: permuorder([2,4,3,5,1]) → 5
perpen_bisector	Perpendicular Bisector Syntax: perpen_bisector(Segment) or perpen_bisector(Point1, Point2) Draws the perpendicular bisector of a segment. The segment is defined either by its name or by its two endpoints. Examples: perpen_bisector(3+2*i, i) draws the perpendicular bisector of segment C. perpen_bisector(GC) draws the perpendicular bisector of segment AB. perpen_bisector(GA, GB) 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.
PIECEWISE	Piecewise Function Syntax: PIECEWISE(test1, case1, ...[, test8], case8) Used with Home settings Entry set to Algebraic to enter a piecewise-defined function in the Function app Symbolic view (among other uses). Takes as arguments pairs, each of which consists of a condition that defines a sub-function domain and an expression that defines the sub-function. Each of these pairs defines a sub-function of the piecewise function and the domain over which it is active. If used with Home settings Entry set to Textbook or if accessed via the Template menu, then the syntax varies slightly and is restricted to two pieces. Syntax : {case1 if test1 ... {case8 [if test8] Example: PIECEWISE(X<-4,X,-4≤X AND X<2, X+1, X≥2,X+2) as F1(X) in the Symbolic view of the Function app appears as: X if X<-4 F1(X)= X+1 if -4≤X AND X<2 X+2 if X≥2 In Plot view, the graph of two rays and a segment is drawn.
PIXOFF	Pixel Off Syntax: PIXOFF([G], x, y) Sets the color of the pixel of GROB G with coordinates (x, y) to white.
PIXOFF_P	Pixel Off Syntax: PIXOFF_P([G], x, y) Sets the color of the pixel of GROB G with coordinates (x, y) to white.

Help Topics Tree 13217	Help Text
PIXON	Pixel On Syntax: PIXON([G], x, y, [color]) Sets the color of the pixel of GROB G with coordinates (x, y). If supplied, color is a hexadecimal integer of the form aaRRGGBB. This is an RGB color with the Alpha Channel in the high order byte. The Alpha Channel number runs from 0 (opaque) to 255 (transparent). If no color is specified, black is used. Examples: PIXON(0,0,RGB(255,0,0)) PIXON(0,0,RGB(255,0,0,128))
PIXON_P	Pixel On Syntax: PIXON_P([G], x, y, [color]) Sets the color of the pixel of GROB G with coordinates (x, y). If supplied, color is a hexadecimal integer of the form aaRRGGBB. This is an RGB color with the Alpha Channel in the high order byte. The Alpha Channel number runs from 0 (opaque) to 255 (transparent). If no color is specified, black is used. Examples: PIXON_P(50,50,RGB(255,0,0)) PIXON_P(50,50,RGB(255,0,0,128))
plotinequation	Plot Inequation Syntax: plotinequation(Expr,[x=xrange,y=yrange],[xstep],[ystep]) Plots the graph of the solution of inequations with two variables. Example: plotinequation([x+y>3,x^2<y],[x,y],xstep=0.2,ystep=0.2)
plotparam	Plot Parametric Syntax: plotparam(f(Var)+i*g(Var), Var= Interval, [tstep=Value]) Used in the Geometry app Symbolic view. 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. 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)
plotpolar	Plot Polar Syntax: plotpolar(Expr,Var=Interval, [Step]) or plotpolar(Expr, Var, Min, Max, [Step]) Used in the Geometry app to draw a polar graph in Plot view. Examples: plotpolar(sin(2*x),x,0,π,tstep=0.1) plotpolar(f(x),x,a,b) draws the polar curve r=f(x) for x in [a,b]
plotseq	Plot Sequence Syntax: plotseq(f(Var), Var={Start, Xmin, Xmax}, Integern) Used in the Geometry app Symbolic view. 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
point	Syntax: point(Real1, Real2) point(Expr1, Expr2) 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. Examples: point(3,4) creates a point whose coordinates are (3,4). This point may be selected and moved later. point(abscissa(GA), ordinate(GB)) 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.
point2d	point2d Function Syntax: point2d(point1, point2, …, pointn) Randomly re-distributes a set of points such that, for each point, x is in the interval [-5, 5] and y is in the interval [-5, 5]. Any further movement of one of the points will randomly re-distribute all of the points. Example:

Help Topics Tree 13217	Help Text
	point2d(GA,GB,GC,GD)
polar	Syntax: polar(Circle, Point) or polar(Circle, Complex) Returns the polar line of the given point as pole with respect to the given circle. Example: polar(circle(x²+y²=1),point(1/3,0)) → line(x=3)
polar_coordinates	Polar Coordinates Syntax: polar_coordinates(Point) Returns a vector containing the polar coordinates of a point. Example: polar_coordinates(point(1+2*i)) → √5 atan(2)
polar_point	Polar Point Syntax: polar_point(Radius, Angle) Given the radius and angle of a point in polar form, returns the point with rectangular coordinates in complex form. Example: polar_point(2,π/3) → point(2*(1/2+i*√3/2))

pole	Syntax:
	pole(Circle, Line)
	Returns the pole of the given line with respect to the given circle.
	Examples:
	pole(circle(x²+y²=1), line(x=3)) → point(1/3, 0)
	pole(circle(0,1),line((1+i),2))

POLYCOEF	Polynomial coefficients Syntax: POLYCOEF(Vector) or POLYCOEFF(List) Returns the coefficients of the polynomial with the roots specified in a vector or list. Example: POLYCOEF({-1,1}) → [1,0,-1]
POLYEVAL	Polynomial Evaluation Syntax: POLYEVAL(Vector, Value) or POLYEVAL(List, Value) Given a vector or list of coefficients and a value, evaluates the polynomial given by those coefficients at the given value. Example: POLYEVAL({1,0,-1},3) → 8
polyEval	Polynomial Evaluation Syntax: polyEval(Vector, Real) Given a polynomial defined by a vector of coefficients, and a real value n, evaluates the polynomial at that value. Examples: polyEval([1,0,-2],1) → -1 polyEval([1,2,-25,-26,120],8) → 3432
polygonplot	Syntax: polygonplot(Mtrx) 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. Example: polygonplot([[1,2,3],[2,0,1],[-1,2,3]]) draws two figures, each with three points connected by segments.
polygonscatterplot	Syntax: polygonscatterplot(Matrix) 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. Example: polygonscatterplot([[1,2,3],[2,0,1],[-1,2,3]]) draws two figures, each with three points connected by segments.
polynomial_regression	Polynomial Regression Syntax: polynomial_regression(List1, List2, Integer) or polynomial_regression(Matrix, Integer) Given a set of points defined by two lists or a matrix, and a positive integer n, returns a vector containing the coefficients of the nth order polynomial which best approximates the given points.

Help Topics Tree	13217	Help Text
		Example: polynomial_regression([[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]],3) → [0 1 0 0]
POLYROOT		Polynomial roots Syntax: POLYROOT(Poly) or POLYROOT(Vector) Returns the zeros of the polynomial given as argument (either as a symbolic expression or as a vector of coefficients). Example: POLYROOT([1,0,-1]) → {-1,1}
potential		Syntax: potential(Vector1, Vector2) Returns a function whose gradient is the vector field defined by a vector and a vector of variables. Example: potential([2*x*y+3,x²-4*z,-4*y],[x,y,z]) → x²*y+3*x-4*y*z
power_regression		Power Regression Syntax: power_regression(List1, List2) or power_regression(Vector1, Vector2) or power_regression(Matrix) 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. Examples: power_regression({1, 2, 3, 4}, {1, 4, 9, 16}) → [2. 1.] power_regression([[1.0,1.0],[2.0,4.0],[3.0,9.0],[4.0,16.0]]) → [2.,1.]
powerpc		Syntax: powerpc(Circle, Point) 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. Examples: powerpc(circle(0,1+i),3+i) → 8 powerpc(circle(0,point(1+i)),3+i)
prepend		Syntax: prepend(List, Element) or prepend(Vector, Element) Adds an element to the beginning of a list or vector. Example: prepend([1,2],3) → [3,1,2]
primpart		Syntax: primpart(Poly,[Var]) Returns a polynomial divided by the greatest common divisor of its coefficients. Example: primpart(2x²+10x+6) → x²+5*x+3
PRINT		Syntax: PRINT(expr) or PRINT(string) PRINT( ) Prints either the result of expr 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 use the up/down keys to view the text, Backspace to erase the text and any other key to hide the terminal. You can show the terminal at anytime using the ON+T combination (press and hold the On key, press the T key, then release both keys). Pressing On stops the interaction with the terminal. PRINT with no argument clears the terminal.
product		Syntax: product(Expr, [Var], [Min], [Max], [Step]) or product(List) or product(Matrix) 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-by-element product of the matrix. Examples: product(n,n,1,10,2)→ 945 product([[2,3,4],[5,6,7]],[[2,3,4],[5,6,7]])
propfrac		Proper Fraction Syntax: propfrac(Fraction) or

Help Topics Tree	13217	Help Text
		propfrac(RatFrac) Returns a fraction or rational fraction A/B simplified to Q+R/B with R<B (or with the degree of R less than the degree of B). Examples: propfrac(28/12) → 2+1/3 propfrac((x^2+2*x-1)/(x+1)) → x+1-2/(x+1)
ptayl		Syntax: ptayl(Poly, Value, [Var]) Given a polynomial P and a value a, returns the Taylor polynomial Q such as P(x)=Q(x-a) Examples: ptayl(x^2+2*x+1,1) → x^2+4*x+4 ptayl(y^2+2*y+1,1.1,y) → y^2+4.2*y+4.41
purge		Purge Variable Syntax: purge(Var) Unassigns a variable name in CAS view. For example, if f is defined, then purge(f) deletes that definition and returns f to its symbolic state.
q2a		Syntax: q2a(QuadExpr, Vector) Given a quadratic form and a vector of variables, returns the symmetric matrix of the quadratic form with respect to the given variables. Example: q2a(x^2+2*x*y+2*y^2,[x,y]) → [[1,1],[1,2]]
quantile		Syntax: quantile(List, Value) or quantile(Vector, Value) 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. Examples: quantile([0,1,3,4,2,5,6],0.25) → 1 quantile([0,1,3,4,2,5,6],0.75) → 5
quartile1		Syntax: quartile1(List) or quartile1(Vector) or quartile1(Matrix) 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. Examples: quartile1([1,2,3,5,10,4]) → 2 quartile1([[1,2],[5,4],[3,6],[7,8]])
quartile3		Syntax: quartile3(List) or quartile3(Vector) or quartile3(Matrix) 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. Examples: quartile3([1,2,3,5,10,4]) → 5 quartile3([[1,2],[5,4],[3,6],[7,8]])
quartiles		Syntax: quartiles(List) or quartiles(Vector) or quartiles(Matrix) 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. Examples: quartiles([1,2,3,5,10,4]) → [[1],[2],[3],[5],[10]] quartiles([[1,2],[5,4],[3,6],[7,8]])
quorem		Quotient and Remainder Syntax: quorem(Poly1, Poly2) or quorem(Vector1, Vector2) Returns the Euclidean quotient and remainder of the quotient of 2 polynomials, each expressed either in symbolic form directly or as a vector of coefficients. If the polynomials are expressed as vectors of their coefficients, then this command returns a similar vector of the quotient and a vector of the remainder. Examples: quorem(x^3+2*x^2+3*x+4,-x+2) → [-x^2-4*x-11,26] quorem([1,2,3,4],[-1,2]) → [[-1, -4, -11] [26]]
quote		Quote Argument Syntax:

Help Topics Tree	13217	Help Text
		quote(Expr) Returns the argument unevaluated. Example: quote(3+2*x) → 3+2*x You can also use this command to purge a variable. a:=quote(a) purges the variable a

radical_axis		Syntax:
		radical_axis(Circle1, Circle2)
		Returns the line whose points all have the same powerpc values for the two given circles.
		
		Examples:
		radical_axis(circle(((x+2)²+y²)=8),circle(((x-2)²+y²) = 8)) → line(x=0)
		radical_axis(circle(0,point(1+i)),circle(1,point(1+i)))

randbinomial		Random Binomial Syntax: randbinomial(n,p) Returns a random integer with binomial distribution given n trials, each with a probability of success of p. Example: randbinomial(10,.4)
randchisquare		Random χ² Syntax: randchisquare(n) Returns a random number with χ² distribution given n degrees of freedom. Example: randchisquare(5)
randexp		Random Exponential Syntax: randexp(Real) Given a positive real number, returns a random real according to an exponential distribution with real a>0. Example: randexp(2)
randfisher		Random F Syntax: randfisher(n,d) Returns a random number with F distribution given numerator n and denominator d degrees of freedom. Example: randfisher(5,2)
randgeometric		Random Geometric Syntax: randgeometric(p) Returns a random integer with geometric distribution given given probability p. Example: randgeometric(.4)
randnorm		Random Normal Syntax: randnorm(mu, sigma) Returns a random real number from the normal distribution N(mu, sigma). Example: RANDNORM(2,1)
randperm		Random Permutation Syntax: randperm(Integer) Given a positive integer, returns a random permutation of [1,2,…,n]. Example: randperm(4) returns a random permutation of the elements of the vector [1, 2, 3, 4]
randpoisson		Random Poisson Syntax: randpoisson(k) Returns a random integer with Poisson distribution given k. Example: randpoisson(5.4)
randstudent		Random T Syntax: randstudent(n) Returns a random number with Student's t distribution given n degrees of freedom. Example: randstudent(5)

Help Topics Tree 13217	Help Text

randvector	Random Vector
	Syntax:
	randvector(Integern, Integerm)
	randvector(Integern, Interval)
	randvector(Integern, distribution_law)
	Returns a vector of size Integern that contains random integers in the range -99 through 99 (or in 0..Integerm-1) with uniform
	distribution, integers in the given Interval, or contains random numbers according to distribution_law.
	
	Examples:
	randvector(3)
	randvector(3,6)
	randvector(3,normald,0,1)
	randvector(3,poisson,1.2)
	randvector(3,exponentiald,1.2)
	randvector(3,multinomial,[1/2,1/3,1/6])
	randvector(3,multinomial,[1/2,1/3,1/6],[a,b,c])
	randvector(3,'rand(3)')
	randvector(3,1..2)

ranm	Syntax: ranm(Integern, [Integerm],[Interval or Distribution]) Returns a vector of size n or a n*m matrix that contains random integers in the range -99 through 99 with uniform distribution or contains random numbers in a given interval or according to the given Distribution. Example: ranm(3) returns a vector with three elements, each of which is an integer between -100 and 100.

ratnormal	Syntax:
	ratnormal(Expr)
	Rewrites an expression as an irreducible rational fraction.
	Examples:
	ratnormal((x^2-1)/(x^3-1)) → (x+1)/(x²+x+1)
	ratnormal((x^2-1)/(x^3-1)+(x-1)/(x^3-1)+1)

reciprocation	Syntax: reciprocation(Circle, [Obj1, Obj2,...Objn]) 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. 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)]
RECT	Draw Rectangle Syntax: RECT([G], [x1, y1], [x2, y2], [Color]) RECT([G], [x1, y1], [x2, y2], [edgeColor],[fillColor]) Draws a rectangle on G, with diagonal defined by points (x1,y1) and (x2,y2), using edgeColor for the perimeter and fillColor for the inside. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G edgeColor=white fillColor=edgeColor To erase a GROB, execute RECT_P(G). To clear the screen, execute RECT_P(). Note: semi-transparent rectangles can be drawn by using the Alpha channel in the color (0 is opaque, 255 is transparent). The color can also be expressed as { color, alpha }. Examples: Demo_RECT
RECT_P	Rectangle Syntax: RECT_P([G], [x1, y1], [x2, y2], [Color]) RECT_P([G], [x1, y1], [x2, y2], [edgeColor],[fillColor]) Draws a rectangle on G, with diagonal defined by points (x1,y1) and (x2,y2), using edgeColor for the perimeter and fillColor for the inside. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G edgeColor=white fillColor=edgeColor Note: To erase a GROB, execute RECT_P(G). To clear the screen, execute RECT_P(). Note: semi-transparent rectangles can be drawn by using the Alpha channel in the color (0 is opaque, 255 is transparent). The color can also be expressed as { color, alpha }.

Help Topics Tree 13217	Help Text
	Example: Demo_RECT_P
rectangular_coordinates	Rectangular Coordinates Syntax: rectangular_coordinates(Vector) Given a vector containing the polar coordinates of a point, returns a vector containing the rectangular coordinates of the point. Example: rectangular_coordinates(1,π/4) → [1/√2 1/√2]
REDIM	Redimension Syntax: REDIM(matrixname, size) Redimensions the specified matrix 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 zeros.
reduced_conic	Syntax: reduced_conic(Expr,[Vector]) 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 Examples: reduced_conic(x^2+2*x-2*y+1) → [[-1,0],[[0,1],[-1,0]],1,y^2+2*x,[[-1-i*(t*t/-2+i*t),t,-4,4,0.1]]] reduced_conic($(x+y)^2$-2*x+1,x,y)
ref	Gaussian Reduction Syntax: ref(Matrix) Performs Gaussian reduction of a matrix. Examples: ref([[3,1,-2],[3,2,2]]) → [[1,1/3,-2/3],[0,1,4]] ref([[2,1,1,-1],[1,1,2,-1],[1,2,1,-4]]) → [[1,1,2,-1],[0,1,-1,-3],[0,0,1,0.5]]
regroup	Regroup expression Syntax: regroup(expr) Collect terms in an expression. This is equivalent to the auto simplication setting of "Minimum" in the CAS Setup screen. Example: regroup(x+3*x+(5*4/x)) → 4*x+20/x
remove	Syntax: remove(Value, List) or remove(Test, List) 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. Examples: remove(5,[1,2,5,6,7,5]) → [1,2,6,7] remove(x->x>=5,[1,2,6,7]) → [1,2]
reorder	Syntax: reorder(Expr, Vector) Given an expression and a vector of variables, reorders the variables in the expression according to the order given in the vector. Example: reorder(x^2+2*x+y^2,[y,x]) → y^2+x^2+2*x
residue	Syntax: residue(Expr, Var, Value) Returns the residue of an expression at a value. Examples: residue(1/z,z,0) → 1 residue(c/(z*(z-b)),z=b) → c/b
restart	Syntax: restart Purges all CAS variables and resets CAS settings. Example: restart
resultant	Syntax: resultant(Poly1,Poly2,Var) Returns the resultant (the determinant of the Sylvester matrix) of two polynomials. Example: resultant(x^3+x+1,x^2-x-2,x)
RETURN	Return Command

Help Topics Tree 13217	Help Text

	Syntax:
	RETURN expression;
	Exits from a function and returns the value of expression (optional).
	Example:
	EXPORT FACTORIAL(N)
	BEGIN
	  IF N==1 THEN
	    RETURN 1;
	  ELSE
	    RETURN N*FACTORIAL(N-1);
	  END;
	END;
	Example:
	$Demo_RETURN

revlist	Reverse List Syntax: revlist(List) or revlist(Vector) Reverses the order of the elements in a list or vector. Example: revlist([1,2,3]) → [3,2,1]
rhombus	Syntax: rhombus(point1, point2, angle) Used in the Geometry app. 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. Example: rhombus(point(0,0),point(2,2),π/4) draws the rhombus whose first two vertices are given by (0, 0) and (2, 2). The angle at (0, 0) has a measure in radians of π/4.
RIGHT	Right Part Syntax: RIGHT(String, n) Returns the last n characters of the string. Example: RIGHT("MOMOGUMBO",5) → "GUMBO"
romberg	Syntax: romberg(Expr, Var, Val1, Val2) Uses Romberg's method to return the approximate value of a definite integral. Example: romberg(e^(x²),x,0,1) → 1.46265174591
ROTATE	Syntax: ROTATE(String, n) ROTATE(grob, angle, [bg_color]) ROTATE([DestGrob], angle, SrcGrob, [dest_point]) ROTATE([DestGrob], SrcGrob, dest_point_1, dest_point_2, dest_point_3, dest_point_4, [src_point_1, src_point_2, src_point_3, src_point_4]) The string form of ROTATE moves n characters from the beginning or end of String to the opposite end of String, depending on the sign of n. If n is positive, takes the first n characters of String and put them on the right of String. If n is negative, takes the last n characters and put them on the left of String. If ABS(n)>dim(string), returns String. The graphical forms of ROTATE use an angle or sets of points to rotate a graphic object (grob). ROTATE(grob, angle, [bg_color]) Rotate grob around its center by angle. grob will be resized to accommodate the extra space needed and that extra space will be filled by bg_color. If bg_color is not specified, the current background color is used. ROTATE([DestGrob], angle, SrcGrob, [dest_point]) Draw SrcGrob, rotated by angle, on DestGrob with the center of SrcGrob at position dest_point (specified in pixels as a list of 2 numbers or a single complex number). If DestGrob is not specified, G0 is used. If dest_point is not specified, the center of DestGrob is used. ROTATE([DestGrob], SrcGrob, dest_point_1, dest_point_2, dest_point_3, dest_point_4, [src_point_1, src_point_2, src_point_3, src_point_4]) Note: src_points and dest_points are specified in pixels as lists of 2 numbers or as complex numbers. If src_points are not specified, then src_point_1 is set to to the top left corner of SrcGrob, src_point_2 is set to to the top right corner of SrcGrob, src_point_3 is set to to the bottom right corner of SrcGrob, and src_point_4 is set to to the bottom left corner of SrcGrob. Draws the part of SrcGrob specified by the 4 src_points in the area of DestGrob specified by the 4 dest_points.

Help Topics Tree	13217	Help Text
		This is done internally by subdividing the work into 2 triangles (points1, 2 and 3 and points 1, 3 and 4). Therefore non homogenous coordinates can yield to different stretching on both triangles. It is possible to have point_1=point_2 to only work with triangles. Examples: ROTATE("12345",2) → "34512" ROTATE("12345",-1) → "51234" ROTATE("12345",6) → "12345" Demo_ROTATE
row		Syntax: row(Matrix, Integer) or row(Matrix, Interval) 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. Example: row([[1,2,3],[4,5,6],[7,8,9]],2) → [4,5,6]
rowAdd		Row Add Syntax: rowAdd(Matrix, Integer1, Integer2) 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. Example: rowAdd([[1,2],[3,4],[5,6]],1,2) → [[1,2],[4,6],[5,6]]
rowDim		Row Dimension Syntax: rowDim(Matrix) Returns the number of rows of a matrix Examples: rowDim([[1,2,3],[4,5,6]]) → 2 rowDim([[1,2],[3,4],[5,6]]) → 3
rowNorm		Row Norm Syntax: ROWNORM(Matrix) Finds the maximum value (over all rows of the matrix) for the sums of the absolute values of all elements in a row. Examples: ROWNORM([[1,2],[3,-4]]) → 7 ROWNORM([[1,2,3,-4],[-5,3,2,1]]) → 11
rowspace		Row Subspace Syntax: rowspace(matrix,[variable]) Returns a matrix where the rows are a basis of the vector space generated by the rows of matrix. If given, the dimension of this space will be stored into variable. Examples: rowspace([[1,2,3],[1,2,3],[1,2,4],[1,2,5]]) rowspace([[1,2,3],[1,3,6],[2,5,9]],d)
rowSwap		Row Swap Syntax: rowSwap(Matrix,Integer1,Integer2) Given a matrix and two integers, returns the matrix obtained from the given matrix after swapping the two rows indicated by the two integers. Example: rowSwap([[1,2],[3,4],[5,6]],1,2) → [[3,4],[1,2],[5,6]]
rref		Syntax: RREF(Matrix) or RREF(Matrix, [Integer, Option]) Reduced Row-Echelon Form. Changes a rectangular matrix to its reduced row-echelon form. Examples: rref([[2,1,1,-1],[1,1,2,-1],[1,2,1,-4]]) rref([[1,1,0,0,-a1],[0,1,1,0,-a2],[0,0,1,1,-a3],[1,0,0,1,-a4]],keep_pivot)
rsolve		Recurrance Solve Syntax: rsolve(Expr, Var, Condition) or rsolve(List1, List2, List3) 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. Examples: rsolve(u(n+1)=2*u(n)+n,u(n),u(0)=1) → [-n+2*2^n-1]

Help Topics Tree 13217	Help Text
	rsolve([u(n+1)=3*v(n)+u(n),v(n+1)=v(n)+u(n)],[u(n),v(n)],[u(0)=1,v(0)=2])
R→B	Real to Base
	Syntax:
	R→B(Real [, bits [,base]])
	Converts a decimal integer (base 10) to an integer.
	Optionally specify bits and base.
	1 ≤ bits ≤ 64 (Unsigned integer)
	-1 ≥ bits ≥ -63 (Signed integer)
	base = 0 System
	base = 1 Binary
	base = 2 Octal
	base = 3 Decimal
	base = 4 Hexadecimal
	Examples:
	R→B(13) → #Dh (If system base is hexadecimal)
	R→B(1800,64,2) → #3410:64o
	R→B({50,50,50},{64,32,16},{1,2,4}) → {#110010:64b,#62o,#32:16h}
SCALE	Syntax:
	SCALE(matrixname, value, row_number)
	Multiplies the specified row_number of the specified matrix by value.
	Examples:
	SCALE([1,2],3,1) → [3,6]
	SCALE([[1,2],[3,4]],3,2) → [[1,2],[9,12]]
	SCALE([[1,2],[3,4]],{3,2},{2,1}) → {[[1,2],[9,12]],[[2,4],[3,4]]}
SCALEADD	Syntax:
	SCALEADD(matrixname, 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 matrixname.
	Examples:
	SCALEADD([[1,2],[3,4]],3,2,1) → [[10,14],[3,4]]
	SCALEADD([[1,2],[3,4]],{3,2},{2,1},{1,1}) → {[[10,14],[3,4]],[[3,6],[3,4]]}
select	Syntax:
	select(Test, List) or select(Test, Vector)
	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.
	Example:
	select(x->x>=5,[1,2,6,7]) → [6,7]
seq	Sequence
	Syntax:
	seq(Expr, Var=Interval, [Step]) or
	seq(Expr, Integer)
	Given an expression, a variable defined over an interval, and an Optional 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.
	Given an expression and an integer n, returns a list with the expression repeated n times.
	Example:
	seq(2^k,k=0..8) → [1,2,4,8,16,32,64,128,256]
seqsolve	Sequence Solve
	Syntax:
	seqsolve(Expr, Vector, Condition) or
	seqsolve(List1, List2, List3)
	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.
	Examples:
	seqsolve(2x+n,[x,n],1) → -n-1+2*2^n
	seqsolve([x+y,x],[x,y,n],[1,1])
SERIAL	Syntax:
	SERIAL
	Returns the calculator serial number
SETBASE	Set Base
	Syntax:
	SETBASE(#integer[m] [,c])
	Displays integer expressed in base m in whatever base is indicated by c.
	Base marker m can be b (for binary), d (for decimal), o (for octal), d (for decimal), or h (for hexadecimal). If m is omitted, the input is assumed to be in the default base.
	c = 0 System

Help Topics Tree 13217	Help Text

	c = 1 Binary
	c = 2 Octal
	c = 3 Decimal
	c = 4 Hexadecimal
	If c is omitted, the output is displayed in the default base.
	Examples:
	SETBASE (#34o,1) → #11100b
	SETBASE (#1101b) → #Dh (if the default base is hexadecimal)
	SETBASE({#100d,#100d,#100d,#100d,#100d},{0,1,2,3,4}) → {#64h,#1100100b,#144o,#100d,#64h}

SETBITS	Set Bits Syntax: SETBITS(#integer[m] [,bits]) Sets the number of bits to represent integer. The value of bits must be in the range –63 to 64. Base marker m can be b (for binary), d (for decimal), o (for octal), d (for decimal), or h (for hexadecimal). If base marker m or bits is omitted, the default value is used. Examples: SETBITS(#1111b, 15) → #1111:15b SETBITS({#FFFFh,#777o},{15,7}) → {#7FFF:15h,#177:7o}
shift	Syntax: shift(List, Integer) or shift(Vector, [Integer]) or shift(Integer1, Integer2) Given a list or vector and an integer n, shifts the elements of the list or vector n places to the left if n>0 or to the right if n<0. Elements leaving the list from one side are replaced by 0 on the other side. If no integer is provided, the value 1 is used, shifting the elements one place to the left. Given an integer and a second integer n, performs a bitwise shift on the first integer; the shift is left if n>0 and right if n<0. Examples: shift([0,1,2,3],2) → [2,3,0,0] shift([0,1,2,3,4]) → [0,0,1,2,3]
shift_phase	Shift Phase Syntax: shift_phase(Expr) Returns the result of applying a phase shift of π/2 to a trigonometric expression. Example: shift_phase(sin(x)) → -cos((π+2*x)/2) with CAS setting Simplify set to None
Si	Sine Integral Syntax: Si(Expr) Returns the sine integral of an expression, int(sin(t)/t,t=0..x) Example: Si(1.0) → 0.946083070367
signature	Syntax: signature(Vector) Returns the signature of a permutation given as a vector. Example: signature([2,1,4,5,3]) → -1
simplex_reduce	Simplex Reduction Syntax: simplex_reduce(Matrix_A, Vector_B, Vector_C) Reduction by simplex algorithm to find max(c.x) under A.x<=b and x>=0, b>=0. Returns the maximum, the augmented solution x and the reduced matrix. Accepts also [[A|I|b],[-c|*|0]] as argument. Examples: simplex_reduce([[3,2,2],[1,1,1]],[3,4],[1,2,3]) simplex_reduce([[2,1,1,1,0,0,2],[1,2,3,0,1,0,5],[2,2,1,0,0,1,6],[-3,-1,-3,1,-1,2,0]])
simult	Syntax: simult(Matrix1, Matrix2) 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. Example: simult([[3,1],[3,2]],[[-2],[2]]) returns [[-2],[4]] Examples: simult([[3,1],[3,2]],[[-2],[2]]) → [[-2],[4]] simult([[3,1],[3,2]],[[-2,1],[2,-1]]) → [[-2, 1],[4,-2]]
SIN	Sine Syntax:

Help Topics Tree 13217	Help Text
	SIN(Value) Returns the sine of Value. Value is interpreted as radians, degrees or gradians, depending on the setting of Angle Measure in Home Settings, CAS Settings, or Symbolic Setup. Example: SIN(30) → 0.5 (Degrees mode) SIN(1+i) → 1.29845758142+0.634963914785*i SIN({30,90}) → {0.5,1} (Degrees mode) SIN((π/6)_rad) → 0.5
sincos	Syntax: sincos(Expr) Returns an expression with the complex exponentials rewritten in terms of sine and cosine. Example: sincos(exp(-i*x)) → cos(x)-i*sin(x)
single_inter	Single Intersection Syntax: single_inter(Curve1, Curve2, [Point]) Returns the intersection of Curve1 and Curve2 that is closest to Point. In Plot view, this command will prompt for two curves. After that, a point will appear; move this point to the intersection desired and press Enter. You can move the point later to change intersections if you wish. Example: single_inter(line(y=x),circle($x^2+y^2=1$), point(1,1)) → point((1+i)*√2/2)
SIZE	Size of a matrix Syntax: SIZE(matrix) Size of a list or matrix. Returns the size of a list or the dimensions of matrix as a list: {Rows, Columns}. Examples: SIZE({1,2,3,4,5}) → 5 SIZE([[1,2,3], [4,5,6]]) → [2 3]
slope	Syntax: slope(Line) or slope(Point1, Point2) Given a line or two points that define a line, returns the slope of the line. Example: slope(line(1,2*i)) → -2
slopeat	Syntax: slopeat(Segment, Point) or slopeat(Line, Point) or slopeat(Ray, Point) Used in Symbolic view of the Geometry app. Displays the slope of a straight object (segment, line, etc.). The measure is displayed, with a label, at the given point in Plot view. Example: In Symbolic view, slopeat(line(point(0,0), point(2,3)), point(-8,8)) displays “sline(point(0,0), point(2,3))=3/2” at point (–8, 8) in Plot view. Example: slopeat(line(point(0,1),point(3,2)),point(-10,4))
sort	Syntax: sort(List) or sort(Obj1, Obj2, ...) Sorts a list of a sequence of objects. If the list or sequence contains numbers, then sorting is done in increasing order. If the list or sequence contains expressions, then the sorting is done in increasing order of numerical values, sums, and products, in increasing exponential order. Examples: sort([3,2,2,4,1,0]) → [0,1,2,2,3,4] sort(x,3*x,4*x^2,5,7,x^2+1) → [5 7 x x^2+1 3*x 4*x^2]
spline	Syntax: spline(ListX, ListY, Var, Integer) or spline(VectorX, VectorY, Var, Integer) 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. Example: spline([0,1,2],[1,3,0],x,3) → [(-5/4)*x^3+(13/4)*x+1, (5/4)*(x-1)^3/4-(15/4)*(x-1)^2-(1/2)(x-1)+3]
sqrfree	Syntax: sqrfree(Expr) Returns a polynomial factorized as a product of powers of coprime factors where each factor has roots of multiplicity 1 Examples: sqrfree(x^4-2*x^2+1) → $(x^2-1)^2$ sqrfree((x-2)^7*(x+2)^7*(x^4-2*x^2+1)) → $(x^2-1)^2$*$(x^2-4)^7$

Help Topics Tree 13217	Help Text
sqrt	Square Root Syntax: sqrt(Expr) Returns the square root of Expr Example: sqrt(50) → 5*√2 (7.07106781178)
STARTVIEW	Start View Syntax: STARTVIEW(ViewNumber[,Redraw]) Starts a view of the current app. Redraw, is optional; if Redraw, is true (non 0), it will force a refresh for the view. The view numbers are as follows: 0=Symbolic 1=Plot 2=Numeric 3=Symbolic Setup 4=Plot Setup 5=Numeric Setup 6=App Info 7=Views key If the current app has views defined under the Views menu, then the following view numbers are used: 8=First special view (Split Screen Plot Detail) 9=Second special view (Split Screen Plot Table) 10=Third special view (Autoscale) 11=Fourth special view (Decimal) 12=Fifth special view (Integer) 13=Sixth special view (Trig) If ViewNumber is negative, the following global views are used: -1=Home Screen -2=Modes -3=Memory Manager -4=App Library -5=Matrix Catalog -6=List Catalog -7=Program Catalog -8=Note Catalog Example: STARTVIEW(-3)
stddev	Sample Standard Deviation Syntax: stddev(List1, [List2]) or stddev(Vector1, [Vector2]) or stddev(Matrix) Returns the standard deviation of the elements of a list or vector, or a list of the standard deviations of the columns of a matrix. The optional second list is a list of weights. Examples: stddev([[1,2,3],[5,6,7]]) → [2,2,2] stddev([1,2,3]) → (√6)/3
stddevp	Population Standard Deviation Syntax: stddevp(List1, [List2]) or stddvp(Vector 1, Vector2) or stddvp(Matrix) Returns the population standard deviation of the elements of a list or vector, or a list of the population standard deviations of the columns of a matrix. The optional second list is a list of weights. Examples: stddevp([1,2,3]) → 1 stddevp([[1,2,3],[5,6,7]])
sto	Store Syntax: sto(Obj, Var) Stores the object given as first argument in the variable given as second argument. Example: sto("hello",b)
STRING	Syntax: STRING(Expression, [Mode], [Precision], [Separator or {Separator, ["[DecimalPoint[Exponent[NegativeSign]]]"], [DotZero]}], [SizeLimit or {SizeLimit, [FontSize], [Bold], [Italic], [Monospaced]}])

Help Topics Tree	13217	Help Text
		Evaluates Expression and returns the result as a string. The Mode, Precision, and Separator parameters specify how numbers are displayed. If Mode is specified, it is: 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 can be a number. -1 means use default, 0 to 10 correspond to the 11 built-in digit grouping choices available in home settings. OR Separator can be a string containing a set of digits and separators. The last digit is assumed to be the one just before the decimal point. "[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 between -1 and 1 are displayed without a leading zero (for example, .1 instead of 0.1) If SizeLimit is specified, the command will attempt to generate a string that fits in the given number of pixels. FontSize is used along with Bold, Italic, and Monospaced (if their value is non-zero) to estimate the maximum string length that will fit. The values for FontSize are: 0=current font (default) 1=font 10 2=font 12 (Small) 3=font 14 (Medium) 4=font 16 (Large) 5=font 18 6=font 20 7=font 22 Examples: Current number format setting Standard: STRING(3*π) → "9.42477796077" Number format Fixed, 4 decimal places: STRING(3*π,2,4) → "9.4248"
STRINGFROMID		String From Identifier Syntax: STRINGFROMID(Integer) Returns the built-in string associated with the ID of the current language. Example: STRINGFROMID(1)
sturm		Syntax: sturm(Poly,[Var, Complexa, Complexb]) Returns the Sturm sequence corresponding to a polynomial or the number of sign changes of this polynomial for the variable in the interval (a,b]. Examples: sturm(x^3-1,x) → [1,[[1,0,0,-1],[3,0,0],9],1] sturm(x^3-1,x,-2-i,5+3i) → 3
sturmseq		Syntax: sturmseq(Poly,[Var]) or sturmseq(RatFrac, [Var]) Returns the Sturm sequence corresponding to a polynomial or to a rational fraction. Examples: sturmseq(x^3-1,x) → [1,[[1,0,0,-1],[3,0,0],9],1] sturmseq((x^5-x^3)/(x+2),x)
SUB		Extract Portion Syntax: SUB(object, start, end) Extracts a portion, of a list or matrix. For a matrix, start and end are two lists of two numbers ({row, col}) specifying the top left and bottom right of the portion to extract. For a vector or list, start and end are two numbers specifying the indexes of the first and last objects of the portion to extract. Examples: SUB([[1,2,1],[2,1,3],[4,2,3]],{2,1},{3,2}) → [[2,1],[4,2]] SUB({5,2,9,4},2,3) → {2,9}
SUBGROB		Copy GROB to Target Syntax:

Help Topics Tree 13217	Help Text
	SUBGROB(srcG, [x1, y1], [x2, y2], trgtG) Sets graphic trgtG to be a copy of the area of srcG between points (x1,y1) and (x2,y2). If both (x1, y1) and (x2, y2) are not specified, then the entire graphic srcG is used. If (x1, y1) is not specified, then the top left corner of srcG is used; if (x2, y2) is not specified, then the bottom right corner of srcG is used. trgtGRB can be any of the graphic variables except G0. SUBGROB(G1, G4) will copy G1 in G4. Example: Demo_SUBGROB
SUBGROB_P	Copy GROB to Target Syntax: SUBGROB_P(srcG, [x1, y1], [x2, y2], trgtG) Sets graphic trgtG to be a copy of the area of srcG between points (x1,y1) and (x2,y2). If both (x1, y1) and (x2, y2) are not specified, then the entire graphic srcG is used. If (x1, y1) is not specified, then the top left corner of srcG is used; if (x2, y2) is not specified, then the bottom right corner of srcG is used. trgtGRB can be any of the graphic variables except G0. SUBGROB_P(G1, G4) will copy G1 in G4. Example: Demo_SUBGROB_P
subMat	Sub Matrix Syntax: subMat(Matrix, Int1, Int2, Int3, Int4) 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. Example: subMat([[1,2],[3,4],[5,6]],1,1,2,1) → [[1],[3]]
suppress	Syntax: suppress(List, Integer) Given a list and a counting number n, deletes the nth element in the list and returns the result. Example: suppress([0,1,2,3],2) → [0,2,3]
surd	Syntax: surd(Expr, Integer) Given an expression and an integer n, returns the expression raised to the power 1/n. Example: surd(-8,3) → -2
SWAPCOL	Swap Columns Syntax: SWAPCOL(matrixname, column1, column2) Exchanges column1 and column2 in the specified matrix matrixname. Examples: SWAPCOL([[1,2,1],[2,1,3],[4,2,3]],2,3) → [[1,1,2],[2,3,1],[4,3,2]] SWAPCOL([[1,2,1],[2,1,3],[4,2,3]],{1,2},{3,3}) → {[[1,2,1],[3,1,2],[3,2,4]],[[1,1,2],[2,3,1],[4,3,2]]} SWAPCOL({[[1,2,1],[2,1,3],[4,2,3]],[[9,8,7],[9,8,7]]},{1,2},{3,3}) → {[[1,2,1],[3,1,2],[3,2,4]],[[9,7,8],[9,7,8]]}
SWAPROW	Swap Rows Syntax: SWAPROW(matrixname, row1, row2) Exchanges row1 and row2 in the specified matrix matrixname. Examples: SWAPROW([[1,2,1],[2,1,3],[4,2,3]],2,3) -→ [[1,2,1],[4,2,3],[2,1,3]] SWAPROW([[1,2,1],[2,1,3],[4,2,3]],{1,2},{3,3}) → {[[4,2,3],[2,1,3],[1,2,1]],[[1,2,1],[4,2,3],[2,1,3]]} SWAPROW({[[1,2,1],[2,1,3],[4,2,3]],[[9,9],[6,6],[5,5],[8,8]]},{1,2},{3,3}) → {[[4,2,3],[2,1,3],[1,2,1]],[[9,9],[5,5],[6,6],[8,8]]}
sylvester	Sylvester Matrix Syntax: sylvester(Poly,Poly,Var) Returns the Sylvester matrix of two polynomials Examples: sylvester(x^2-1,x^3-1,x) → [[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]] sylvester(x^3-p*x+q,3*x^2-p,x)
table	CAS Table Syntax: table(SeqEqual(index=value)) Defines an array where the index are strings or real numbers Example: table(3=-10,"a"=10,"b"=20,"c"=30,"d"=40)

Help Topics Tree			13217	Help Text

		tabvar		Table of Variation
				Syntax:
				tabvar(function,[variable, minimum, maximum])
				Creates a table of variation for function. If variable is not given, a default variable is assumed. If minimum or maximum is not specified, they are also assumed.
				Examples:
				tabvar(sin(x))
				tabvar(x^2+x+1,x)
				tabvar(a^2,a,-3,5)

		tail		Syntax: tail(Vector) or tail(List) or tail(String) or tail(Obj1, Obj2, ...) Given a vector, list, string, or sequence of objects, returns a vector with the first element deleted. Examples: tail([3,2,4,1,0]) → [2,4,1,0] tail("bonjour") → "onjour"
		tan2cossin2		Syntax: tan2cossin2(Expr) Returns an expression with tan(x) rewritten as (1–cos(2*x))/sin(2*x). Example: tan2cossin2(tan(x/2)) → (1-cos(x))/sin(x)
		tan2sincos2		Syntax: tan2sincos2(Expr) Returns an expression with tan(x) rewritten as sin(2*x)/(1+cos(2*x)). Example: tan2sincos2(tan(x/2)) → sin(x)/(1+cos(x))
		tcoeff		Syntax: tcoeff(Poly) tcoeff(Vector) tcoeff(List) Returns the coefficient of the term of lowest degree of a polynomial. The polynomial can be expressed in symbolic form or as a vector or list of coefficients. Examples: tcoeff(1-2*x^3+x^2+7*x) → 7 tcoeff([-2,1,7,0]) → 7
		TEVAL		Time Evaluation Syntax: TEVAL(Param) Returns the time it takes to evaluate the parameter. Example: TEVAL(WAIT(5)) → ~5.095_s Note: actual result will vary but should be close to 5.00_s
		TEXTOUT		Draw Text Syntax: TEXTOUT(text, [G], x, y, [font], [textColor], [width], [backgroundColor]) Draws text on graphic G at position (x, y) using font and textColor. Paints the background before drawing the text using color backgroundColor. If width is specified, does not draw text more than width pixels wide. If backgroundColor is not specified, the background is not erased. The sizes for font are: 0=current font (default) 1=font 10 2=font 12 (Small) 3=font 14 (Medium) 4=font 16 (Large) 5=font 18 6=font 20 7=font 22 Returns the X (in pixels, not Cartesian) coordinate at which the next character of the string should be drawn if the string had more characters Examples: TEXTOUT("Hello HP Prime",-5,0,4,RGB(128,0,128),200,RGB(255,255,0)); FREEZE Demo_PISERIES
		TEXTOUT_P		Draw Text Syntax: TEXTOUT_P(text, [G], x, y, [font], [textColor], [width], [backgroundColor])

Help Topics Tree	13217	Help Text
		Draws text on graphic G at position (x, y) using font and textColor. Paints the background before drawing the text using color backgroundColor. If width is specified, does not draw text more than width pixels wide. If backgroundColor is not specified, the background is not erased. The sizes for font are: 0=current font (default) 1=font 10 2=font 12 (Small) 3=font 14 (Medium) 4=font 16 (Large) 5=font 18 6=font 20 7=font 22 Returns the X coordinate at which the next character of the string should be drawn if the string had more characters Examples: TEXTOUT_P("Hello HP Prime",100,100,4,RGB(255,0,0),200,RGB(0,255,255)); FREEZE Demo_PISERIES_P
TICKS		Internal Ticks Value Syntax: TICKS() Returns the internal millisecond clock value. Example: TICKS
transpose		Transpose Matrix Syntax: transpose(Matrix) Transposes a matrix (without conjugation). Examples: transpose([[1,2,3],[1,3,6],[2,5,7]]) → [[1,1,2],[2,3,5],[3,6,7]] transpose(conj([[1+i,2,3],[1,3,6],[2,5,9-i]]))
TRIANGLE_P		Draw Triangle Syntax: TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha]) TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3]) TRIANGLE_P([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"]) TRIANGLE_P([G], points_definition, triangle_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) TRIANGLE_P([G], pre_rotated_points, triangle_definitions, [zstring]) TRIANGLE_P([G]) The basic form of TRIANGLE_P draws one triangle between specified pixel coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If 3 colors are specified, blends the colors in between the vertexes. The advanced form of TRIANGLE_P allows the rendering of multiple triangles at a 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 2 to 4 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)… triangle_definitions is either a list or a matrix of triangle definition. Each triangle is defined by 3 to 5 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 re-specifying the same color for each triangle. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the point using usual 3/4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed 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 using 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 provided easier z clipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). TRIANGLE_P returns a string which contains all the transformed points. If you plan to call TRIANGLE_P or LINE_P multiple times in a row using the same points and transformation, you can do so by replacing the points_definition by this string and omitting the transformation definition in subsequent calls to TRIANGLE_P and LINE_P. About zstring

Help Topics Tree 13217	Help Text
	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 TRIANGLE_P with appropriate z (0-255) values for each of the triangle vertexes and TRIANGLE_P will not draw pixels further than the already drawn pixels. zstring is automatically updated as appropriate. Examples: TRIANGLE_P(0,20,150,50,100,100,#FFh,#FF00h,#FF0000h,128); FREEZE Demo_TRIANGLE_P Demo_Tetrahedron_P
trunc	Syntax: TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha]) TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3]) TRIANGLE_P([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"]) TRIANGLE_P([G], points_definition, triangle_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) TRIANGLE_P([G], pre_rotated_points, triangle_definitions, [zstring]) TRIANGLE_P([G]) The basic form of TRIANGLE_P draws one triangle between specified pixel coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If 3 colors are specified, blends the colors in between the vertexes. The advanced form of TRIANGLE_P allows the rendering of multiple triangles at a 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 2 to 4 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)… triangle_definitions is either a list or a matrix of triangle definition. Each triangle is defined by 3 to 5 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 re-specifying the same color for each triangle. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the point using usual 3/4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed 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 using 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 provided easier z clipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). TRIANGLE_P returns a string which contains all the transformed points. If you plan to call TRIANGLE_P or LINE_P multiple times in a row using the same points and transformation, you can do so by replacing the points_definition by this string and omitting the transformation definition in subsequent calls to TRIANGLE_P and LINE_P. 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 TRIANGLE_P with appropriate z (0-255) values for each of the triangle vertexes and TRIANGLE_P will not draw pixels further than the already drawn pixels. zstring is automatically updated as appropriate. Examples: TRIANGLE_P(0,20,150,50,100,100,#FFh,#FF00h,#FF0000h,128); FREEZE Demo_TRIANGLE_P Demo_Tetrahedron_P
truncate	Syntax: truncate(Poly, Integer) Given a polynomial and an integer n, truncates the polynomial at order n. Example: truncate($(x^2+x)^2$,3) → $2*x^3+x^2$
tsimplify	Transcendental Simplify Syntax: tsimplify(Expr) Returns an expression with transcendental rewritten as complex exponentials Example: tsimplify(e^(2*x)+e^(x)) → $e^{(x)^2}$+e^(x)
TYPE	Object Type Syntax: TYPE(object)

Help Topics Tree	13217	Help Text
		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
	type	Syntax: type(Object) Returns the type of an object. Examples: type("abc") → DOM_STRING type([1,2,3]) → DOM_LIST
	QPI	Real to Quotient Approximation Syntax: QPI(expr,[digits]) QPI attempts to approximate expr into one of the following forms using digits of precision: p/q, (a/b)*√(p/q), (p/q)*π, ln(p/q) or e^(p/q) expr may be a number, complex, list, vector or matrix. Examples: QPI(1.23) → 123/100 QPI(2.2360679775) → √(5) QPI(1.46633706879) → LN((13/3)) QPI(4.71238898038) → (3/2)*π QPI({-0.714285714286,-1.41421356237,-0.405465108108,1.11751906874,-2.44346095279,0.657047293577}) → {-(5/7),-√(2),LN((2/3)),e^(1/9),-((7/9)*π),(5/7)*√(11/13)}
	▶	Store Syntax: value ▶ variable Stores value in variable. Example: 3▶A stores the value 3 in the variable A.
U-Z		Function Catalog U-Z Toolbox function catalog U-Z
	UFACTOR	Unit factor conversion Syntax: UFACTOR(Value_Unit1, 1_Unit2) Converts a measurement using a compound unit into a measurement expressed in constituent units. 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. Using UFACTOR, you can express a measurement in Coulombs as a product of Amperes and time. Examples: UFACTOR(100_C,1_A) → 100_A*s UFACTOR(100_C,1_min) → 1.66666666667_min*A
	unapply	Syntax: unapply(Expr,Var) Returns the function defined by an expression and a variable. Example: unapply(2*x^2,x) → (x)->2*x^2
	uniform	Discrete Uniform Syntax: uniform(a,b,x) Uniform probability density function Computes the probability density of the uniform distribution at x given parameters a and b. Example: uniform(1.2,3.5,3) → 0.434782608696
	uniform_cdf	Cumulative Uniform Syntax: uniform_cdf(a,b,x,[x2]) Cumulative uniform distribution function Returns the lower-tail probability of the uniform probability density function for the value x, given parameters a and b. Examples: uniform_cdf(1.2,3.5,3) → 0.782608695652

Help Topics Tree 13217	Help Text
	uniform_cdf(1.2,3.5,2,3) → 0.434782608696
uniform_icdf	Inverse Cumulative Uniform Syntax: uniform_icdf(a,b,p) Inverse cumulative uniform distribution function Returns the value x such that the uniform lower-tail probability of x, given parameters a and b, is p. Example: uniform_icdf(3.2,5.7,0.48) → 4.4
UPPER	Uppercase Syntax: UPPER(string) Returns string with lowercase characters converted to uppercase. Examples: UPPER("abc") → "ABC" UPPER("αβγ") → "ΑΒΓ"
USIMPLIFY	Unit Simplification Syntax: USIMPLIFY(Value_Unitsexpr) Simplifies Value in a complex unit expression Unitsexpr to an equivalent value in a simpler unit expression. Example: a Joule is defined as 1 kg*m^2/s^2. USIMPLIFY(5_kg*1_m^2/1_s^2) → 5_J
UTPC	Upper-Tail Chi-Square Probability Syntax: UTPC(Degrees, Value) Upper-Tail Chi-Squared probability distribution function. Returns the Upper-Tail Chi-Squared probability, given degrees of freedom, evaluated at the given value. Returns the probability that a Chi-Squared random variable is greater than the given value. Example: UTPC(5,1.1) → 0.954103676028
UTPF	Upper-Tail F Probability Syntax: UTPF(Numerator, Denominator, Value) Upper-Tail Snedecor's F Probability distribution function. Returns the Upper-Tail Snedecor's F probability, given Numerator degrees of freedom and Denominator degrees of freedom, evaluated at the given Value. Returns the probability that a Snedecor's F random variable is greater than the given value. Example: UTPF(3,2,1.1) → 0.508688301183
UTPN	Upper-Tail Normal Probability Syntax: UTPN(Mean, Variance, Value) Upper-Tail Normal Probability distribution function. Returns the Upper-Tail Normal probability, given Mean and Variance, evaluated at Value. Returns the probability that a normal random variable is greater than the given value. Note: The variance is the square of the standard deviation. Example: UTPN(0,1,1.1) → 0.135666060946
UTPT	Upper-Tail t Probability Syntax: UTPT(Degrees, Value) Upper-Tail Student's t probability distribution function. Returns the Upper-Tail Students t probability, given degrees of freedom, evaluated at Value. Returns the probability that the Student's t random variable is greater than the given value. Example: UTPT(5,1) → 0.181608733825
valuation	Syntax: valuation(Poly,[Var]) 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. Examples: valuation(x^4+x^3) → 3 valuation([5,0,0,3,0,0])
variance	Syntax: variance(List1, [List2]) or variance(Matrix) 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. Examples: variance([3,4,2]) → 2/3 variance([[1,2,3],[5,6,7]])

Help Topics Tree 13217	Help Text
vector	Syntax: vector(Point1, [Point2]) Given one point, defines a vector from the origin to the given point. With two points, defines the vector from the first point to the second. Example: vector(point(1,1),point(3,0)) creates a vector from (1,1) to (3,0).
VERSION	Syntax: VERSION([n]) Returns a string that contains the version numbers for the various components of the system. This is equivalent to the About Prime help page If given integer n, returns that specific part of the version string. Examples: VERSION VERSION(1) VERSION(5)
vertices	Syntax: vertices(Polygon) Returns a vector containing the list of the vertices of a polygon. Examples: vertices(isoceles_triangle(0,1,π/4))[2] vertices(isopolygon(0,1,4) → [point(0,0) point(1,0) point(1,1) point(0,1)]
vertices_abca	Closed Vertices Syntax: vertices_abca(Polygon) Returns a vector containing the closed list of the vertices of a polygon. Example: vertices_abca(isopolygon(0,1,4) → [point(0,0) point(1,0) point(1,1) point(0,1) point(0,0)]
vpotential	Syntax: vpotential(Vector1, Vector2) Given a vector V and a vector of variables, returns the vector U such that curl(U)=V. Example: vpotential([2*x*y+3,x²-4*z,-2*y*z],[x,y,z]) → [0,-2*x*y*z,(-1/3)*x³+4*x*z+3*y]
WAIT	Syntax: WAIT(n) Pauses program execution. If n ≥ 1 : Execution paused for the specified number (n) seconds. Returns the value of n. If n = 0 or omitted : Execution paused until a key is pressed. If a key is pressed, the key code is returned. After a 1-minute timeout, returns -1 If n = -1 : Execution paused until a key is pressed or there is a mouse event. If a key is pressed, the key code is returned. If a mouse event happens, a list of the form { type, [x, y], [dx, dy] } is returned. Normally x/y is the event position unless otherwise indicated. After a 1-minute timeout, returns -1 Event type can be: 0: Mouse Down 1: Mouse Move 2: Mouse Up (x/y is not provided) 3: Mouse Click (if a click is detected, there is no Mouse Up) 5: Mouse Stretch. x/y is the delta since the last event. dx/dy is the delta since the original mouse down. 6: Mouse Rotate, x is original angle, y is new angle in 32nd of a circle. 7: Mouse Long Click, indicates the mouse stayed down for 1 second. Example: WAIT(3)
weibull	Discrete Weibull Syntax: weibull(k,n,[t],x) Weibull probability density function Computes the probability density of the Weibull distribution at x given parameters k, n and t. By default, t is 0. Examples: weibull(2.1,1.2,1.3) → 0.58544681204 weibull(2.1,1.2, 0,1.3) → 0.58544681204
weibull_cdf	Cumulative Weibull Syntax:

Help Topics Tree 13217	Help Text
	weibull_cdf(k,n,[t],x,[x2]) Cumulative Weibull distribution function Returns the lower-tail probability of the Weibull probability density function for the value x, given parameters k, n and t. By default, t is 0. Examples: weibull_cdf(2.1,1.2,1.9) → 0.927548261801 weibull_cdf(2.1,1.2,0,1.9) → 0.927548261801 weibull_cdf(2.1,1.2,1,1.9) → 0.421055367782
weibull_icdf	Inverse Cumulative Cauchy Syntax: weibull_icdf(k,n,[t],x) Inverse cumulative Weibull distribution function Returns the value x such that the Weibull lower-tail probability of x, given parameters k , n and t Examples: weibull_icdf(4.2,1.3,.95) → 1.68809330364 weibull_icdf(4.2,1.3,0,.95) → 1.68809330364
when	When Conditional Syntax: when(Cond,Expr1,Expr2) If condition is true, returns Expr1; otherwise, returns Expr2. Example: when(n,1,0)
WHILE	While Loop Structure Syntax: WHILE test DO commands END; Executes commands WHILE test is true. Example: A:=5; WHILE A>0 DO PRINT(A); A:= A-1; END; will print 5 4 3 2 1 Examples: Demo_WHILE ISPERFECT PERFECTNUMS
wilcoxonp	Wilcoxon Distribution Syntax: wilcoxonp(Integer1,[Integer2]) Distribution of the Wilcoxon or Mann-Whitney test for one or two samples. Examples: wilcoxonp(4) wilcoxonp(7,5)

wilcoxons	Wilcoxon statistic
	Syntax:
	wilcoxons(List1,Median)
	wilcoxons(List1,List2)
	Rank statistic of Wilcoxon or Mann-Whitney test for 1 sample (List1) and Median, or 2 samples (List1,List2).
	Examples:
	wilcoxons([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , [2, 6, 10, 11, 13, 14, 15, 18, 19, 20])
	wilcoxons([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , 10)

wilcoxont	Wilcoxon test
	Syntax:
	wilcoxons(List1,Median, [Method],[Significance])
	wilcoxons(List1,List2)
	wilcoxont(List,List || Real,[Func],[Real])
	Wilcoxon or Mann-Whitney test for 1 sample (List1) and Median, or 2 samples (List1,List2). Optionally, specify Method to be '<' or '>', and Significance.
	Examples:
	wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , [2, 6, 10, 11, 13, 14, 15, 18, 19, 20])
	wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , [2, 6, 10, 11, 13, 14, 15, 18, 19, 20],0.01)
	wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , 10,'>')
	wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , 10,'>',0.05)

XOR	Logical XOR Syntax: Value1 XOR Value2 For Real numbers, returns 1 if either Value1 or Value2 is non-zero but not both; otherwise, returns 0.

Help Topics Tree		13217	Help Text
			For Integers and Strings, XOR is performed bitwise, returning 1 if exactly one bit is 1 and the corresponding bit is 0, otherwise 0. Examples: 3 XOR 2 → 0 0 XOR 2 → 1 0 XOR 0 → 0 {3,0,0} XOR {2,1,0} → {0,1,0} 3_inch==7.62_cm XOR 9_(inch²)==58.0644_(cm²) → 0 #CC44h XOR #44CCh → #8888h "C" XOR "b" → "!" X:=0; 0 XOR (X:=7); 1 XOR (X:=9); X → 9 7 ≤ 3 XOR 5 < 9 XOR 3 ≠ 2.9 + 0.2 → 0
		zip	Syntax: zip('Function', List1, List2, Default) or zip('Function', Vector1, Vector2, Default) 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. Examples: zip('+',[a,b,c,d], [1,2,3,4]) → [a+1,b+2,c+3,d+4] zip(sum,[a,b,c,d], [1,2,3,4]) → [a+1,b+2,c+3,d+4]
		ztrans	Z Transform Syntax: ztrans(Expr,[Var],[ZtransVar]) Returns the Z transform of a sequence. Examples: ztrans(a^x) → -x/(a-x) ztrans(a^n,n,z) → -z/(a-z)
	Other		Function Catalog - Other Toolbox function catalog - Other
		%CHANGE	Percent Change Syntax: %CHANGE(x, y) Percent change from x to y. Returns 100*(y-x)/x. Examples: %CHANGE(20,50) → 150 %CHANGE(4.5,8.3) → 84.4444444444 %CHANGE({10,20,30},{75,75,75}) → {650,275,150}
		%TOTAL	Percent Total Syntax: %TOTAL(x, y) Percent total; the percentage of x that is y. Returns 100*y/x. Examples: %TOTAL(20,50) → 250 %TOTAL(1.5,7.5) → 500 %TOTAL({10,20,30},{75,75,75}) → {750,375,250}
		.*	Syntax: List1 .* List2 or Matrix1 .* Matrix2 Performs an element-by-element multiplication of 2 lists or 2 matrices. Examples: [[1,2],[3,4]] .* [[3,4],[5,6]] → [[3,8],[15,24]] [[1+2*i,3+2*i],[1+2*i,2+i]] .* [[1,2],[3,4]] → [[1+2*i,6+4*i],[3+6*i,8+4*i]] {1,2,3} .* {4,5,6} → {4,10,18}
		.+	Syntax: Matrix .+ Value or Value <Space> .+ Matrix or List .+ Value or Value <Space> .+ List Adds a value (real or complex) to each element of a list or matrix. Note that when the value precedes the operator, you must put a space between the value and the operator; otherwise, the point in the operator is read as a (possible extraneous) decimal point in the value. Examples: [1,2] .+ 3 → [4,5] 2.5 .+ [[1,2],[3,4]] → [[3.5,4.5],[5.5,6.5]] [[1,2],[3,4]] .+ (2+5i) → [[3+5*i,4+5*i],[5+5*i,6+5*i]] 2_slug .+ [[1,2],[3,4]]_slug → [[3,4],[5,6]]_slug {[[6,6],[3,1]],[[1,2],[8,7]]} .+ 3 → {[[9,9],[6,4]],[[4,5],[11,10]]}

Help Topics Tree 13217	Help Text
.-	Syntax: Matrix .- Value or Value <Space> .- Matrix or List .- Value or Value <Space> .- List Subtracts a value (real or complex) from each element of a list or vector. When the value precedes the operator, then each element in the list or matrix is subtracted from the value. In this latter case, you must put a space between the value and the operator; otherwise, the point in the operator is read as a (possible extraneous) decimal point in the value. Examples: 3 - 2 → 1 0.5 - 3.7 → -3.2 (2+5i) - (3+2i) → 1+3i [[1,2],[3,4]] - [[0.5,0.6],[1.5,4.1]] → [[0.5,1.4],[1.5,−0.1]] [[1,2],[3,4]] - {1,2,3} → {[[0,1],[2,3]],[[−1,0],[1,2]],[[−2,−1],[0,1]]} {1_l,20_cm,3_kg}-{4_ozfl,5_inch,5_lb} → {0.88170588175_l,7.3_cm,0.73203815_kg}
./	Syntax: List1 ./ List2 or Matrix1 ./ Matrix2 Element-by-element division. Returns the term-by-term division of two lists or two matrices. Examples: {3,6,10}./{3,2,5} → {1,3,2} [[1,2],[3,4]]./[[3,4],[5,6]] → [[1/3,1/2],[3/5,2/3]] [[1+2*i,6+4*i],[3+6*i,8+4*i]] ./ [[1+2*i,3+2*i],[1+2*i,2+i]] → [[1,2],[3,4]]
.^	Syntax: Object1 .^ Object2 Returns the result of raising Object1 to the Object2 power. One or both objects are typically lists or matrices. The objects may be numerical values or expressions that return numerical results. . Examples: [[1,2],[3,4]] .^ 3 → [[1,8],[27,64]] {3,5,7} .^ 4 → {81, 625, 2,401} [[1+i,2+2*i],[i,3*i]].^3 → [[−2+2*i,−16+16*i],[-i,−27*i]] [[1+i,2+2*i],[i,3*i]].^i → [[0.428829006294+0.154871752464*i,0.230914901055+0.3931385059*i],[0.207879576351,9.45503713678ε−2+0.18513277813*i]]
QUOTE	QUOTE or ' ' Syntax: QUOTE(expression) Returns the expression unchanged and un-evaluated. This function is used mainly with Sto▸ in order to store a function in a function variable. For example, if you want to store SIN(X) in F1, you cannot do SIN(X)►F1 as SIN(X) would be evaluated and a numerical result would be stored into F1. Instead, use QUOTE(SIN(X))►F1 to store SIN(X) in F1.
|	Where Function Syntax: Expr| Var=Value Expr| {Var1=Val1, Var2=Val2, ...} Substitutes Value for Var in Expr and evaluates the result. This command can also take a list of substitutions for multiple variables. Examples: SIN(X)|X=π/6 → 0.5 (X+Y)|{X=2,Y=6} → 8
π	Mathematical Constant π The ratio of the circumference to the diameter of any circle. Internally represented as 3.14159265359. Example: π → 3.14159265359
Σ	Σ - Summation Syntax: Σ(expr, var, ivalue, fvalue) Finds the sum of expr with respect to var as var goes from ivalue to fvalue in steps of 1. Examples: Σ(X²,X,1,5) → 55 Σ({'X^2','Y-3'},{'X','Y'},{1,3},{5,10}) → {55,28}
∂	∂ - Numerical Derivative Syntax: ∂(Expr, Var=Value)

Help Topics Tree		13217	Help Text
			Numerical derivative. Returns the numerical derivative of an expression, with respect to a variable, at a given value. Example: ∂(1/X-X,X=5) returns -1.04 in Home view and -26/25 in CAS view This command can be used in the Symbolic view of the Function app to plot the graph of a function defined as a derivative. F1(X)= ∂(1/X-X, X=X) will plot the graph of the derivative of f(X)=1/X-X.
		∫	∫ - Numerical Integral Syntax: ∫(Expr, Var, Val1, Val2) 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. Example: ∫(2*X,X,0,3) → 9
		i	Imaginary Value Syntax: ***i*** The square root of -1.
Units			You can add units to numbers to make them measurements. You can also retrieve the values of many common physical constants. These features are available by pressing Shift Units. The following menu buttons appear: Menu Buttons: • Tools: for performing certain operations on measurements • Units: opens a menu with 18 sub-menus, each offering units from a particular domain (area, volume, electricity, etc.). • Const: opens a menu with 4 sub-menus, each offering constants from a particular domain (chemistry, physics, quantum mechanics, etc.). • Value: choose whether to retrieve the value of a constant or its name. • OK: copies the selected command to the entry line Press the Esc key to close the menu without making a selection.
	Unit Tools		Four tools are available for you to manipulate measurements: • CONVERT: convert between units • MKSA: meter, kilogram, second, ampere system • UFACTOR: factor units • USIMPLIFY: simplify units
		CONVERT	Syntax: CONVERT(Value Unit1, 1_Unit2) Converts Value Unit1 to the corresponding value in compatible Unit2. Example: CONVERT(20_m,1_ft) → 65.6167979003_ft Alternative: 20_m ►_ft
		MKSA	Convert to Metric System Syntax: MKSA(Value_Unit) Converts the measurement Value_Unit to its corresponding value and unit in Unit's MKSA equivalent. MKSA stands for the Meter-Kilogram-Second-Ampere system. Examples: MKSA(32_yd) → 29.2608_m MKSA(75_mph) →33.528_m/s MKSA({33_(cm),4_(yd^3)}) → {0.33_m,3.05821943194_(m^3)}
		UFACTOR	Unit factor conversion Syntax: UFACTOR(Value_Unit1, 1_Unit2) Converts a measurement using a compound unit into a measurement expressed in constituent units. 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. Using UFACTOR, you can express a measurement in Coulombs as a product of Amperes and time. Examples: UFACTOR(100_C,1_A) → 100_A*s UFACTOR(100_C,1_min) → 1.66666666667_min*A
		USIMPLIFY	Unit Simplification Syntax: USIMPLIFY(Value_Unitsexpr) Simplifies Value in a complex unit expression Unitsexpr to an equivalent value in a simpler unit expression. Example: a Joule is defined as 1 kg*m²/s². USIMPLIFY(5_kg*1_m²/1_s²) → 5_J
	Units		Units Menu Press Shift Units to open the Units menu and then tap on the Units menu button. The menu has 18 sub-menus, each offering units from a particular domain (area, volume, electricity, etc.).

Help Topics Tree 13217	Help Text
	By adding a unit to a number you turn that number into a measurement. Certain arithmetic operations can be performed on measurements, even if the measurements have different units. (However, the units must all be consistent: all length, all volume, etc.) For example, to add 5 cm to 2 inches: 1. Enter 5 on the entry line. 2. Press Shift Units. 3. With the Units menu open, tap Length and select cm. 5_cm appears on the entry line. Note the underscore character. 4. Enter + 2. 5. Press Shift Units. 6. With the Units menu open, tap Length and select in. 7. Press Enter. The result is 10.08_cm. Note that the unit of the first entered measurement is used for the result. PREFIXES Note that the very first item on the Units menu is Prefix. Prefixes are not units, but multipliers. Choose the prefix you want before choosing the unit.
Prefix	Unit 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 10^{24} Z: zetta 10^{21} E: exa 10^{18} P: peta 10^{15} T: tera 10^{12} G: giga 10^{9} M: mega 10^{6} k: kilo 10^{3} h: hecto 10^{2} D: deca 10 d: deci 10^{-1} c: centi 10^{-2} m: milli 10^{-3} µ: micro 10^{-6} n: nano 10^{-9} p: pico 10^{-12} f: femto 10^{-15} a: atto 10^{-18} z: zepto 10^{-21} y: yocto 10^{-24} Note: D stands for deca (more commonly abbreviated as da).
Length	Length Units m : meter * cm : centimeter (1/100 m) mm : millimeter (1/1000 m) km : kilometer (1,000 m) au : astronomical unit (149,597,870,700 m) lyr : light-year (9.46052840488ᴇ15 m) pc : parsec (3.261563407982 lyr) Å : angstrom (10ᴇ-10 m) fermi : fermi (10ᴇ-15 m) yd : yard (3 ft) ft : foot (12 in) in : inch (2.54 cm) mile : mile (5,280 ft) fath : fathom (6 ft) ftUS : US survey foot (1200/3937 m) miUS : US survey mile (5,280 ftUS) rod : rod (16.5 ft) chain : chain (66 ft) nmi : nautical mile (1,852 m) mil : (1/1000 in) * = SI base unit
Area	Area Units m^2 : square meter km^2 : square kilometer cm^2 : square centimeter yd^2 : square yard ft^2 : square foot in^2 : square inch $mile^2$: square mile

Help Topics Tree 13217	Help Text
	ha : hectare (10,000 m^2)
	a : are (100 m^2)
	acre : acre (43,560 ft^2)
	b : barn (10ᴇ-28 m^2)
	$miUS^2$: square US survey mile
Volume	Volume Units
	m^3: cubic meter
	cm^3 : cubic centimeter (1ᴇ-6 m^3)
	l : liter (1,000 cm^3)
	ml : milliliter (1 cm^3)
	yd : cubic yard
	ft^3 : cubic foot
	in^3 : cubic inch
	ozfl : US fluid ounce (1/128 galUS)
	ozUK : UK fluid ounce (1/160 galUK)
	qt : US liquid quart (1/4 galUS)
	liqpt : US liquid pint (1/8 galUS)
	ptUK : UK liquid pint (1/8 galUK)
	galUS : US liquid gallon (3.785411784 l)
	galUK : UK liquid gallon (4.54609 l)
	tbsp : US tablespoon (1/2 ozfl)
	tsp : US teaspoon (1/6 ozfl)
	bu : UK bushel (≈ 2,219.36 in^3)
	pk : US peck (1/4 buUS)
	bbl : Oil barrel (42 galUS)
	st : stère (1 m^3)
	buUS : US bushel (≈ 2150.42 in^3)
	cu : US cup (1/16 galUS)
	fbm : board foot (1/12 ft^3)
Time	Units of Time
	s : second *
	min : minute (60 s)
	h : hour (60 min)
	d : day (24 hrs.)
	yr : year (≈ 365.2422 d)
	Hz : hertz (Cycles per Second)
	* = SI base unit
Speed	Units of Speed
	m/s : meters per second
	cm/s : centimeters per second
	ft/s : feet per second
	kph : kilometers per hour
	km/h : kilometers per hour
	mph : miles per hour (1.609344 km/h)
	mile/h : miles per hour
	knot : knot (1 nmi/h or 1.852 km/h)
	rad/s : radians per second
	tr/min : turns per minute
	tr/s : turns per second
Mass	Units of Mass
	kg : kilogram *
	g : gram (1/1000 kg)
	slug : slug (≈ 32.174 lb)
	lb : pound (0.45359237 kg)
	oz : ounce (1/16 lb)
	tonUK : long ton or imperial ton (2,240 lb)
	tonUS : ton or short ton (2,000 lb)
	ozt : troy ounce (31.1034768 g)
	grain : grain (1/7,000 lb)
	lbt: troy pound (≈ 373.24 g)
	u : unified atomic mass unit (1.660538921ᴇ-27 kg)
	t : tonne or metric ton (1,000 kg)
	ct : carat (0.2 g)
	mol : mole (6.0221412927ᴇ23) *
	* = SI base unit
Acceleration	Units of Acceleration
	m/s^2 : meters per second per second
	ft/s^2 : feet per second per second
	grav : acceleration of gravity (≈ 32.174 ft/s^2 or ≈ 9.81 m/s^2)

Help Topics Tree 13217	Help Text
	Gal : galileo (1 cm/s^2)
	rad/sec^2 : radians per second per second
Force	Units of Force
	kg*m/s^2 : kilogram meter per second per second
	N : newton (1 kg m/s^2)
	dyn : dyne (1 g cm/s^2)
	lbf : pound-force (4.4482216 N)
	kip : kip (1,000 lbf)
	gf : gram-force (9.80665 mN)
	pdl : poundal (1 lb ft/s^2)
Energy	Units of Energy
	kg*m^2/s^2 : (1 Joule)
	J : joule (1 kg m^2/s^2)
	Wh : watt Hour (3,600 J)
	kWh : kilowatt hour (3.6E6 J)
	ft*lbf : foot-pound force (≈ 1.355818 J)
	kcal : kilocalorie (4186.8 J)
	cal : calorie (4.1868 J)
	eV : electron volt (1.602177733E-19 J)
	MeV : mega-electron volt (1,000,000 eV)
	Btu : British thermal unit (1,055.05585262 J)
	erg : erg (1E-7 J)
	thermEC : therm(EC) (1.05506E+8 J)
	thermUS : therm(US) (1.054804E+8 J)
	thermUK : therm(UK) (≈ 105,505,585.257 J)
	toe : tonne of oil equivalent (41,868,000,000 J)
	tec : tonne of coal equivalent (2.784E10 J)
	lep : liter of oil equivalent (35,788,320 J)
	boe : barrel of oil equivalent (6.1178632E9 J)
Power	Units of Power
	kg*m^2/s^3 : (1 Watt)
	W : watt (1 kg m^2/s^3)
	MW : megawatt (1,000,000 W)
	hp : horsepower (745.699871582 W)
	ft*lbf/s : foot-pound-force per second (1.35581794833 W)
Pressure	Units of Pressure
	kg/m*s^2 : (1 Pascal)
	Pa : pascal (1 kg/m·s^2)
	bar : bar (100,000 Pa)
	atm : atmosphere (101,325 Pa)
	psi : pound-force per square inch (6,894.75729317 Pa)
	torr : torr (133.322368421 Pa)
	mmHg : millimeter of mercury (133.322368421 Pa)
	inHg : inch of mercury (3,386.38815789 Pa)
	inH20 : inch of water (248.84 Pa @ 60 °F)
Temperature	Temperature Scales
	°C : Celsius scale
	(Freezing point of water 0 °C)
	°F : Fahrenheit scale
	(Freezing point of water 32 °F)
	K : Kelvin scale *
	(Freezing point of water 273.15 K)
	°R : Rankine scale
	(Freezing point of water 459.67 °R)
	* = SI base unit
Electrical	Electrical Units
	A : ampere (electric current) *
	V : volt (electric potential)
	C : coulomb (electric charge)
	Ohm : ohm Ω (resistance)
	F : farad (capacitance)
	Fdy : faraday (96,485.3365 C)
	Wb : weber (magnetic flux)
	H : henry (inductance)
	mho : mho ℧ (conductance)
	S : siemens (conductance)
	T : tesla (magnetic flux density)
	A*h : ampere-hour (3,600 C)
	* = SI base unit

Help Topics Tree 13217	Help Text
Light	Photometric Units cd : candela (lm/sr) * flam : footlambart (cd/m^2) * = SI base unit
Angle	Angle Units rad : radian (180/π degrees) deg : degree (π/180 radian) grad : gradient (1/400 turn) gon : gon (1/400 turn) arcmin : arc minute (1/60 degree) arcs : arc second (1/3600 degree) tr : turn (360 degrees)
Viscosity	Viscosity Units m^2/s : SI unit of kinematic viscosity P : poise dynamic viscosity (0.1 kg/m·s) St : stokes kinematic viscosity (1 cm^2/s)
Radiation	Radiation Units Activity Bq : becquerel Ci : curie (3.7ᴇ10 Bq) Absorbed Dose Gy : gray (1 J/kg) rd : rad (0.1 J/kg) Equivalent Dose rem : roentgen equivalent man (0.01 J/kg) Sv : sievert (1 J/kg) Exposure R : roentgen (2.58ᴇ-4 C/kg)
Constants	Physical Constants There are 29 physical constants from the fields of math, chemistry, physics and quantum mechanics that you can use in calculations. Press Shift Units and tap Const to display a menu of these constants. Tap Value• to retrieve the value of a constant with its units. Tap OK or press Enter to copy the constant into the entry line.

Math	Math Constants
	Euler's number
	e = 2.71828182846
	Maximum real number HP Prime is capable of representing
	MAXREAL=1.79769313486ᴇ308 (CAS View)
	MAXREAL=9.99999999999ᴇ499 (Home View)
	Minimum real number (closest to zero) HP Prime is capable of representing
	MINREAL=2.22507385851ᴇ-308 (CAS View)
	MINREAL=1ᴇ-499 (Home View)
	pi
	π = 3.14159265359
	Imaginary value
	i = √-1

Chemistry	Chemistry Constants NA : Avogadro (6.02214129ᴇ23 1/mol) k : Boltzmann (1.3806488ᴇ-3 J/K) Vm : molar volume of ideal gas (22.413968 l/mol) R : molar gas (8.3144621 J/(mol K)) StdT : standard temperature (273.15 K) StdP : standard pressure (101.325 kPa)
Physics	Physics Constants σ : Stefan–Boltzmann (5.670373ᴇ-8 W/(m^2 K^4)) c : speed of light (299,792,458 m/s) ε0 : vacuum permittivity (8.8541878176ᴇ-12 F/m) μ0 : vacuum permeability (1.25663706144ᴇ-6 H/m) g : acceleration of gravity (9.80665 m/s^2) G : gravitation (6.67384ᴇ-11 m3/(s^2 kg))
Quantum	Quantum Constants h : Planck (6.62606957ᴇ-34 J s) ħ : Dirac (1.054571729ᴇ-34 J s) q : electronic charge (1.602176565ᴇ-19 C) me : electron mass (9.10938291ᴇ-31 kg) qme : q/me ratio (175,882,008,800 C/kg) mp : proton mass (1.672621777ᴇ-27 kg) mpme : mp/me ratio (1,836.15267245)

Help Topics Tree	13217	Help Text
		α : fine-structure (7.2973525698ᴇ-3) ϕ : magnetic flux quantum (2.067833758ᴇ-15 Wb) F : Faraday (96,485.3365 C/mol) R∞ : Rydberg (10,973,731.5685 1/m) α_o : Bohr radius (5.2917721092ᴇ-11 m) μB : Bohr magneton (9.27400968ᴇ-24) μN : nuclear magneton (5.05078353ᴇ-27 J/T) λ_o : photon wavelength (1.2398419292ᴇ-6 m) f_o : photon frequency (2.41798934961ᴇ14 Hz) λc : Compton wavelength (2.4263102389ᴇ-12 m)
Variables		Variables are named references to objects (such as function definitions, numbers, matrices, the results of calculations, and the like). Some are built-in and cannot be deleted. But you can also create your own. Using a variable name in a formula returns the variable content (or value). Using a variable name as a destination in a STO (or :=) operation modifies its content (or value). Press the Vars key to access the variables menus. These menus are: • Home • CAS • App • User • Catalog The CAS menu appears only if you have created your own variables in CAS view, or created objects in the Geometry app. The User menu appears only if you have created your own variables in Home view, or created global variables in a program. The Catalog menu opens an integrated menu of all variables currently defined on your HP Prime. Tap the OK menu key to copy the selected variable name to the entry line or press the Esc key to back out of the Vars menu. In a menu of options, you can also select an entry by its number or by typing in the first letter or two of its name and pressing the Enter key.
Home Variables		The Home variables menu lists those variables that are commonly accessed in the Home view or that affect the working of Home view.
Real Variables		This menu contains the names of the real variables: A through Z and θ.
Complex Variables		This menu contains the names of the ten complex variables: Z0 through Z9.
List Variables		This menu contains the names of the ten list variables: L0 through L9.
Matrix Variables		This menu contains the names of the ten matrices: M0 through M9.
Graphics Variables		This menu contains the names of the ten graphic variables: G0 through G9.
Home Settings Variables		Variables for Home Settings This menu lists the names of the variables used to control the Home settings. They are: • HAngle • HFormat • HDigits • HComplex • Entry • Integer: Base • Integer: Bits • Integer: Signed

HAngle		The Home variable HAngle is used to set the Home view angle mode.
		HAngle := 0 for Radians (default)
		HAngle := 1 for Degrees
		HAngle := 2 for Gradians

HFormat		The Home variable HFormat controls how numbers are displayed in Home view. This variable may contain any integer from 0 through 5, each value having the following meaning:
		
		HFormat :=0 for Standard
		HFormat :=1 for Fixed
		HFormat :=2 for Scientific
		HFormat :=3 for Engineering
		HFormat :=4 for Floating
		HFormat :=5 for Rounded

HDigits		The Home variable HDigits controls the number of digits displayed after the decimal point when the number format is not Standard. HDigits := n, where n is an integer such that 0 ≤ n ≤ 11.
HSeparator		The Home variable HSeparator is used to control the separators used in number display. HSeparator may contain any integer from 0 through 10 corresponding to the selected Digit Grouping on Home Settings Page 1.
HComplex		The Home variable HComplex is used to control the Complex settings. This variable contains a two-digit integer. The units digit of HComplex controls the complex result from real input (1: enabled, 0: disabled) The tens digit controls the display of complex numbers (0: a+bi, 1: (a,b))

Help Topics Tree	13217	Help Text
		Thus HComplex := 10; sets the display of complex numbers as (a,b) and blocks complex results from real inputs. With the units digit of HComplex set to 0, ASIN(2) returns an error; set to 1, ASIN(2) returns 1.57079632679–1.31695789692*i.

Entry		Entry Variable
		Returns or sets the entry mode.
		Entry := 0 for Textbook (default)
		Entry := 1 for Algebraic
		Entry := 2 for RPN

Base		Base Variable
		Returns or sets the integer base format.
		Base := 0 for binary
		Base := 1 for octal
		Base := 2 for decimal
		Base := 3 for hexadecimal (default)

Signed		Signed Variable
		Returns or sets the integer signed format.
		Signed := 0 for unsigned (default)
		Signed := 1 for signed

Bits		Bits Variable Returns or sets the integer bit size. Bits := x, where 1≤x≤64
System Variables		This menu lists the names of the variables used to control the system settings or objects. They are: • Date • Time • Language • Notes • Programs • TOff
Date		Date Variables Returns the system date in YYYY.MMDD format. Date:= YYYY.MMDD changes the date.
Time		System Time Variable Returns the system time in DMS format. DMS format is HH°MM'SS" where HH = hours, MM = minutes, and SS = seconds. Time:= HH°MM'SS" sets the time.

Language		Language Variable
		The Home variable Language controls which language is used in Home view.
		Language:= 1 for English (Default)
		Language:= 2 for Chinese
		Language:= 3 for French
		Language:= 4 for German
		Language:= 5 for Spanish
		Language:= 6 for Dutch
		Language:= 7 for Portuguese
		Language:= 8 for Japanese

Notes		Notes Variable The Notes variable gives access to the notes saved in the calculator. With no argument, 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 number of notes). Notes("name") returns the contents of the note called 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 erased. Similarly, Notes("name"):="string" sets the value of note "name". If string is empty, the note is erased. If there is no note called "name", creates it with string as content.
Programs		Programs Variable 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 content of the nth program in the calculator (1 to number of programs) Programs(n):="string" sets the program source code for program n. If String is empty, erases the program. Programs("name") returns the source of program "name". Programs("name"):="string" sets program "name" source code to string. If string is empty, erases the program. If there is no program called "name", creates it.
TOff		Time Off Variable 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.

Help Topics Tree 13217	Help Text
Theme	Theme Variable Theme variable contains a list representing the current theme and the color shade as defined on Home Settings Page 2. May take a single number as input to return the indexed item from the list. Examples: Theme() Theme(1)
HVars	Home Variables 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) if the variable is a user function, returns the list of the parameters for that function; else returns 0. HVars(n):=value stores value in the nth home user variable. HVars("name"):=object stores object in the home user variable called "name". If no such variable exists, creates it. HVars(n or "name", 2):= {"Param1Name", ..., "ParamNName"} assumes that the specified user variable contains a function. Specifies what the parameters of that function are. HVars("name", {"param",...}):='function' creates a user function called name with params as the inputs and function defined in terms of those inputs. Examples HVars("Current",{"V","R"}):=QUOTE(V/R) creates a user function called Current to use Ohm's Law to calculate the value of I (current) given the input variables V (voltage) and R (resistance).
DelHVars	Delete Home Variables DelHVars(n) or DelHVars("name") deletes the specified home user variable.
CAS Variables	The CAS menu appears only if you have created your own variables in CAS view or created objects in the Geometry app. Objects created in the Geometry app are given a "G" prefix (such as GA, GB, etc.).
App Variables	App Variables Menu App variables are variables for settings and results within a specific app. The variables are arranged by app. For apps that yield results, each RESULTS variable can be used in an expression. For example, the Function app enables you to calculate various critical points, such as intersections, roots, extrema, and the like. If, for example, you have used the Function app to calculate the roots of a function, you could select the Root variable and press Enter. The value of the variable is retrieved. You could also include Root in an expression, such as X^2–2*X+Root. Note that variable names are case-sensitive.
User Variables	The User menu appears only if you have created your own variables in Home view, or created global variables in a program using the EXPORT command.
Vars Catalog	The Variables Catalog The Vars Catalog contains a comprehensive listing of all variables currently defined on your HP Prime. On the right side of the Vars Catalog header is a small information icon (i). Tap the icon to see the number of each type of variable currently defined on your HP Prime.
Memory Manager	The Shift of the Toolbox key (Mem) takes you to the Memory Manager, which is a catalog of catalogs. All the catalogs are listed here. Tap the View menu button or press Enter to enter a selected catalog and delete objects you no longer need. Menu Buttons: • Info: displays memory and storage space • Clone: clones the current HP Prime to an attached HP Prime • Send: sends all the data of a certain category (Lists, Matrices, etc.) to an attached HP Prime • View: opens the selected catalog
Backups	This screen lets you save, restore, or create new complete backups of your calculator. Menu Buttons • Restore: replace the current content of your calculator with the files from the selected backup archive • Delete: delete the currently selected backup archive • New: open a dialog to create a new backup. By default, the word "Backup" and the current date are provided.
Backup	Backup Calculator Memory Syntax: Backup("name") Backs up all the calculator memory under "name". Example: Backup("name")
Restore	Restore Calculator Memory Syntax: Restore(“name”) Restore() Returns the list of all the available backups. Restore("name") Restores the calculator memory to the state saved under "name". The current calculator memory will be erased.
DelBackup	Delete Calculator Backup

Help Topics Tree	13217		Help Text
			Syntax: DelBackup("name") Deletes the backup "name". Example: DelBackup("name")
Characters Menu			The Shift of the Vars key (Char) takes you to the Characters menu, which contains the extended character set of the HP Prime. Use the rocker wheel to select a character from the current set. Double-tap the character or tap Echo to add it to your current string in the input area. Add as many characters as you like. Tap OK to enter them at the current cursor location or press Esc to drop all characters and return to the edit line. Menu Buttons: • Echo: add the highlighted character to a string of one or more characters to be inserted into the current edit line • More: choose from a selection of character sets, each with multiple pages of characters • ▲ Page ▼: move up and down through the various pages of the current character set • OK: exit the Chars menu and paste the current string of characters into the edit line
Shortcut Palettes			Many operations and characters can be entered simply by selecting them from palettes. There is a: • Math Template palette – the Math Template key displays a palette of pre-formatted templates representing common mathematical operations and expressions, such as nth root, differentiation, and integration (as well as vectors, matrices, and sexagesimal numbers). • Relations palette – (Shift 6) displays Boolean and relational operators useful in mathematics and programming (such as greater than, not equal to, and OR). • Special characters palette – (Shift 9) displays a palette of characters common in mathematics and statistics (such as infinity, mean, and standard deviation), as well as Greek characters
	Math Template		The Math Template key displays a palette of pre-formatted templates representing common mathematical operations and expressions, such as nth root, differentiation, and integration (as well as vectors, matrices, and sexagesimal numbers). Either tap on the template you want, or user the rocker wheel to highlight the template and press Enter. The template is copied to the entry line ready for you to add in the values. Press Esc to close the palette without making a selection.
		Math Template	The Math Template key displays a palette of pre-formatted templates representing common mathematical operations and expressions, such as nth root, differentiation, and integration (as well as vectors, matrices, and sexagesimal numbers). Either tap on the template you want, or user the rocker wheel to highlight the template and press Enter. The template is copied to the entry line ready for you to add in the values. Press Esc to close the palette without making a selection.
	Special Characters Palette		Press Shift 9 (!,∞,→) to display the Special Characters palette. This palette displays many common mathematical symbols (such as infinity, mean, and standard deviation), as well as Greek characters. Press Esc to close the palette without making a selection. Note that a full list of characters can be displayed by pressing Shift Vars (Chars).
	Relations Palette		Press Shift 6 (≤,≥,≠) to display the relations palette. This palette displays Boolean and relational operators useful in mathematics and programming (such as ≤, ==, >, AND, and OR). Press Esc to close the palette without making a selection.
Keyboard			Here you can get help on any key on the keyboard and any Shift-key combination.
	a_b⁄c		Decimal Conversions: a_b⁄c Decimal to fraction conversion In Home view, toggles the last entry 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.
	Directional Pad		The directional pad moves the cursor around the display. Press the Shift key first to move to the beginning, end, top, or bottom of a list or the display. Press the ALPHA key first to jump 1 screen at a time. In the Plot view, with TRACE on, the rocker wheel left/right move the tracer left and right along the current graph. Use the rocker wheel up/down to switch to the previous or next active graph.
	X,T,θ,N		X,T,θ,N Key The X,T,θ,N key is a typing aid that enters the appropriate independent variable for the current app. In the Parametric app, it enters T. In the Polar app, it enters θ. In the Sequence app, it enters N. In the Solve app, it enters the first variable found in the checked equation. In all other apps, it enters X.
	Backspace		Backspace Key

Help Topics Tree 13217	Help Text
	The Backspace key deletes the character to the left of the cursor. In a setup screen, such as Plot Setup, press this key to reset any field to its default value. In the history, press this key to delete the highlighted entry or result.
TAN	Tangent Syntax: TAN(Value) Returns the tangent of Value. Value is interpreted as radians, degrees or gradians, depending on the setting of Angle Measure in Home Settings, CAS Settings, or Symbolic Setup. Example: TAN(45) → 1 (Degrees mode) TAN(1+i) → 0.27175258532+1.08392332734*i TAN({45,0}) → {1,0} (Degrees mode) TAN((π/4)_rad) → 1
x^2	Square Function Syntax: Value^2 Returns the square of Value. Examples: 3^2 →9 2.4^2 → 5.76 (2+3i)^2 →-5+12i [[2,2],[3,3]]^2 → [[10,10],[15,15]] (5_m)^2 → 25_(m^2) {2,4,6,8}^2 → {4,16,36,64}
x^y	Power Function Syntax: x^y Returns the result of raising x to the power of y. Also accepts complex numbers. Examples: 2^3 → 8 4.5^3.2 → 123.10623351 (2+3i)^3 → −46+9*i (1-2*i)^i → 2.09777264942+2.18044142399*i [[1,2],[3,4]]^3 → [[37,54],[81,118]] (10_km)^3 → 1000_(km^3) {1,2,3}^4 → {1,16,81}
÷	Divide Syntax: Object1 ÷ Object2 Returns the result of dividing Object1 by Object2. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions. Examples: 3 / 2 → 1.5 4.6 / 2.5 → 1.84 (3+2*i) / (1-2*i) → −0.2+1.6*i [[1,2,3],[4,5,6],[7,8,9]] / 4 → [[0.25,0.5],[0.75,1]] [[1,2],[3,4]] / [[5,6],[7,8]] → [[5,4],[−4,−3]] [[1,2],[3,4]] / {1,2,4} → {[[1,2],[3,4]],[[0.5,1],[1.5,2]],[[0.25,0.5],[0.75,1]]} 12_((kg*m)/s^2)/4_(m/s^2) → 3_kg {12,9,8} / {6,3,2} → {2,3,4}
,	Comma Enters a separator character. The character entered depends on the selected Digit Grouping on Home Settings Page 1.
Numerals	Enters the numerals 0 through 9.
×	Multiply Syntax: Object1×Object2 Returns the result of multiplying Object1 and Object2. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions. Examples: 3 * 2 → 6 4.1 * 2.4 → 9.84 (3+2*i) * (1-2*i) → 7-4i [[1,2,3],[4,5,6],[7,8,9]] * 3 → [[3,6,9],[12,15,18],[21,24,27]] (3+2*i) .* [[1,2],[3,4]] → [[3+2*i,6+4*i],[9+6*i,12+8*i]] [[1,2],[3,4]] * [[5,6],[7,8]] → [[19,22],[43,50]] [[1,2],[3,4]] * {1,2,3} → {[[1,2],[3,4]],[[2,4],[6,8]],[[3,6],[9,12]]}

Help Topics Tree 13217	Help Text
	3_kg*4_(m/s^2) → 12_((kg*m)/s^2) {1,2,3} * {4,5,6} → {4,10,18}
-	Subtract Syntax: Object1 - Object2 Returns the result of subtracting Object2 from Object1. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions. Examples: 3 - 2 → 1 0.5 - 3.7 → -3.2 (2+5i) - (3+2i) → 1+3i [[1,2],[3,4]] - [[0.5,0.6],[1.5,4.1]] → [[0.5,1.4],[1.5,−0.1]] [[1,2],[3,4]] - {1,2,3} → {[[0,1],[2,3]],[[−1,0],[1,2]],[[−2,−1],[0,1]]} {1_l,20_cm,3_kg}-{4_ozfl,5_inch,5_lb} → {0.88170588175_l,7.3_cm,0.73203815_kg}
+	Add Syntax: Object1 + Object2 Returns the result of adding Object2 to Object1. The objects may be numerical values or expressions that return numerical results. The objects may also be lists or matrices of appropriate dimensions. Examples: 3 + 2 → 5 0.5 + 3.7 → 4.2 (2+5i) + (3+2i) → 5+7i [[1,2],[3,4]] + [[0.5,0.6],[1.5,4.1]] → [[1.5,2.6],[4.5,8.1]] [[1,2],[3,4]] + {1,2,3} → {[[2,3],[4,5]],[[3,4],[5,6]],[[4,5],[6,7]]} 32_tonUS+3_t → 35.3069339328_tonUS {1,2,{1,0,1}}+{5,4,3} → {6,6,{4,3,4}}
.	Decimal Point Pressing the decimal point (.) key enters the decimal mark character. The character entered depends on the selected Digit Grouping on Home Settings Page 1. If the selected Digit Grouping uses the comma as the decimal mark, press ALPHA before pressing this key to enter a period.
Enter	Enter Key Executes the expression in the entry line. Also works like the OK menu button to accept the current state of an input field.
Define	Create a user-defined function The Define dialog box allows you to define a user function without having to create a program. Simply enter the name of the function in the Name field and the function in the Function field. When you tap OK, you will be presented with a list of the variables used in the function. Check each variable that is an INPUT to your function, press OK and you are done. You can now use your function in the system. Example: To create the function SINCOS(A,B) = SIN(A)+COS(B)+C Enter SINCOS as the name, SIN(A)+COS(B)+C as the function, tap OK, check A and B and uncheck C. Now tap OK. Now, on the entry line, you can type SINCOS(1,2) to calculate SIN(1)+COS(2)+C. If you enter the name of an existing function in the Name field, the Function field will be filled automatically with the function associated with that name. This will allow you to easily modify the definition. In the Define dialog box, tap Choose to see a list of all the user-defined functions. To delete a function, use the rocker wheel to highlight it in the list and then press the backspace key. To edit a function, just select it from the list.
° ' "	Degrees Minutes Seconds Decimal to sexagesimal conversion This Shift-key combination toggles a numerical result between decimal and sexagesimal representations. Any decimal result can de displayed in sexagesimal; that is, in units subdivided into groups of 60. This includes degrees, minutes, and seconds as well as hours, minutes, and seconds. Enter your expression in the Home view and then press the Shift key, followed by this key, to convert the result to sexagesimal; repeat to return to a decimal representation. During entry of any expression, this Shift-key combination inserts the degree, minute, and seconds symbols (°, ', ").
EEX	EEX Key The EEX key is used to enter numbers in 'exponential' notation, also known as scientific notation, scientific form, standard form or standard index form. For HP Prime, a number in exponential notation is represented by 2 parts separated by the ᴇ character (corresponding to the EEX key). The first part, usually referred to as the mantissa, is a real number. The second part, usually referred to as the exponent, is an integer. The number represented by this notation is mantissa*10^exponent. Thus typing 3 EEX 5 generates 3ᴇ5 on the command line, which returns the number 300000. Example: 3ᴇ5 → 300000

Help Topics Tree 13217	Help Text
Clear	Clear Key The Shift-Esc (Clear) key combination clears the edit line, if active. In a field in a view, such as Plot Setup, press this key to reset all fields in the view to their default values. In the display history, press this key to delete all entries and results in the history.
e^x	Exponential Function Syntax: e^Value e^(Expr) The Shift LN (e^x) key combination is the exponential function; it returns e raised to the power of Value. Example: e^1 → 2.71828182846
√	Square Root Syntax: √Value or √(Expr) The Shift x^2 (√) key combination is the square root function and returns the positive square root of Value or the positive square root of the numerical result of Expr. Examples: √9 → 3 √3 → 1.73205080757 √(2+3i) → 1.67414922804+0.89597747613*i √[[10,10],[15,15]] → [[2,2],[3,3]] √(9_(km²)) → 3_km √{9,4,36,81} → {3,2,6,9}
Copy	Press Shift View (Copy) to copy the highlighted value, expression, text, or object to the clipboard.
Paste	Press Shift Menu (Paste) to open the Paste clipboard. Once the clipboard is open, you can scroll to highlight an object. Tap the OK menu key or press Enter to paste the highlighted object into the edit line at the cursor position or into a selected field in an input form. Press the Esc key to close the clipboard without pasting anything. Menu Buttons: • Show: displays the selected item full-screen using textbook format • Clear: clears all items in the clipboard • Delete: deletes the selected item from the clipboard • OK: pastes the selected item into the edit line at the cursor position or into a selected field in an input form
x^{-1}	Inverse Syntax: Object⁻¹ Returns the inverse of Object. The object may be a number, an expression that results in a numerical value, a list or a square matrix. Examples: 2^-1 → 0.5 4.5⁻¹ → 0.222222222222 (2+4*i)⁻¹ → 0.1-0.2*i [[1,2,3],[2,0,1],[3,1,2]]⁻¹ → [[−1/3,−1/3,2/3],[−1/3,−7/3,5/3],[2/3,5/3,-4/3]] (42.3_(m/s²))⁻¹ → 2.36406619385ᴇ−2_(s²/m) {1/3,4/5,1/6}⁻¹ → {3,5/4,6}
∡	Angle Syntax: Value1 ∡ Value2 ∡: Angle of a complex number in polar mode. Enters the angle symbol. Used to enter complex numbers in polar form. With a complex number in Ans and in the display as the last result, press this key to toggle between polar and rectangular forms of the complex number.
Base	Base Key The Base key enters the character # in the edit field, unless the currently selected item (or most recent result) is an integer, in which case the Edit Integer screen is displayed. In the Edit Integer dialog box, the Was field at the top shows the original integer you selected in Home view. The Out field shows the edited integer. Both integers are initially displayed in the default base as specified in Home Settings Page 1.

Help Topics Tree 13217	Help Text
	The 16 field is the hexadecimal representation of Out. The 10 field is the decimal representation of Out. The box below the decimal value shows the 64 bit binary (bit) representation of Out. Changing the value by any of the following methods updates the value of Out and all representations: • Shift the integer left or right. Bits shifted off either end are lost. - Drag the screen left or right to shift by 1 bit in the corresponding direction - Rocker wheel left/right to shift 1 bit - Alpha rocker wheel right/left to shift by 4 bits (1 nibble) - Shift rocker wheel right/left shifts by 8 bits (one byte) • Change word size (number of bits) of the integer - Drag the screen down/up to increase/decrease by 1 - Press rocker wheel up/down to increase/decrease by 1 - Alpha rocker wheel up/down to increase/decrease by 4 (1 nibble) - Shift rocker wheel up/down to increase/decrease by 8 (one byte) • Change sign of the integer: Press +/- key • Cycle through base settings (system, hex, decimal, octal and binary): + and – keys Menu Buttons: • Reset: returns all changes to their original state • Base: cycles through the bases; same as pressing + • Signed: toggles the word size between signed and unsigned • NOT: returns the ones' complement (that is, each bit in the specified word size is inverted: 0s are replaced by 1s and vice versa) • Edit: toggles edit mode. Edit mode is active if a bullet character appears on the Edit button. When Edit mode is active, a digit selector highlightes a single digit and you can move abut the dialog using the rocker wheel. The hex and decimal fields can be modified, as can the bit representation, one digit at a time. A change in any 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. Validate the change in the number using OK or cancel using the Esc key.
=	Equal Key Syntax: Expr1=Expr2 Enters the equal sign. Used to enter equations in the Solve or Advanced Graphing apps as well as in the CAS. Other uses of the equal sign: • The Boolean (logical) operator for equivalence is == • The operator for greater-than-or-equal-to can be entered as >= • The operator for less-than-or-equal-to can be entered as <= • The assignment operator is :=
CAS	CAS Key Opens the Computer Algebra System (CAS).
Vars	Vars Key The Vars key opens menus for you to choose variables. Home, CAS, App, and User variables can be selected from these menus.
Apps	Apps Key Pressing Apps opens the Application Library from where you can select an app to reset, open, or send.
Symb	Symbolic View Symb Key Opens the Symbolic View for the active app. What you do in this view depends on the app. For instance, you can define functions and open sentences, create geometric objects, set up a hypothesis test, and define statistical analyses. Note that the Symbolic view is not used in the Spreadsheet app, Finance app, the Solver apps, and the Explorer apps.
Plot	Plot View Plot Key Opens the Plot View for the active app. What you do in this view depends on the app. For instance, you can plot functions and open sentences, create geometric objects, and explore linear, quadratic, sinusoidal, and amortization graphs. Note that the Plot view is not used in the Spreadsheet, Linear Solver, or Triangle Solver apps.
Num	Numeric View Num Key Opens the Numeric View for the active app. What you do in this view depends on the app. For instance, you can explore tables of values generated by functions, make geometric measurements, do spreadsheet calculations, and enter data for statistical analyses. Note that the Numeric view is not used in the Explorer apps.
Menu	Menu Key Menu Key The context-sensitive menu provides options for you to: • copy an item from Home view to CAS view • copy an item from CAS view to Home view

Help Topics Tree 13217	Help Text
	• view messages you have received via the Connectivity Kit The Menu key also may give you access to functions specific to your current app. For example, in the Spreadsheet app, the menu includes such functions as SUM, AVERAGE, AMORT, etc.
Esc	Escape Key Esc: Escape • Clears the entry line or closes a menu. • Closes a menu or pop-up window • Closes most views and input forms • Cancels changes in an input field being edited
( )	Parentheses Press ( ) to insert a pair of parentheses.
ALPHA	ALPHA Key Press the ALPHA key to access the text character printed in orange on the bottom right of a key. For example, to type Z in Home view, press Alpha 2. When you press ALPHA, the annunciator A...Z is displayed in the title bar. This indicates the next key pressed will insert uppercase text. However, in CAS view, the annunciator a...z will be displayed, indicating the next key will insert lowercase text. Press Shift to change the case: the annunciator will change to indicate the case of the next character. Press ALPHA ALPHA to put the keyboard into lock mode where you can type more than one letter consecutively. The title bar annunciator changes to AZ indicating uppercase lock (or az in CAS). Press Shift before the key to change case; the annunciator changes to a..z indicating next character lowercase (A..Z in CAS). Press Shift ALPHA to change the case of the lock; the annunciator will change from AZ to az or from az to AZ. Press ALPHA to leave alpha-lock. The text annunciator is cleared.
Shift	Shift Key Press Shift once to insert a blue-colored key character. Used in conjunction with ALPHA to enter lowercase characters.
On	On Key Turns on the calculator. Once on, the key works as an alternative Esc key. You can use it to clear the entry line or close a menu. Press and hold the On key then press a second key to perform the following. • Clear Memory : On & Apps & Esc • Reboot : On & Symb • Terminal screen : On & T (÷) • Decrease display brightness: On & - • Increase display brightness : On & + • Exam Mode screen : On & Esc or On & E (a b/c)
⌴	Space Key Space: alphabetical space Enters a space
Sto ►	Sto ► Key Stores a value in a variable (that is, assigns 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. Example: 3 ► A stores the value 3 in the variable A You can access the store command by pressing Shift EEX.
Symb Setup	Symbolic Setup View Displays the Symbolic Setup view. This 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 setting "System" indicates the system setting for that field will be used. The override applies to every function as long as the modified app is active. Switching to another app changes the settings to match that app. Switching back to the modified app restores the settings for it. Some apps have some overrides pre-set due to the nature of calculations primarily performed in that app. For example, the Finance app has number format set to Fixed, 2 decimals.
Plot Setup	Plot Setup View Displays the Plot Setup view. This view is used primarily to modify the appearance of graphs and the plotting environment. It is not used in apps which have no Plot view.
Num Setup	Numeric Setup View Displays the Numeric Setup view. This view is used primarily to specify the appearance of Numeric view and to set the zoom factor.
'□'	Single Quotes Press the '□' key to insert a pair of single quotation marks. Single quotes are used to enter algebraic expressions.
{ }	Braces { } : Braces Inserts a pair of braces. Braces are used to enclose lists.
[ ]	Square Brackets [ ] : Square Brackets

Help Topics Tree 13217	Help Text
	Inserts a pair of square brackets. These are used to enclose vectors and matrices.
Eval	Eval Key Syntax: EVAL(Expr) Evaluates the expression Expr. Example: EVAL(2+3) → 5
!,∞,→	Special Characters Palette Press Shift 9 (!,∞,→) to display the Special Characters palette. This palette displays many common mathematical symbols (such as infinity, mean, and standard deviation), as well as Greek characters. Press Esc to close the palette without making a selection. Note that a full list of characters can be displayed by pressing Shift Vars (Chars).
≤,≥,≠	Relations Palette Press Shift 6 (≤,≥,≠) to display the relations palette. This palette displays Boolean and relational operators useful in mathematics and programming (such as ≤, ==, >, AND, and OR). Press Esc to close the palette without making a selection.
<	Less Than Syntax: Value1 < Value2 Tests whether or not Value1 is less than Value2. Returns 1 if true, 0 if false. Examples: 2 < 1 → 0 1.999999999999 < 2.000000000001 → 0 75_mph < 120_kph → 0 {5<9<18,-2<0<9<122<3} → {1,0} ((x+1)*(x-2))<(x^2-x-2) → false
≤	Less Than or Equal To Syntax: Value1 ≤ Value2 Tests whether or not Value1 is less than or equal to Value2. Returns 1 if true, 0 if false. Example: 2 ≤ 1 → 0 Alternative: <=
>	Greater Than Syntax: Value1 > Value2 Tests whether or not Value1 is greater than Value2. Returns 1 if true, 0 if false. Examples: 2 > 1 → 1 2.000000000001 > 1.999999999999 → 0 75_mph > 120_kph → 1 {4>2,5>3>1>0} → {1,1} ((x+1)*(x-2))<(x^2-x+2) → 1
≥	Greater Than or Equal To Syntax: Value1 ≥ Value2 ≥ Greater than or equal to Tests whether or not Value1 is either greater than or equal to Value2. Returns 1 if true, 0 if false. Example: 3 ≥ 4 → 0 Alternative: >=
==	Equivalence Syntax: Value1 == Value2 Tests if Value1is equal toValue2. Returns 1 if true, 0 if false. If Value1 and Value2 are listes, returns a list containing 0 or 1 for each pair of items. If two expressions are tested for equivalence, the test results are sensitive to the CAS Settings. If the setting Simplify is set to None, a==b checks that a and b have the same tree representation. If Simplify is set Maximum, the auto simplification function is called on a-b and the system returns true if the result of the Simplify function on a-b is 0. Note: when the calculator evaluates an expression containing a test, each test is evaluated as it is encountered before proccessing rest of the expression. Therefore a compound expression such as A==B==C will evaluate A==B first and then compare the result of that evaluation (either 0 or 1) to C. In other words, there are implied parentheses around each successive argrument, so A==B==C==D is evaulated as ((A==B)==C)==D. Examples: 3 == 2 → 0 2.375 == 2+0.375 → 1

Help Topics Tree		13217	Help Text
			(2+5i) == (3,2) + (-1,3) → 1 [[1,2],[3,4]] == [[3,4],[5,6]]-2 → 1 {"red","white","blue"} == {"red","white","black"} → {1,1,0} (((x+1)*(x-2))==(x^2-x-2)) → 1 3==3==3==3 → 0 (3==3)==(3==3) → 1
		≠	Not Equal To Syntax: Value1 ≠ Value2 Tests if Value1 is not equal to Value2. Returns 1 if true, 0 if false. Alternative: <> Note: when the calculator evaluates an expression containing a test, each test is evaluated as it is encountered before proccessing rest of the expression. Therefore a compound expression such as A≠B≠C will evaluate A≠B first and then compare the result of that evaluation (either 0 or 1) to C. In other words, there are implied parentheses around each successive argrument, so A≠B≠C≠D is evaulated as ((A≠B)≠C)≠D. Examples: 3 ≠ 5 → 13 ≠ 5 ----> 1 √25 ≠ 4 NTHROOT 625 → 0 (2+5i) ≠ (3+2*i)*(16/13,11/13) → 0 [[1,2],[3,4]] ≠ [[3,4],[5,6]]-[[2,1],[0,2]] → 1 {"red","white","blue"} ≠ {"red","white","black"} → {0,0,1} 180_deg ≠ 3.14159_rad → 1 ((x+1)*(x-2))≠(x^2-x-2) → false 3≠4≠0≠2 → 1 (3≠4)≠(0≠2) → 0
		AND	Logical AND Syntax: Value1 AND Value2 For Real numbers, returns 1 if both Value1 and Value2 are non-zero; otherwise returns 0. For Integers and Strings, AND is performed bitwise, returning 1 if corresponding bits are both 1, otherwise 0. Examples: 3 AND 2 → 1 0 AND 1 → 0 0 AND 0 → 0 {3,0,0} AND {2,1,0} → {1,0,0} 75_mph > 120_kph AND 180_deg ≠ 3.14159_rad → 1 #CC44h AND #44CCh → #4444h "a" AND "b" → "`" X:=0; 1 AND (X:=3); 0 AND (X:=5); X → 3 7 > 3 AND 5 < 9 AND 3 ≠ 2 → 1
		NOT	Logical NOT Syntax: NOT Value For Real numbers, returns 1 if Value is zero; otherwise returns 0. For Integers and Strings, NOT is performed bitwise, flipping all 1's to 0's and all 0's to 1's. Examples: NOT 3 → 0 NOT 0 → 1 A:=32; B:=2^5; NOT (A=B) → 0 NOT #DFCA:16h → #2035:16h NOT #1776:16o → #176001:16o NOT "abcdefg" → "ﾞﾝﾜﾛﾚﾙﾘ" NOT {"ab","cd"} → {"ﾞﾝ","ﾜﾛ"}
		OR	Logical OR Syntax: Value1 OR Value2 For Real numbers, returns 1 if either Value1 or Value2 is non-zero; otherwise returns 0. For Integers and Strings, OR is performed bitwise, returning 1 if either corresponding bit is 1, otherwise 0. Examples: 3 OR 2 → 1 0 OR 2 → 1 0 OR 0 → 0 {3,0,0} OR {2,1,0} → {1,1,0} 3_inch==7.62_cm OR 9_(inch²)==58.0644_(cm²) → 1 #CC44h OR #44CCh → #CCCCh

Help Topics Tree 13217			Help Text
			"c" OR "d" → "g" X:=0; 0 OR (X:=7); 1 OR (X:=9); X → 7 7 ≤ 3 OR 5 < 9 OR 3 ≠ 2.9 + 0.1 → 1
		XOR	Logical XOR Syntax: Value1 XOR Value2 For Real numbers, returns 1 if either Value1 or Value2 is non-zero but not both; otherwise, returns 0. For Integers and Strings, XOR is performed bitwise, returning 1 if exactly one bit is 1 and the corresponding bit is 0, otherwise 0. Examples: 3 XOR 2 → 0 0 XOR 2 → 1 0 XOR 0 → 0 {3,0,0} XOR {2,1,0} → {0,1,0} 3_inch==7.62_cm XOR 9_(inch²)==58.0644_(cm²) → 0 #CC44h XOR #44CCh → #8888h "C" XOR "b" → "!" X:=0; 0 XOR (X:=7); 1 XOR (X:=9); X → 9 7 ≤ 3 XOR 5 < 9 XOR 3 ≠ 2.9 + 0.2 → 0
	_		Underscore _ : Underscore Enters the underscore character (_).
	CAS Settings		Displays the CAS Settings screen with various settings for you to configure how the computer algebra system works.
	Del		Delete Key Press Shift Backspace (Del) to delete the character to the right of the cursor.
	Help		Help Key Displays the online help. The help provided will be relevant to the screen that is open at the time you press this key. The online help also gives you help on all the keys on the keyboard. It also provides a tree of help topics that you can browse. Menu Buttons: • Tree: displays the full help tree • Keys: puts the help system into Key mode; once in Key mode, press any keyboard key to get help on that key • ▲ Page ▼: Page up and down; press the right side to go down a page and the left side to go up a page in a multi-page help script • OK: leave the help system
	User		User Keyboard Press Shift Help (User) to activate the user keyboard. You can assign alternative functionality to any key on the keyboard, including the functionality provided by the Shift and ALPHA keys. This enables you to customize the keyboard to your particular needs. When the user keyboard is active, the keyboard works as you have defined it. Press User once (1U appears in the title bar) to activate the user keyboard for just the next key press. Press User twice (↑U appears in the title bar) to keep the user keyboard active (locked on). Press it once more to deactivate it.
	≈		Approximate ≈ : Numerical approximation Provides a numerical approximation to the selected item in the CAS history. Example: In CAS 1 ÷ 4 ≈ returns 0.25
	ABS		Absolute Value Syntax: ABS(expr) or ABS(matrix) For numerical arguments, returns the absolute value of the expression. For matrix arguments, returns the Frobenius (Euclidean) norm of the array. Examples: ABS(-3.14) → 3.14 ABS([[1,2],[3,4]]) → 5.47722557505 ABS(2-3*i) → 3.60555127546 CAS(ABS([[1,2],[3,4]])) → √30
	ACOS		Inverse Cosine Syntax: ACOS(Value) Returns the inverse cosine of Value. The output depends on the Angle Measure setting in Home Settings, CAS Settings, or Symbolic Setup. Example: ACOS(0.5) → 60 (Degrees mode)

Help Topics Tree 13217	Help Text
	ACOS(0.833730025131-0.988897705763*i) → 1+i ACOS({0.5,1}) → {60,0} (Degrees mode)
Ans	Last Answer Syntax: Ans In Home view, Ans returns the result of the last calculation made in Home view to its full precision. The variable Ans is different from the numbers in Home's history. A value in Ans is stored internally with the full precision of the calculated result, whereas the displayed numbers match the display mode. Ans(n) returns the nth entry in the Home view history. In CAS view, Ans returns the last result in the CAS history and Ans(n) does not recall the nth item in history. Here, Ans(n) will attempt to substitute n for x (or the default variable) in the last item in history and return the result. In CAS view, if Ans is a matrix, Ans(m,n)returns the element in row m and column n.
ASIN	Inverse Sine Syntax: ASIN(Value) Returns the inverse sine of Value. The output depends on the Angle Measure setting in Home Settings, CAS Settings, or Symbolic Setup. Example: ASIN(1) → 90 (Degrees mode) ASIN(1.29845758142+0.634963914785*i) → 1+i ASIN({0.5,1}) → {30,90} (Degrees mode)
ATAN	Inverse Tangent Syntax: ATAN(Value) Returns the inverse tangent of Value. The output depends on the Angle Measure setting in Home Settings, CAS Settings, or Symbolic Setup. Example: ATAN(1) → 45 (Degrees mode) ATAN(0.27175258532+1.08392332734*i) → 1+i ATAN({1,0}) → {45,0} (Degrees mode)
COS	Cosine Function Syntax: COS(Value) Returns the cosine of Value. Value is interpreted as radians, degrees or gradians, depending on the setting of Angle Measure in Home Settings, CAS Settings, or Symbolic Setup. Example: COS(60) → 0.5 (Degrees mode) COS(1+i) → 0.833730025131-0.988897705763*i COS({60,0}) → {0.5,1} (Degrees mode) COS((π/3)_rad) → 0.499999999997
LN	Natural Logarithm Syntax: LN(value) Natural Logarithmic function Returns the logarithm of value in base e, Euler's number. Examples: LN(1) → 0 LN(2+3*i) → 1.28247467873+0.982793723247*i LN({0.1,1}) → {−2.30258509299,0}
LOG	General Logarithm Syntax: LOG(value, [base]) General logarithmic function Returns the logarithm of value in base. By default, base=10. Examples: LOG(8) → 0.903089986992 LOG(8,2) → 3 LOG(2+3*i) → 0.556971676153+0.426821890855*i LOG(2+3*i,2) → 1.85021985907+1.41787163075*i LOG({100,10}) → {2,1} LOG({8,27,10000},{2,3,10}) → {3,3,4}
+/-	Negative Syntax: - Value or - Expression Unary minus. Changes the sign of Value or Expression. Used to enter negative numbers.

Help Topics Tree	13217	Help Text
	$^{n}\sqrt{}$	Nth Root Key Syntax: Value1 √ Value2 NTHROOT: the nth root function This Shift-key combination brings up a template for the NTHROOT function. It returns the primary Value1 root of Value2. On the keyboard, NTHROOT is represented by $^{n}\sqrt{}$. Examples: 3 NTHROOT 8 → 2 3 NTHROOT 79.507 → 4.3 2.3 NTHROOT 5413.44050218 → 42 2.1 NTHROOT 3+2*i → 1.76999848019+0.508973095403*i (1.2-0.5*i) NTHROOT (0.2+4*i) → 0.137162958212+1.70241905473*i 3 NTHROOT {27,8,64} → {3,2,4}
	SIN	Sine Syntax: SIN(Value) Returns the sine of Value. Value is interpreted as radians, degrees or gradians, depending on the setting of Angle Measure in Home Settings, CAS Settings, or Symbolic Setup. Example: SIN(30) → 0.5 (Degrees mode) SIN(1+i) → 1.29845758142+0.634963914785*i SIN({30,90}) → {0.5,1} (Degrees mode) SIN((π/6)_rad) → 0.5
	Math Template	The Math Template key displays a palette of pre-formatted templates representing common mathematical operations and expressions, such as nth root, differentiation, and integration (as well as vectors, matrices, and sexagesimal numbers). Either tap on the template you want, or user the rocker wheel to highlight the template and press Enter. The template is copied to the entry line ready for you to add in the values. Press Esc to close the palette without making a selection.
	Math Chars	Press Shift 9 to open a menu of commonly used mathematical symbols and Greek characters. All of these characters can also be found by pressing Shift Vars (Chars).
Integer Arithmetic		You can perform 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. Indicate that you are about to engage in integer arithmetic by preceding the number with the pound symbol: # (press Alpha 3). Then indicate what base to use for the number by appending the appropriate base marker. d: decimal b: binary o: octal h: hexadecimal [blank]: default base Examples: #423:d #01101010:b #6537:o #CFF0:h #B1D4
	Change Integer Base	The calculator’s default base for integer arithmetic is 16 (hexadecimal). To change the default base: 1. Press Shift Home (Home settings). 2. Choose the base you want from the Integers menu: Binary, Octal, Decimal, or Hex. 3. The field to the right of the Integers menu is the word size field. This is the maximum number of bits that can represent an integer. The default value is 32, but you can change it any value between 1 and 64. 4. If you want to allow for signed integers, select the ± option to the right of the word size field. Choosing this option reduces the maximum size of an integer to one bit less than the word size.
	Manipulate Integers	The Base key enters the character # in the edit field, unless the currently selected item (or most recent result) is an integer, in which case the Edit Integer screen is displayed. In the Edit Integer dialog box, the Was field at the top shows the original integer you selected in Home view. The Out field shows the edited integer. Both integers are initially displayed in the default base as specified in Home Settings Page 1. The 16 field is the hexadecimal representation of Out. The 10 field is the decimal representation of Out. The box below the decimal value shows the 64 bit binary (bit) representation of Out. Changing the value by any of the following methods updates the value of Out and all representations: • Shift the integer left or right. Bits shifted off either end are lost. - Drag the screen left or right to shift by 1 bit in the corresponding direction

Help Topics Tree 13217	Help Text
	- Rocker wheel left/right to shift 1 bit
	- Alpha rocker wheel right/left to shift by 4 bits (1 nibble)
	- Shift rocker wheel right/left shifts by 8 bits (one byte)
	• Change word size (number of bits) of the integer
	- Drag the screen down/up to increase/decrease by 1
	- Press rocker wheel up/down to increase/decrease by 1
	- Alpha rocker wheel up/down to increase/decrease by 4 (1 nibble)
	- Shift rocker wheel up/down to increase/decrease by 8 (one byte)
	• Change sign of the integer: Press +/- key
	• Cycle through base settings (system, hex, decimal, octal and binary): + and – keys
	Menu Buttons:
	• Reset: returns all changes to their original state
	• Base: cycles through the bases; same as pressing +
	• Signed: toggles the word size between signed and unsigned
	• NOT: returns the one's complement (that is, each bit in the specified word size is inverted: a 0 is replaced by 1 and a 1 by 0.
	• Edit: toggles edit mode. Edit mode is active if a bullet character appears on the Edit button. When Edit mode is active, a digit selector highlightes a single digit and you can move abut the dialog using the rocker wheel. The hex and decimal fields can be modified, as can the bit representation, one digit at a time. A change in any 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.
	Validate the change in the number using OK or cancel using the Esc key.
List Catalog	Press Shift 7 (List) to see the List Catalog. There are ten lists available, named L1-L9 and L0.
	Use the rocker wheel up/down to select a list name.
	Tap Edit or press Enter to edit the selected list.
	Menu Buttons:
	• Edit: opens the selected list for editing in the List Editor
	• Delete: deletes the contents of the highlighted list
	• Send: when present, sends a list to another HP Prime
List Editor	The List Editor is designed to help you create and edit any of the lists available on the HP Prime. The default lists are named L1-L9 and L0.
	When you select a list, the List Editor opens. This is where you add elements to, or change elements in, a list. When you first open a list, it will be blank. To enter an element, just start keying it. The menu items in the List Editor are:
	• Edit: copies the selected element to the entry line where it can be edited. This item is only visible when an element in the list is selected.
	• More: opens a menu with options for editing the list
	• Go To: jumps to a specific element in the list. Useful for very large lists.
	• Go: toggles how the cursor moves when you press Enter. The options are Down, Right, and None.
	The List Editor More Menu
	The List Editor More menu contains the following options for editing a list:
	• Insert
	• Row: Inserts a new row in the current list. The new row contains 0 as its element.
	• Delete
	• Column: Deletes the contents of the current list. To delete a single element, select it and press the Delete key.
	• Select
	• Row: Selects the current row. Once selected, the row can be copied.
	• Column: Selects the current list. Once selected, the list 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 on a cell to start selection, then drag to select a rectangular array of elements. Once selected, the array can be copied.
	• Swap Ends: Swaps the starting and ending cells for the selected rectangular array of cells.
	• Selection: Toggles selection mode on and off. You can also tap and hold on a cell, then drag to select.
	• Swap
	• Column: Swaps the contents of two columns (lists).
	Type in the first entry in the list and press the Enter key. Continue until you have completed the list. When you have completed the list you can return to the List Catalog.
	In programs or Home view, you can reference your list by name (L1, L2, etc.) to perform operations on your new list. Use the rocker wheel left/right to scroll through all ten lists once you are in the List Editor.
Matrix Catalog	Press Shift 4 (Matrix) to enter the Matrix Catalog.
	There are ten matrices available, named M1-M9 and M0.
	Menu Buttons:
	• Edit: opens the selected matrix for editing in the Matrix Editor
	• Delete: deletes the contents of the selected matrix
	• Vect: changes a matrix into a vector
	• Send: when present, sends a matrix to another HP Prime

Help Topics Tree	13217	Help Text
	Matrix Editor	When you select a matrix, the Matrix Editor opens. The menu items in the Matrix Editor are: • Edit: copies the selected element to the entry line where it can be edited. This item is only visible when an element in the matrix or vector is selected. • More: opens a menu with options for editing the matrix • Go To: jumps to a specific element in the matrix. Useful for large matrices. • Go: toggles how the cursor moves when you press Enter. The options are Down, Right, and None. The Matrix Editor More Menu The Matrix Editor More menu contains options similar to those in the List Editor More menu, but with an expanded offering appropriate to editing matrices. The options are as follows: • Insert • Row: Inserts a new row in the current matrix, above the current row. The new row contains zeroes. • Column: Inserts a new column in the matrix, to the left of the current column. 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. Once selected, the row can be copied. • Column: Selects the current column. Once 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 on a cell to start selection, then drag to select a rectangular array of elements. Once selected, the array can be copied. • Swap Ends: Swaps the beginning and ending cells for the selected array of cells. • Selection: Toggles selection mode on and off. You can also tap and hold on a cell, then drag to select. • Swap • Row: Swaps the contents of two rows. • Column: Swaps the contents of two columns. You do not have to define the dimensions of a matrix first; just start typing in values. You can enter values row by row, or column by column; the Go button toggles through the options. As with a list, you can send a matrix to another HP Prime or receive one from another HP Prime. In programs or the Home view, you can reference these matrices by name to perform operations on them.
Program Catalog		Press Shift 1 (Program) to enter the Program Catalog. In the Program Catalog, you can view the list of existing programs, edit or run any of them, or send them to another HP Prime. You can also create a new program. An HP Prime program can be as simple as a single user-defined function, or it can contain a set of related functions that are exported to show up as a submenu in the User menu of the Toolbox menus. These functions can have their variables exported to show up in the User menu under the Vars key or kept local. An HP Prime program could also be a full-blown application in its own right. The choice is up to you. Menu Buttons: • Edit: opens the highlighted program for editing in the Program Editor • New: prompts for a new program name, then opens the Program Editor • More: opens more menu options (Save, Rename, Sort, Delete, and Clear) for the highlighted program • Send: sends a program to another HP Prime that supports unit-to-unit connectivity • Debug: debugs the highlighted program • Run: runs the highlighted program Tap New to create a new program and enter a name for the program. Use the rocker wheel up/down to select an existing program. Tap Edit to start the Program Editor and edit an existing program. 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, RollDice, and Spin2 are valid program names.
	Program Editor	Once you enter your program name and tap OK, you enter the Program Editor. Here, a template for your program is created. The template consists of a heading for a function with the same name as the program, and a BEGIN–END pair that blocks off the statements for the function. Menu Buttons: • Cmds: opens a menu from which you can choose from common programming commands • Tmplt: opens a menu from which you can choose from templates for common programming control structures • ▲ Page ▼: moves from page to page in a multi-page program • Check: checks the program for syntax errors If you press the Menu key while in the Program Editor, you will see two new options: • Create user key: tap this option and then press any key to paste a template into your program for redefining that key as a user key. • Insert pragma: tap this option to paste a #pragma mode definition into your program. The #pragma mode definition is of the following form: #pragma mode(separator(), integer())

Help Topics Tree 13217	Help Text
	Use the #pragma mode definition to define the set of separators used for digit grouping and the integer type. The #pragma mode definition will force 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. You can type in your program letter-by-letter if you know the command names, or use the Cmds or Tmplt menus. You can also press the Toolbox key see the Toolbox menus to see more functions and commands. Once you have finished your program, you can return to the Program Catalog by pressing Shift 1 (Program). From the catalog, you can send your program to another HP Prime or receive a program from another HP Prime. Enter your program name in the Home view and press Enter to run your program. Note: the Editor saves your changes automatically when you exit the editor. If you want to save the original version of your program before you make changes, be sure to use the More button in the Program Catalog and select Save.
Commands Menu	The Commands (Cmds) menu contains the programming commands for the HP Prime. This main menu contains the command categories, and the sub-menus contain the specific commands within each category. Use the rocker wheel to select a category in the main menu and then right rocker wheel to the sub-menu of commands within that category. Use the rocker wheel left/right to select the command you want within a category. An HP Prime program contains a sequence of commands that execute automatically to perform a task. 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, command arguments are optional. If an optional argument is omitted, a default value is used in its place. Optional command arguments appear inside [square brackets]. Note that square brackets are also used for vectors, which are usually not optional! 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 and 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. A function can return a value using the RETURN command.
String	String Commands This menu contains all the string manipulation commands. A string is a set of characters enclosed in double quotes; for example, "ABC", "12A", and "3-A" are all strings. The \ character starts an escape sequence, and the character(s) immediately following are interpreted specially, as described below. • "": " • \": " • \\: \ • \0: 0 • \n” new line • \r: carriage return • \t: tab • \nnn: octal, as in \115 for character 77 • \xnn: hex, as in \x45 for character 69 • \unnnn: Unicode for character nnnn To put a new line into a string, you can also press Enter to wrap the text at that point.
REPLACE	Syntax: REPLACE(Object1,Start,Object2) Replaces portion of a matrix, vector or string (Object1) starting at Start with Object2. For a matrix, Start is a list containing two numbers; for a vector or string it is a single number. Note: for strings, you can do: REPLACE("string", "sub_string", "replace_string") Examples: REPLACE([[1,2,3],[4,5,6]],{1,2},[[8,8],[9,9]]) → [[1,8,8],[4,9,9]] REPLACE([10,12,23],3,[9,8,7,6]) → [10,12,9,8,7,6] REPLACE("Replacement","place","configure") → "Reconfigurement" L1:={8,9,7,3,4,6,6,8,0}; REPLACE(L1,5,{1,2,3,4,5}) → {8,9,7,3,1,2,3,4,5}

ASC	Syntax:
	ASC(String)
	Returns a list containing the numerical Unicode values of String.
	Examples:
	ASC("AB") → {65,66}
	ASC("ʥ♘♕♤♦☎") → {677,9816,9813,9828,9830,9742}
	ASC({"HE","LLO"}) → {{72,69},{76,76,79}}

CHAR	Syntax: CHAR(List) or CHAR(Vector) or CHAR(Integer) Returns the string corresponding to the numerical Unicode character codes in List or Vector, or the numerical Unicode character code of Integer. Examples: CHAR(65) → "A"

Help Topics Tree	13217	Help Text
		CHAR({82,77,72}) → "RMH" CHAR({#261Eh,#265Eh,#266Ch,#266Dh,#2680h,#2685h}) → "☞♞♬♭⚀ ⚅"
DIM		String Dimensions Syntax: DIM(String) or DIM(Matrix) Returns the number of characters in String or the dimensions of Matrix. Examples: DIM("12345") → 5 DIM([[1,2],[4,5],[7,8]]) → {3,2} DIM({"12345","HP Prime"}) → {5,8}
EXPR		Evaluate String Syntax: EXPR(String) Parses a string into a number or expression and returns the result evaluated. Examples: EXPR("2+3") → 5 X:=90; EXPR("X+10") → 100 X:=90; Y:=3; EXPR({"X/2","2^Y"}) → {45,8}
INSTRING		In String Syntax: INSTRING(String1, String2) Returns the index of the first occurrence of String2 in String1. Returns 0 if String2 is not present in String1. Note that the first character in a string is position 1. Examples: INSTRING("vanilla", "van") → 1 INSTRING("banana","na") → 3 INSTRING("ab","abc") → 0 INSTRING({"vanilla","banana","ab"},{"van","ana","abc"}) → {1,2,0}
LEFT		Left Part Syntax: LEFT(String, Integer) Given a string and an integer n, return the first n characters of the string. If n ≥ DIM(str) or n ≤ 0, returns the entire string. Example: LEFT("MOMOGUMBO",3) → "MOM"
LOWER		Lowercase Syntax: LOWER(string) Returns string with uppercase characters converted to lowercase. Examples: LOWER("ABC") → "abc" LOWER("ΑΒΓ") → "αβγ"
MID		Middle Syntax: MID(String, Position, [n]) Extracts n characters from String starting at Position. If n is not specified, then MID extracts the remainder of String from Position. Examples: MID("MOMOGUMBO",3,5) → "MOGUM" MID("PUDGE",4) → "GE"
STRINGFROMID		String From Identifier Syntax: STRINGFROMID(Integer) Returns the built-in string associated with the ID of the current language. Example: STRINGFROMID(1)
ROTATE		Syntax: ROTATE(String, n) ROTATE(grob, angle, [bg_color]) ROTATE([DestGrob], angle, SrcGrob, [dest_point]) ROTATE([DestGrob], SrcGrob, dest_point_1, dest_point_2, dest_point_3, dest_point_4, [src_point_1, src_point_2, src_point_3, src_point_4]) The string form of ROTATE moves n characters from the beginning or end of String to the opposite end of String, depending on the sign of n. If n is positive, takes the first n characters of String and put them on the right of String. If n is negative, takes the last n characters and put them on the left of String. If ABS(n)>dim(string), returns String. The graphical forms of ROTATE use an angle or sets of points to rotate a graphic object (grob).

Help Topics Tree	13217	Help Text
		ROTATE(grob, angle, [bg_color]) Rotate grob around its center by angle. grob will be resized to accommodate the extra space needed and that extra space will be filled by bg_color. If bg_color is not specified, the current background color is used. ROTATE([DestGrob], angle, SrcGrob, [dest_point]) Draw SrcGrob, rotated by angle, on DestGrob with the center of SrcGrob at position dest_point (specified in pixels as a list of 2 numbers or a single complex number). If DestGrob is not specified, G0 is used. If dest_point is not specified, the center of DestGrob is used. ROTATE([DestGrob], SrcGrob, dest_point_1, dest_point_2, dest_point_3, dest_point_4, [src_point_1, src_point_2, src_point_3, src_point_4]) Note: src_points and dest_points are specified in pixels as lists of 2 numbers or as complex numbers. If src_points are not specified, then src_point_1 is set to to the top left corner of SrcGrob, src_point_2 is set to to the top right corner of SrcGrob, src_point_3 is set to to the bottom right corner of SrcGrob, and src_point_4 is set to to the bottom left corner of SrcGrob. Draws the part of SrcGrob specified by the 4 src_points in the area of DestGrob specified by the 4 dest_points. This is done internally by subdividing the work into 2 triangles (points1, 2 and 3 and points 1, 3 and 4). Therefore non homogenous coordinates can yield to different stretching on both triangles. It is possible to have point_1=point_2 to only work with triangles. Examples: ROTATE("12345",2) → "34512" ROTATE("12345",-1) → "51234" ROTATE("12345",6) → "12345" Demo_ROTATE
RIGHT		Right Part Syntax: RIGHT(String, n) Returns the last n characters of the string. Example: RIGHT("MOMOGUMBO",5) → "GUMBO"
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 Mode, Precision, and Separator parameters specify how numbers are displayed. If Mode is specified, it is: 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 can be a number. -1 means use default, 0 to 10 correspond to the 11 built-in digit grouping choices available in home settings. OR Separator can be a string containing a set of digits and separators. The last digit is assumed to be the one just before the decimal point. "[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 between -1 and 1 are displayed without a leading zero (for example, .1 instead of 0.1) If SizeLimit is specified, the command will attempt to generate a string that fits in the given number of pixels. FontSize is used along with Bold, Italic, and Monospaced (if their value is non-zero) to estimate the maximum string length that will fit. The values for FontSize are: 0=current font (default) 1=font 10 2=font 12 (Small) 3=font 14 (Medium) 4=font 16 (Large) 5=font 18 6=font 20 7=font 22 Examples: Current number format setting Standard: STRING(3*π) → "9.42477796077"

Help Topics Tree	13217	Help Text
		Number format Fixed, 4 decimal places: STRING(3*π,2,4) → "9.4248"
UPPER		Uppercase Syntax: UPPER(string) Returns string with lowercase characters converted to uppercase. Examples: UPPER("abc") → "ABC" UPPER("αβγ") → "ΑΒΓ"
Drawing		Drawing Commands This menu contains commands for creating simple graphic objects, such as line segments, in graphic variables. There are 10 graphic variables in the HP Prime, called G0 to G9. G0 is always the current screen graphic and is used by default if no GROB is specified for any of the drawing commands. G1 to G9 can be used to store temporary graphic objects (called GROBs for short) when programming applications that use graphics. Variables G1 to G9 are temporary and are cleared when the calculator turns OFF. There are two identical sets of functions that can be used to modify graphic variables. The first set of them work based on Cartesian coordinates using the Cartesian plane defined in the current app by the variables Xmin, Xmax, Ymin, and Ymax in the Plot setup. The rest work on absolute pixel references based on the physical display of the HP Prime. For these functions, (0,0) is the top left pixel of the GROB, and (320,1240) is the bottom right. This second set of functions—those that work with pixels—have a _P suffix attached to their name, as in ARC_P and LINE_P. In many of the commands, the color used can be specified as well. Unless otherwise specified, colors are defined in #A8R8G8B8 format (8 bits for R, G, B and A). It is highly recommended to use the RGB function when defining colors to provide compatibility for future devices in your programs. If color is not specified for a drawing command, it will default to black unless otherwise specified.
PX→C		Syntax: PX→C(x, y) or PX→C({x, y}) Transform pixel coordinates into Cartesian coordinates. Returns a list. Examples: PX→C(319,219) → {15.9,−10.9} (assuming current app Plot Settings are set to default) PX→C({320,0}) → {16,10.9} (assuming current app Plot Settings are set to default)
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 components values (0 to 255). The Alpha Channel number A runs from 0 (opaque) to 255 (transparent).
Pixels		Pixel Commands The commands for drawing using pixel coordinates are listed in this section.
ARC_P		Draw Arc Syntax: ARC_P(G, x, y, r or {rx, ry}, [∡1, ∡2], [border_color, [fill_color]]) Draws a circle on GROB G, centered at (x,y), with radius r (in pixels). If r is replaced by a list {rx, ry} then the Arc becomes an ellipse centered at (x,y) with radius in the x dimension of rx and in the y dimension of ry. If ∡1 and ∡2 are specified, draws an arc from ∡1 to ∡2 using the current angle mode. Example: Demo_ARC_P
BLIT_P		Copy GROB Syntax: BLIT_P([trgtG], [dx1, dy1], [dx2, dy2], srcG, [sx1, sy1], [sx2, sy2], [c], [alpha]) Copies the region of graphic srcG between point (sx1, sy1) and (sx2, sy2) into the region of trgtG between points (dx1, dy1) and (dx2, dy2). Pixels from srcG that are color c are not copied. alpha is a number from 0 (transparent) to 255 (opaque) which represent the transparency (alpha channel) of the source bitmap. The defaults for the optional arguments are: trgtG = G0 srcG = G0 sx1, sy1 = srcGRB top left corner sx2, sy2 = srcGRB bottom right corner dx1, dy1 = trgtGRB top left corner dx2, dy2 = calculated so destination area is the same as source area c = all pixel colors alpha= 255 (fully opaque) Note: when using the c and alpha options, it is highly recommended to specify the source x/y coordinates in order to make sure that the system can distinguish what each parameter is. Example: Demo_BLIT_P
DIMGROB_P		Size GROB

Help Topics Tree					13217	Help Text
						Syntax: DIMGROB_P(G, w, h, [color]) or DIMGROB_P(G, w, h, list) Sets the dimensions of GROB G to w*h. Initializes the graphic G with color or with the graphic data provided in list. 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 (1 bit for alpha channel and 5 bits for R, G and B). Example: Demo_DIMGROB_P
					GETPIX_P	Get Pixel Color Syntax: GETPIX_P([G], x, y) Returns the color of the pixel of G with pixel coordinates (x, y). Examples: Demo_GETPIX_P
					GROBH_P	GROB Height Syntax: GROBH_P(G) Returns the height of the graphic object G in pixels. Example: GROBH(G0) → 240
					GROBW_P	GROB Width Syntax: GROBW_P(G) Returns the width of the graphic object G in pixels. Example: GROBW_P(G0) → 320
					INVERT_P	Invert GROB Syntax: INVERT_P([G, x1, y1, x2, y2]) Inverts the rectangle on G defined by the diagonal points (x1, y1) and (x2, y2). The effect is reverse video. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G If only one (x,y) pair is specified, it refers to the top left corner of G. Example: Demo_INVERT_P
					FILLPOLY	Draw Filled Polygon Syntax: FILLPOLY([G], {Coordinates}, Color, [Alpha]) FILLPOLY([G], [Coordinates], Color, [Alpha]) Fills the polygon specified by the provided Cartesian coordinates using the color provided. If Alpha (0 to 255) is provided, the polygon is drawn with transparency. Examples: FILLPOLY([(0,0),(1,1),(2,0),(3,-1),(2,-2)],#FF,128) Demo_FILLPOLY
					FILLPOLY_P	Draw Filled Polygon Syntax: FILLPOLY_P([G], {Coordinates}, Color, [Alpha]) FILLPOLY_P([G], [Coordinates], Color, [Alpha]) Fills the polygon specified by the provided pixel coordinates using the color provided. If Alpha (0 to 255) is provided, the polygon is drawn with transparency. Examples: FILLPOLY_P([(20,20),(120,120),(150,20),(180,150),(50,100)],#FF,128) Demo_FILLPOLY_P
					LINE_P	Line Drawing Syntax: LINE_P([G], x1, y1, x2, y2, [color]) LINE_P([G],points_definition, lines_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) LINE_P([G],pre_rotated_points, line_definitions, [zstring]) The basic form of LINE_P draws one line between specified coordinates in the graphic using the specified color. The advanced form of LINE_P allows the rendering of multiple lines at a time with a potential 3D transformation of the points that define the line. This is mostly used if you have a set of vertices and lines and want to display them all at once (faster).

Help Topics Tree	13217	Help Text
		points_definition is either a list or a matrix of point definitions. Each point is defined by 2 to 4 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). lines_definitions is either a list or a matrix of line definitions. Each line is defined by 2 to 4 numbers. p1, p2, color and alpha. p1 and p2 are the index in the points_definition of the 2 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 re-specifying the same color for each line. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the points using the usual 3D or 4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed objects. Each point is rotated and translated through a multiplication by rotation_matrix. It is then projected on the view plane using the eye position using 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 provided easier zClipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). LINE_P returns a string which contains all the transformed points. If you plan to call TRIANGLE_P or LINE_P multiple times in a row using the same points and transformation, you can do so by replacing the points_definition by this string and omitting the transformation definition in subsequent calls to TRIANGLE_P and LINE_P. 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 vertexes and LINE_P will not draw pixels further than the already drawn pixels. ZString is automatically updated as appropriate. Example: Demo_LINE_P
	PIXOFF_P	Pixel Off Syntax: PIXOFF_P([G], x, y) Sets the color of the pixel of GROB G with coordinates (x, y) to white.
	PIXON_P	Pixel On Syntax: PIXON_P([G], x, y, [color]) Sets the color of the pixel of GROB G with coordinates (x, y). If supplied, color is a hexadecimal integer of the form aaRRGGBB. This is an RGB color with the Alpha Channel in the high order byte. The Alpha Channel number runs from 0 (opaque) to 255 (transparent). If no color is specified, black is used. Examples: PIXON_P(50,50,RGB(255,0,0)) PIXON_P(50,50,RGB(255,0,0,128))
	RECT_P	Rectangle Syntax: RECT_P([G], [x1, y1], [x2, y2], [Color]) RECT_P([G], [x1, y1], [x2, y2], [edgeColor],[fillColor]) Draws a rectangle on G, with diagonal defined by points (x1,y1) and (x2,y2), using edgeColor for the perimeter and fillColor for the inside. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G edgeColor=white fillColor=edgeColor Note: To erase a GROB, execute RECT_P(G). To clear the screen, execute RECT_P(). Note: semi-transparent rectangles can be drawn by using the Alpha channel in the color (0 is opaque, 255 is transparent). The color can also be expressed as { color, alpha }. Example: Demo_RECT_P
	SUBGROB_P	Copy GROB to Target Syntax: SUBGROB_P(srcG, [x1, y1], [x2, y2], trgtG) Sets graphic trgtG to be a copy of the area of srcG between points (x1,y1) and (x2,y2). If both (x1, y1) and (x2, y2) are not specified, then the entire graphic srcG is used. If (x1, y1) is not specified, then the top left corner of srcG is used; if (x2, y2) is not specified, then the bottom right corner of srcG is used. trgtGRB can be any of the graphic variables except G0. SUBGROB_P(G1, G4) will copy G1 in G4. Example:

Help Topics Tree	13217	Help Text
		Demo_SUBGROB_P
	TEXTOUT_P	Draw Text Syntax: TEXTOUT_P(text, [G], x, y, [font], [textColor], [width], [backgroundColor]) Draws text on graphic G at position (x, y) using font and textColor. Paints the background before drawing the text using color backgroundColor. If width is specified, does not draw text more than width pixels wide. If backgroundColor is not specified, the background is not erased. The sizes for font are: 0=current font (default) 1=font 10 2=font 12 (Small) 3=font 14 (Medium) 4=font 16 (Large) 5=font 18 6=font 20 7=font 22 Returns the X coordinate at which the next character of the string should be drawn if the string had more characters Examples: TEXTOUT_P("Hello HP Prime",100,100,4,RGB(255,0,0),200,RGB(0,255,255)); FREEZE Demo_PISERIES_P
	TRIANGLE_P	Draw Triangle Syntax: TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha]) TRIANGLE_P([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3]) TRIANGLE_P([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"]) TRIANGLE_P([G], points_definition, triangle_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) TRIANGLE_P([G], pre_rotated_points, triangle_definitions, [zstring]) TRIANGLE_P([G]) The basic form of TRIANGLE_P draws one triangle between specified pixel coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If 3 colors are specified, blends the colors in between the vertexes. The advanced form of TRIANGLE_P allows the rendering of multiple triangles at a 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 2 to 4 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)… triangle_definitions is either a list or a matrix of triangle definition. Each triangle is defined by 3 to 5 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 re-specifying the same color for each triangle. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the point using usual 3/4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed 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 using 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 provided easier z clipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). TRIANGLE_P returns a string which contains all the transformed points. If you plan to call TRIANGLE_P or LINE_P multiple times in a row using the same points and transformation, you can do so by replacing the points_definition by this string and omitting the transformation definition in subsequent calls to TRIANGLE_P and LINE_P. 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 TRIANGLE_P with appropriate z (0-255) values for each of the triangle vertexes and TRIANGLE_P will not draw pixels further than the already drawn pixels. zstring is automatically updated as appropriate. Examples: TRIANGLE_P(0,20,150,50,100,100,#FFh,#FF00h,#FF0000h,128); FREEZE Demo_TRIANGLE_P Demo_Tetrahedron_P
Cartesian		Cartesian Drawing Commands

Help Topics Tree	13217	Help Text
		The commands for drawing using Cartesian coordinates are listed in this section.
	TRIANGLE	Draw Triangle Syntax: TRIANGLE([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha]) TRIANGLE([G], x1, y1, x2, y2, x3, y3, c1, [c2, c3], [Alpha], ["ZString", z1, z2, z3]) TRIANGLE([G], {x1, y1, [c1], [z1]}, {x2, y2, [c2], [z2]},{x3, y3, [c3], [z3]}, ["ZString"]) TRIANGLE([G], points_definition, triangle_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) TRIANGLE([G], pre_rotated_points, triangle_definitions, [zstring]) TRIANGLE([G]) The basic form of TRIANGLE draws one triangle between specified pixel coordinates in the graphic using the specified color and transparency (0 ≤ Alpha ≤ 255). If 3 colors are specified, blends the colors in between the vertexes. The advanced form of TRIANGLE allows the rendering of multiple triangles at a 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 2 to 4 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)… triangle_definitions is either a list or a matrix of triangle definition. Each triangle is defined by 3 to 5 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 re-specifying the same color for each triangle. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the point using usual 3/4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed 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 using 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 provided easier z clipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). TRIANGLE 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 by this string and omitting the transformation definition in subsequent calls to TRIANGLE and LINE. About zstring TRIANGLE([G]) returns a string adapted for z clipping. To use Z clipping, call TRIANGLE to create a Z clipping string (initialized at 255 for each pixels). You can then call TRIANGLE with appropriate z (0-255) values for each of the triangle vertexes and TRIANGLE will not draw pixels further than the already drawn pixels. zstring is automatically updated as appropriate. Examples: TRIANGLE(0,0,5,5,5,-5,#FFh,#FF00h,#FF0000h,128); FREEZE Demo_TRIANGLE Demo_Tetrahedron
	ARC	Draw Arc Syntax: ARC(G, x, y, r or {rx, ry}, [∡1, ∡2], [border_color, [fill_color]]) Draws a circle on GROB G, centered at (x,y), with radius r (in pixels). If r is replaced by a list {rx, ry} then the Arc becomes an ellipse centered at (x,y) with radius in the x dimension of rx and in the y dimension of ry. If ∡1 and ∡2 are specified, draws an arc from ∡1 to ∡2 using the current angle mode. Example: Demo_ARC
	BLIT	Copy GROB Syntax: BLIT([trgtG], [dx1, dy1], [dx2, dy2], srcG, [sx1, sy1], [sx2, sy2], [c], [alpha]) Copies the region of graphic srcG between point (sx1, sy1) and (sx2, sy2) into the region of trgtG between points (dx1, dy1) and (dx2, dy2). Pixels from srcG that are color c are not copied. alpha is a number from 0 (transparent) to 255 (opaque) which represent the transparency (alpha channel) of the source bitmap. The defaults for the optional arguments are: trgtG = G0 srcG = G0 sx1, sy1 = srcGRB top left corner sx2, sy2 = srcGRB bottom right corner dx1, dy1 = trgtGRB top left corner dx2, dy2 = calculated so destination area is the same as source area

Help Topics Tree	13217	Help Text
		c = all pixel colors alpha= 255 (fully opaque) Note: when using the c and alpha options, it is highly recommended to specify the source x/y coordinates in order to make sure that the system can distinguish what each parameter is. Example: Demo_BLIT
	DIMGROB	Size GROB Syntax: DIMGROB(G, w, h, [color]) or DIMGROB(G, w, h, list) Sets the dimensions of GROB G to w*h. Initializes the graphic G with color or with the graphic data provided in list. 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 (1 bit for alpha channel and 5 bits for R, G and B). Example: Demo_DIMGROB
	GETPIX	Get Pixel Color Syntax: GETPIX([G], x, y) Returns the color of the pixel of G with Cartesian coordinates (x, y). Examples: Demo_GETPIX
	GROBH	GROB Height Syntax: GROBH(G) Returns the height of the graphic object G. Example: GROBH(G0) → 24
	GROBW	GROB Width Syntax: GROBW(G) Returns the width of the graphic object G. Example: GROBW(G0) → 32
	INVERT	Invert GROB Syntax: INVERT([G, x1, y1, x2, y2]) Inverts the rectangle on G defined by the diagonal points (x1, y1) and (x2, y2). The effect is reverse video. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G If only one (x,y) pair is specified, it refers to the top left corner of G. Example: Demo_INVERT
	LINE	Line Drawing Syntax: LINE([G], x1, y1, x2, y2, [color]) LINE([G],points_definition, lines_definitions, rotation_matrix or {rotation_matrix or -1, ["N"], [{eye_x, eye_y, eye_z} or -1], [{xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D}]}, [zstring]) LINE([G],pre_rotated_points, line_definitions, [zstring]) The basic form of LINE draws one line between specified coordinates in the graphic using the specified color. The advanced form of LINE allows the rendering of multiple lines at a time with a potential 3D transformation of the points that define the line. 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 2 to 4 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). lines_definitions is either a list or a matrix of line definitions. Each line is defined by 2 to 4 numbers. p1, p2, color and alpha. p1 and p2 are the index in the points_definition of the 2 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 re-specifying the same color for each line. rotation_matrix is a matrix of sizes 2*2 to 3*4 which specifies the rotation and translation of the points using the usual 3D or 4D geometry. {eye_x, eye_y, eye_z} defines the eye position (projection). {xmin3D, xmax3D, ymin3D, ymax3D, zmin3D, zmax3D} is used to perform 3D clipping on the pre-transformed objects.

Help Topics Tree	13217	Help Text
		Each point is rotated and translated through a multiplication by rotation_matrix. It is then projected on the view plane using the eye position according to 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 provided easier zClipping. If zstring is provided, per pixel z clipping will happen using the z value string (see below). LINE 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 by this string and omitting the transformation definition in subsequent calls to TRIANGLE and LINE. About ZString TRIANGLE([G]) returns a string adapted for z clipping. To use Z clipping, call TRIANGLE to create a Z clipping string (initialized at 255 for each pixels). You can then call LINE with appropriate z (0-255) values for each of the triangle vertexes and LINE will not draw pixels further than the already drawn pixels. ZString is automatically updated as appropriate. Example: Demo_LINE
	PIXOFF	Pixel Off Syntax: PIXOFF([G], x, y) Sets the color of the pixel of GROB G with coordinates (x, y) to white.
	PIXON	Pixel On Syntax: PIXON([G], x, y, [color]) Sets the color of the pixel of GROB G with coordinates (x, y). If supplied, color is a hexadecimal integer of the form aaRRGGBB. This is an RGB color with the Alpha Channel in the high order byte. The Alpha Channel number runs from 0 (opaque) to 255 (transparent). If no color is specified, black is used. Examples: PIXON(0,0,RGB(255,0,0)) PIXON(0,0,RGB(255,0,0,128))
	RECT	Draw Rectangle Syntax: RECT([G], [x1, y1], [x2, y2], [Color]) RECT([G], [x1, y1], [x2, y2], [edgeColor],[fillColor]) Draws a rectangle on G, with diagonal defined by points (x1,y1) and (x2,y2), using edgeColor for the perimeter and fillColor for the inside. The following values are optional and their defaults are listed: x1, y1=top left corner of G x2, y2=bottom right corner of G edgeColor=white fillColor=edgeColor To erase a GROB, execute RECT_P(G). To clear the screen, execute RECT_P(). Note: semi-transparent rectangles can be drawn by using the Alpha channel in the color (0 is opaque, 255 is transparent). The color can also be expressed as { color, alpha }. Examples: Demo_RECT
	SUBGROB	Copy GROB to Target Syntax: SUBGROB(srcG, [x1, y1], [x2, y2], trgtG) Sets graphic trgtG to be a copy of the area of srcG between points (x1,y1) and (x2,y2). If both (x1, y1) and (x2, y2) are not specified, then the entire graphic srcG is used. If (x1, y1) is not specified, then the top left corner of srcG is used; if (x2, y2) is not specified, then the bottom right corner of srcG is used. trgtGRB can be any of the graphic variables except G0. SUBGROB(G1, G4) will copy G1 in G4. Example: Demo_SUBGROB
	TEXTOUT	Draw Text Syntax: TEXTOUT(text, [G], x, y, [font], [textColor], [width], [backgroundColor]) Draws text on graphic G at position (x, y) using font and textColor. Paints the background before drawing the text using color backgroundColor. If width is specified, does not draw text more than width pixels wide. If backgroundColor is not specified, the background is not erased. The sizes for font are: 0=current font (default) 1=font 10 2=font 12 (Small) 3=font 14 (Medium) 4=font 16 (Large)

Help Topics Tree	13217	Help Text
		5=font 18 6=font 20 7=font 22 Returns the X (in pixels, not Cartesian) coordinate at which the next character of the string should be drawn if the string had more characters Examples: TEXTOUT("Hello HP Prime",-5,0,4,RGB(128,0,128),200,RGB(255,255,0)); FREEZE Demo_PISERIES
ICON		ICON keyword Insert "ICON name hexPngFile;" in the core of a program (i.e., not in a function) to create a Named graphic to use in subsequent graphic functions such as BLIT_P(G0, "name"). Although ICONs can be modified the intent is more as a source object. Size of icons cannot be changed. Therefore, ICONs can not be the target of SUB and DIMGROB functions. ICON are recreated each time a program is reloaded from storage, so any changes made will not be permanent unlike changes in EXPORT variables.
C→PX		Syntax: C→PX(x, y) or C→PX({x, y}) Converts from Cartesian coordinates to screen coordinates. Examples: C→PX(0,0) → {160,110} (assuming current app Plot Settings are set to default) C→PX({15.9,10.9}) → {319,0} (assuming current app Plot Settings are set to default)
DRAWMENU		Draw Button Menu Syntax: DRAWMENU(string1 or graphic, string2 or graphic,… string6 or graphic) Draws a six-button menu at the bottom of the display, with labels string1, string2, …, string6, or using the provided graphic (G0-G9 or "icon name"). Example: DRAWMENU("ABC","","DEF"); FREEZE creates a menu with the first and third buttons labeled ABC and DEF, respectively. The other four menu keys are blank.
FREEZE		Freeze Screen Syntax: FREEZE Prevents the screen from being redrawn after the program ends. Leaves the modified display on the screen for the user to see. This command does not pause and wait for input. Rather, it prevents a redraw until any other operation (key press, screen touch, or data communication, or command) triggers the screen to be drawn. Example: FREEZE
Matrix		Matrix Commands The Matrix commands allow matrices to be manipulated from within a program. In this help section, matrixname refers to the name of a matrix and must be M0, M1, M2, …, M9.
ADDCOL		Add Column Syntax: ADDCOL(matrixname, vector, column_number) Inserts values from vector into a column before column_number in the specified matrix. The size of vector must be the same as the number of rows in the matrix matrixname. Examples: ADDCOL([[1,3],[4,6]],[2,5],2) → [[1,2,3],[4,5,6]] ADDCOL([[1,3],[4,6]],{[2,5],[3,4]},{2,1}) → {[[1,2,3],[4,5,6]],[[3,1,3],[4,4,6]]} ADDCOL({[[1,3],[4,6]],[[1,9],[5,6]]},[2,5],2) → {[[1,2,3],[4,5,6]],[[1,2,9],[5,5,6]]} ADDCOL({[[1,3],[4,6]],[[1,9],[5,6]]},{[2,5],[3,4]},{2,1}) → {[[1,2,3],[4,5,6]],[[3,1,9],[4,5,6]]}
ADDROW		Add Row Syntax: ADDROW(matrixname, vector, row_number) Inserts values from vector into a row before row_number in the specified matrix. The size of vector must be the same as the number of columns in the matrix matrixname. Examples: ADDROW([[1,2],[5,6]],[3,4],2) → [[1,2],[3,4],[5,6]] ADDROW([[1,2],[5,6]],{[3,4],[2,5]},2) → {[[1,2],[3,4],[5,6]],[[1,2],[2,5],[5,6]]} ADDROW({[[1,3],[4,6]],[[1,9],[5,6]]},[2,5],2) → {[[1,3],[2,5],[4,6]],[[1,9],[2,5],[5,6]]} ADDROW({[[1,3],[4,6]],[[1,9],[5,6]]},{[2,5],[3,4]},{2,1}) → {[[1,3],[2,5],[4,6]],[[3,4],[1,9],[5,6]]}
DELCOL		Delete Column Syntax: DELCOL(name, column_number) Deletes column column_number from matrix name. Example: DELCOL([[1,2,3],[4,5,6]],2) → [[1,3],[4,6]]

Help Topics Tree	13217	Help Text
	DELROW	Delete Row Syntax: DELROW(name, row_number) Deletes row row_number from matrix name. Example: DELROW([[1,2][3,4][5,6]],2) → [[1,3],[4,6]]
	EDITMAT	Edit Matrix Syntax: EDITMAT(matrixvar, [title], [read only]) EDITMAT(matrix, [title], [read only] Allows the user to edit or view a specified matrix. If a matrix variable is used (e.g., M0-M9), updates the variable when the user taps the OK menu key. The optional title can be either "title" or { "title", {"row names"…}, {"column names"…}} If supplied, "title" will be displayed at the top of the editor. If "row names" and "column names" are specified, they will be used as row and column headers in the editor. If read only is not 0, the user will not be able to modify the matrix, but can only view it. EDITMAT returns the edited matrix upon completion. If used in programming, returns to the program when the user taps the OK menu key. Example: EDITMAT(M1) edits matrix M1.
	REDIM	Redimension Syntax: REDIM(matrixname, size) Redimensions the specified matrix 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 zeros.
	SCALEADD	Syntax: SCALEADD(matrixname, 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 matrixname. Examples: SCALEADD([[1,2],[3,4]],3,2,1) → [[10,14],[3,4]] SCALEADD([[1,2],[3,4]],{3,2},{2,1},{1,1}) → {[[10,14],[3,4]],[[3,6],[3,4]]}
	SUB	Extract Portion Syntax: SUB(object, start, end) Extracts a portion, of a list or matrix. For a matrix, start and end are two lists of two numbers ({row, col}) specifying the top left and bottom right of the portion to extract. For a vector or list, start and end are two numbers specifying the indexes of the first and last objects of the portion to extract. Examples: SUB([[1,2,1],[2,1,3],[4,2,3]],{2,1},{3,2}) → [[2,1],[4,2]] SUB({5,2,9,4},2,3) → {2,9}
	SWAPCOL	Swap Columns Syntax: SWAPCOL(matrixname, column1, column2) Exchanges column1 and column2 in the specified matrix matrixname. Examples: SWAPCOL([[1,2,1],[2,1,3],[4,2,3]],2,3) → [[1,1,2],[2,3,1],[4,3,2]] SWAPCOL([[1,2,1],[2,1,3],[4,2,3]],{1,2},{3,3}) → {[[1,2,1],[3,1,2],[3,2,4]],[[1,1,2],[2,3,1],[4,3,2]]} SWAPCOL({[[1,2,1],[2,1,3],[4,2,3]],[[9,8,7],[9,8,7]]},{1,2},{3,3}) → {[[1,2,1],[3,1,2],[3,2,4]],[[9,7,8],[9,7,8]]}
	SWAPROW	Swap Rows Syntax: SWAPROW(matrixname, row1, row2) Exchanges row1 and row2 in the specified matrix matrixname. Examples: SWAPROW([[1,2,1],[2,1,3],[4,2,3]],2,3) -→ [[1,2,1],[4,2,3],[2,1,3]] SWAPROW([[1,2,1],[2,1,3],[4,2,3]],{1,2},{3,3}) → {[[4,2,3],[2,1,3],[1,2,1]],[[1,2,1],[4,2,3],[2,1,3]]} SWAPROW({[[1,2,1],[2,1,3],[4,2,3]],[[9,9],[6,6],[5,5],[8,8]]},{1,2},{3,3}) → {[[4,2,3],[2,1,3],[1,2,1]],[[9,9],[5,5],[6,6],[8,8]]}
	SCALE	Syntax: SCALE(matrixname, value, row_number) Multiplies the specified row_number of the specified matrix by value. Examples: SCALE([1,2],3,1) → [3,6] SCALE([[1,2],[3,4]],3,2) → [[1,2],[9,12]] SCALE([[1,2],[3,4]],{3,2},{2,1}) → {[[1,2],[9,12]],[[2,4],[3,4]]}

Help Topics Tree 13217	Help Text
REPLACE	Syntax: REPLACE(Object1,Start,Object2) Replaces portion of a matrix, vector or string (Object1) starting at Start with Object2. For a matrix, Start is a list containing two numbers; for a vector or string it is a single number. Note: for strings, you can do: REPLACE("string", "sub_string", "replace_string") Examples: REPLACE([[1,2,3],[4,5,6]],{1,2},[[8,8],[9,9]]) → [[1,8,8],[4,9,9]] REPLACE([10,12,23],3,[9,8,7,6]) → [10,12,9,8,7,6] REPLACE("Replacement","place","configure") → "Reconfigurement" L1:={8,9,7,3,4,6,6,8,0}; REPLACE(L1,5,{1,2,3,4,5}) → {8,9,7,3,1,2,3,4,5}
App Functions	This menu contains commands for configuring Apps from within a program.
CHECK	Check (Select) Definition Syntax: CHECK(n) Checks (selects) the corresponding symbolic definition field in the current app. The integer n must be between 0 and 9 for most apps. For Statistics 1-Var and Statistics 2-Var apps, n must be between 1 and 5. For example, CHECK(3) would check F3 if the current app is Function. Then a checkmark would appear next to F3 in Symbolic view, F3 would be plotted in Plot view, and evaluated in Numeric view.
ISCHECK	Is Checked (Selected) Syntax: ISCHECK(n) Returns 1 or 0 depending if the corresponding symbolic definition field in the current app is checked or not. The integer n must be between 0 and 9 for most apps. For Statistics 1-Var and Statistics 2-Var apps, n must be between 1 and 5.
STARTAPP	Start Application Syntax: STARTAPP("AppName") Starts the app AppName. The App's START function will run if present. The App’s default view will be started. Note that the START function is always executed when the user presses the START menu key in the App Library. Also works for apps saved in the App Library.
UNCHECK	Uncheck (Deselect) Definition Syntax: UNCHECK(n) Unchecks (deselects) the corresponding symbolic definition field in the current app. The integer n must be between 0 and 9 for most apps. For Statistics 1-Var and Statistics 2-Var apps, n must be between 1 and 5. For example, UNCHECK(3) would uncheck F3 if the current app is Function.
VIEW	View Keyword Syntax: VIEW "Text" Function() BEGIN END; Only works in an app program. Allows a programmer to customize the View menu. Causes "Text" to appear when the View is pressed and Function to be executed when the OK menu key (or ENTER key) is pressed. Note that a view function can also be exported.
Integer	Integer Commands This menu contains commands working with integers.
BITAND	Bitwise AND Syntax: BITAND(int1, int2, … intn) Returns the bitwise logical AND of the specified integers. Example: BITAND(20,13) → 4
BITNOT	Bitwise NOT Syntax: BITNOT(int) Returns the bitwise logical NOT of the specified integer. Example: BITNOT(47) → 549755813840
BITOR	Bitwise OR Syntax: BITOR(int1, int2, … intn) Returns the bitwise logical OR of the specified integers. Example: BITOR(9,26) → 27
BITSL	Bitwise Shift Left Syntax:

Help Topics Tree 13217	Help Text
	BITSL(int1 [, int2]) 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 of places indicated by the second integer. If there is no second integer, then the bits in the first integer are shifted to the left one place. Examples: BITSL(28,2) → 112 BITSL(5) → 10
BITSR	Bitwise Shift Right Syntax: BITSR(int1 [, int2]) 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 of places indicated by the second integer. If there is no second integer, then the bits in the first integer are shifted to the right one place. Examples: BITSR(112,2) → 28 BITSR(10) → 5
BITXOR	Bitwise XOR Syntax: BITXOR(int1, int2, … intn) Returns the bitwise logical exclusive OR of the specified integers. Example: BITXOR(9,26) → 19
B→R	Base to Real Syntax: B→R(#integer[m]) Converts an integer in base m to a decimal integer (base10). The base marker m can be b (for binary), o (for octal), or h (for hexadecimal). If m is omitted, the current system base is assumed. Examples: B→R(#1101b) → 13 B→R(#1101) → 4353 (If system base is hexadecimal) B→R({#101h,#101o,#101b}) → {257,65,5}
GETBASE	Get Base Syntax: GETBASE(#integer[m]) Returns the base number for integer with base marker m. The base number is used by the SETBASE function. 0 = System 1 = Binary 2 = Octal 3 = Decimal 4 = Hexadecimal The base marker m can be b (for binary), o (for octal), d (for decimal), or h (for hexadecimal). If m is omitted, the current system base is assumed. Examples: GETBASE(#1101b) → #1h GETBASE(#1101) → #0h (if default base is hexadecimal) GETBASE({#100h,#100d,#100o,#100b}) → {#4h,#3h,#2h,#1h}
GETBITS	Get Bits Syntax: GETBITS(#integer) Returns the number of bits used for encoding an integer. If not specified, then the value in the Integers field of Page 1 of Home Settings is used. Examples: GETBITS(#22122) → 32 (If Home Settings Integers is set to 32 bits) GETBITS(#1:45h) → 45 GETBITS(#153:-16o) → -16 GETBITS({#FFFF:16h,#777:-23o}) → {16,−23}
R→B	Real to Base Syntax: R→B(Real [, bits [,base]]) Converts a decimal integer (base 10) to an integer. Optionally specify bits and base. 1 ≤ bits ≤ 64 (Unsigned integer) -1 ≥ bits ≥ -63 (Signed integer) base = 0 System base = 1 Binary base = 2 Octal base = 3 Decimal base = 4 Hexadecimal

Help Topics Tree 13217	Help Text
	Examples: R→B(13) → #Dh (If system base is hexadecimal) R→B(1800,64,2) → #3410:64o R→B({50,50,50},{64,32,16},{1,2,4}) → {#110010:64b,#62o,#32:16h}
SETBASE	Set Base Syntax: SETBASE(#integer[m] [,c]) Displays integer expressed in base m in whatever base is indicated by c. Base marker m can be b (for binary), d (for decimal), o (for octal), d (for decimal), or h (for hexadecimal). If m is omitted, the input is assumed to be in the default base. c = 0 System c = 1 Binary c = 2 Octal c = 3 Decimal c = 4 Hexadecimal If c is omitted, the output is displayed in the default base. Examples: SETBASE (#34o,1) → #11100b SETBASE (#1101b) → #Dh (if the default base is hexadecimal) SETBASE({#100d,#100d,#100d,#100d,#100d},{0,1,2,3,4}) → {#64h,#1100100b,#144o,#100d,#64h}
SETBITS	Set Bits Syntax: SETBITS(#integer[m] [,bits]) Sets the number of bits to represent integer. The value of bits must be in the range –63 to 64. Base marker m can be b (for binary), d (for decimal), o (for octal), d (for decimal), or h (for hexadecimal). If base marker m or bits is omitted, the default value is used. Examples: SETBITS(#1111b, 15) → #1111:15b SETBITS({#FFFFh,#777o},{15,7}) → {#7FFF:15h,#177:7o}
I/O	Input/Output Commands The Input/output commands allow users to interact with programs.
CHOOSE	Choose Box Syntax: CHOOSE(var, “title”, “item1”, “item2”,[…"item14"]) or CHOOSE(var,"title",{"item1"..."itemN"}) Displays a choose box with the given "title" and containing items with the strings "item1", etc. If the user chooses an object, var is updated to contain the number of the selected object (an integer, 1, 2, 3, …) and CHOOSE returns true (non zero). If the user exits without choosing, var is not changed and CHOOSE returns false (0). Examples: CHOOSE(A, "Pick a Number",1,2,3,4) CHOOSE(B, "Direction", {"Up","Left","Right","Down"})
EDITLIST	Edit List Syntax: EDITLIST(listvar or list, [title], [read only]) Allows the user to edit the specified list. If a list variable is used (e.g., L0-L9), updates the variable if OK is clicked. The title can be either "title" or { "title", {"row names"…}, {"column names"…}} "title" will be displayed above the editor as a "title" or "name". if "row names" and "column names" are specified, they will be used as row and column headers. If read only is non 0, the user will not be able to modify the object. Returns the edited list upon completion. Example: L1:={"123","456"};EDITLIST(L1) edits list L1 EDITLIST({1,2,3},"My List",1) displays a list but does not allow editing
EDITMAT	Edit Matrix Syntax: EDITMAT(matrixvar, [title], [read only]) EDITMAT(matrix, [title], [read only] Allows the user to edit or view a specified matrix. If a matrix variable is used (e.g., M0-M9), updates the variable when the user taps the OK menu key. The optional title can be either "title" or { "title", {"row names"…}, {"column names"…}} If supplied, "title" will be displayed at the top of the editor. If "row names" and "column names" are specified, they will be used as row and column headers in the editor. If read only is not 0, the user will not be able to modify the matrix, but can only view it.

Help Topics Tree 13217	Help Text
	EDITMAT returns the edited matrix upon completion. If used in programming, returns to the program when the user taps the OK menu key. Example: EDITMAT(M1) edits matrix M1.
GETKEY	Get Key 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). 0 = Apps 1 = Symb 2 = Up 3 = Help 4 = Esc 5 = Home 6 = Plot 7 = Left 8 = Right 9 = View 10 = CAS 11 = Num 12 = Down 13 = Menu After that, the keys are numbered from top left (14 = Vars) to bottom right (50 = +)
INPUT	Input Form Syntax: INPUT(var,[“title”], [“label”], [“help”], [reset_value], [initial_value]) INPUT({vars},[“titles”], [{“labels”}], [{“helps”}], [{reset_values}], [{initial_values}]) var -> {var_name, real, [{pos}]} var -> {var_name, [allowed_types_matrix], [{pos}]} var -> {var_name, {choose_items}, [{pos}]} 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 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, var 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 format: {var_name, real, [{pos}]} to create a checkbox control. If real is >1, then this checkbox gets pooled with the next n -1 checkboxes in a radio group (i.e., only one of the n checkboxes 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, then the edition will hide 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 percentage, field width in screen percentage, line (starts at 0) }. This allows you to control precisely the 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 7 lines of controls per page. Controls with more than 7 lines will be placed in subsequent pages. If more than one page is created, titles can be a list of titles.
ISKEYDOWN	Is Key Pressed Syntax: ISKEYDOWN(KeyIdentifier) Returns true (non-zero) if the key whose KeyIdentifier is provided is currently pressed, and false (0) if it is not.
MOUSE	Get Touch Event 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 z, 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	Message Box Syntax: MSGBOX(expr,[OK_Cancel]) or MSGBOX(string,[OK_Cancel]) Displays a message box with either the value of expr or string.

Help Topics Tree 13217	Help Text
	If OK_Cancel is true, displays OK and CANCEL menu keys, otherwise only displays the OK menu key. Default value for OK_Cancel is false. Returns true (non-zero) if the user presses OK, false (0) if the user presses CANCEL. Example: MSGBOX("Click OK to continue")
STARTVIEW	Start View Syntax: STARTVIEW(ViewNumber[,Redraw]) Starts a view of the current app. Redraw, is optional; if Redraw, is true (non 0), it will force a refresh for the view. The view numbers are as follows: 0=Symbolic 1=Plot 2=Numeric 3=Symbolic Setup 4=Plot Setup 5=Numeric Setup 6=App Info 7=Views key If the current app has views defined under the Views menu, then the following view numbers are used: 8=First special view (Split Screen Plot Detail) 9=Second special view (Split Screen Plot Table) 10=Third special view (Autoscale) 11=Fourth special view (Decimal) 12=Fifth special view (Integer) 13=Sixth special view (Trig) If ViewNumber is negative, the following global views are used: -1=Home Screen -2=Modes -3=Memory Manager -4=App Library -5=Matrix Catalog -6=List Catalog -7=Program Catalog -8=Note Catalog Example: STARTVIEW(-3)
PRINT	Syntax: PRINT(expr) or PRINT(string) PRINT( ) Prints either the result of expr 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 use the up/down keys to view the text, Backspace to erase the text and any other key to hide the terminal. You can show the terminal at anytime using the ON+T combination (press and hold the On key, press the T key, then release both keys). Pressing On stops the interaction with the terminal. PRINT with no argument clears the terminal.
wilcoxonp	Wilcoxon Distribution Syntax: wilcoxonp(Integer1,[Integer2]) Distribution of the Wilcoxon or Mann-Whitney test for one or two samples. Examples: wilcoxonp(4) wilcoxonp(7,5)

wilcoxons	Wilcoxon statistic
	Syntax:
	wilcoxons(List1,Median)
	wilcoxons(List1,List2)
	Rank statistic of Wilcoxon or Mann-Whitney test for 1 sample (List1) and Median, or 2 samples (List1,List2).
	Examples:
	wilcoxons([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , [2, 6, 10, 11, 13, 14, 15, 18, 19, 20])
	wilcoxons([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , 10)

wilcoxont	Wilcoxon test
	Syntax:
	wilcoxons(List1,Median, [Method],[Significance])
	wilcoxons(List1,List2)
	wilcoxont(List,List || Real,[Func],[Real])

Help Topics Tree	13217	Help Text

		Wilcoxon or Mann-Whitney test for 1 sample (List1) and Median, or 2 samples (List1,List2). Optionally, specify Method to be '<' or '>', and Significance.
		Examples:
		wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , [2, 6, 10, 11, 13, 14, 15, 18, 19, 20])
		wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , [2, 6, 10, 11, 13, 14, 15, 18, 19, 20],0.01)
		wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , 10,'>')
		wilcoxont([1, 3, 4, 5, 7, 8, 8, 12, 15, 17] , 10,'>',0.05)

WAIT		Syntax: WAIT(n) Pauses program execution. If n ≥ 1 : Execution paused for the specified number (n) seconds. Returns the value of n. If n = 0 or omitted : Execution paused until a key is pressed. If a key is pressed, the key code is returned. After a 1-minute timeout, returns -1 If n = -1 : Execution paused until a key is pressed or there is a mouse event. If a key is pressed, the key code is returned. If a mouse event happens, a list of the form { type, [x, y], [dx, dy] } is returned. Normally x/y is the event position unless otherwise indicated. After a 1-minute timeout, returns -1 Event type can be: 0: Mouse Down 1: Mouse Move 2: Mouse Up (x/y is not provided) 3: Mouse Click (if a click is detected, there is no Mouse Up) 5: Mouse Stretch. x/y is the delta since the last event. dx/dy is the delta since the original mouse down. 6: Mouse Rotate, x is original angle, y is new angle in 32nd of a circle. 7: Mouse Long Click, indicates the mouse stayed down for 1 second. Example: WAIT(3)
More		This menu contains additional programming commands.
CAS		CAS Evaluation Syntax: CAS(expression) or CAS.function(...) or CAS.variable[(...)] Evaluate an expression or variable using the CAS. Note that outputs in numerical mode are transformed into strings or lists of expressions for symbolic matrices.
EXECON		Execute On Element 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}) → {2,3,4} In the example above, &1 indicates an element in the list. &1+1 means to add 1 to each element of the list. Where the & is followed directly by a number, the relative position in the list is indicated. For example: EXECON("&2-&1",{1,4,3,5}) → {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}) → {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. These element pairs are added 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 refers to the element in the list; 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}) → {7,12}

Help Topics Tree	13217	Help Text
		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. EXECON can also operate on matrices in the same way as lists: EXECON("&1+&2",[[1,2],[3,4]],[[5,6],[6,7]]) → [[6,8],[9,11]] In the example above, the result is the sum of the two matrices.
	EVALLIST	Evaluate List Syntax: EVALLIST({list}) Evaluates the content of each element in the list and returns the resulting list. Example: EVALLIST({'1+1','4/2*(6-3)'}) → {2,6}
	→HMS	Syntax: →HMS(value) Displays a decimal value in sexagesimal format; that is, in units subdivided into groups of 60. This includes degrees, minutes, and seconds as well as hours, minutes, and seconds. Examples: →HMS(8.5) → 8°30' →HMS({8.5,37.7539}) → {8°30’00”,37°45’14.04”}
	HMS→	Syntax: HMS→(value) Displays a sexagesimal value in decimal format. Examples: HMS→(8°30) → 8.5 HMS→({8°30’00”,286°15’00”}) → {8.5,286.25}
	ITERATE	Iterate Expression Syntax: ITERATE(expr, var, ivalue, times) For times, recursively evaluates expr in terms of var, beginning with var = ivalue. Examples: ITERATE(X^2, X, 2, 3) → 256 ITERATE({'X^2','Y^3','Z+1'},{'X','Y','Z'},{2,3,4},{3,2,3}) → {256,19683,7}
	TEVAL	Time Evaluation Syntax: TEVAL(Param) Returns the time it takes to evaluate the parameter. Example: TEVAL(WAIT(5)) → ~5.095_s Note: actual result will vary but should be close to 5.00_s
	TICKS	Internal Ticks Value Syntax: TICKS() Returns the internal millisecond clock value. Example: TICKS
	TYPE	Object 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
	%CHANGE	Percent Change Syntax: %CHANGE(x, y) Percent change from x to y. Returns 100*(y-x)/x. Examples: %CHANGE(20,50) → 150 %CHANGE(4.5,8.3) → 84.4444444444 %CHANGE({10,20,30},{75,75,75}) → {650,275,150}
	%TOTAL	Percent Total

Help Topics Tree	13217	Help Text
		Syntax: %TOTAL(x, y) Percent total; the percentage of x that is y. Returns 100*y/x. Examples: %TOTAL(20,50) → 250 %TOTAL(1.5,7.5) → 500 %TOTAL({10,20,30},{75,75,75}) → {750,375,250}
Template Menu		This menu contains all the programing structure and templates that are useful when editing a program
Block		Block Menu This menu contains commands for entering block structures in programs.

BEGIN END		BEGIN END Block
		Syntax:
		BEGIN commands; END;
		Defines a set of commands to be executed in a block.
		Example:
		EXPORT SQM1(X)
		BEGIN
		  RETURN X^2-1;
		END;
		This program defines a user function named SQM1(X). Entering SQM1(8) returns 63.

KILL		Stop Execution Syntax: KILL; Stops the execution of a program. Example: Demo_KILL

RETURN		Return Command
		Syntax:
		RETURN expression;
		Exits from a function and returns the value of expression (optional).
		Example:
		EXPORT FACTORIAL(N)
		BEGIN
		  IF N==1 THEN
		    RETURN 1;
		  ELSE
		    RETURN N*FACTORIAL(N-1);
		  END;
		END;
		Example:
		$Demo_RETURN

Branch		Branch Commands This menu contains common branch commands such as IF … THEN.

CASE		Starts a "CASE … END" branch structure.
		Syntax:
		    CASE
		    IF test1 THEN commands1 END
		    IF test2 THEN commands2 END
		    …
		     IF testN THEN commandsN END
		    [DEFAULT] [commandsD]
		    END;
		Evaluates test1. If true, executes commands1 and ends the CASE. Otherwise, evaluates test2. If true, executes commands2. Continues evaluating tests until a true is found. If no true test is found, executes commandsD, if provided.
		Example:
		Demo_CASE

IF THEN ELSE END		IF Branch Structure
		Syntax:
		IF test THEN commands1 [ELSE commands2] END;
		Starts an "IF … THEN … END" or "IF … THEN … ELSE … END" branch structure.
		Evaluate test. If test is true (non 0), executes commands1, otherwise, executes commands2
		
		Example:
		IF A<1
		  THEN PRINT("A<1");
		  ELSE PRINT("A>1");
		END;
		Examples:

Help Topics Tree	**13217**	**Help Text**
		Demo_IF
IFERR		Error Trapping Structure Syntax: 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. Many conditions are automatically recognized by the HP Prime as error conditions and are automatically treated as errors in programs. This command facilitates error-trapping of such errors. Note: the error number will be stored in the Ans variable. So you can access it and use it in the THEN clause of the IFERR. Example: IFERR 1/0 THEN PRINT("1/0 Error"); END; Example: Demo_IFERR
Variable		Variable Menu This menu contains options relating to the variables you can create in programs.
EXPORT		EXPORT function or variables Syntax: EXPORT FunctionName(Parameters) EXPORT Var1[,Var2, ... ,Var8]; EXPORT Var1[:=Val1, Var2:=Val2, ... Var8:=Val8]; In a program, declares functions or variables to export globally. Exported functions appear in the Toolbox User menu; exported variables appear in the Vars CAS, App, or User menus. For an exported function: Forward function declaration: EXPORT function(params); Normal function declaration: EXPORT function[(params)] BEGIN //Function definition goes here END; Examples: EXPORT X2m1(X); EXPORT ratio:=0.15; EXPORT X2M1(X) BEGIN RETURN X^2-1; END; Examples: Demo_EXPORT
:=		Assign Syntax: variable := object Assigns object to variable. Examples: A := 3 stores the value 3 in the variable A F1 := 3-X makes F1(X)=3-X M5 := [1, 2] stores a vector in M5
LOCAL		LOCAL keyword Syntax: LOCAL Var1[:=Val1, Var2:=Val2, ... Var8:=Val8]; Declares one or more local variables. Each variable can be assigned an optional initial value as well. If the declaration is in a function block, these variables will be local to the function. If the declaration is in the main program body, the variables are local to the program. There can only be 8 variables per LOCAL keyword. To create more variables, you must add another LOCAL keyword. Examples: Demo_LOCAL
▶		Store Syntax: value ▶ variable Stores value in variable. Example: 3▶A stores the value 3 in the variable A.
Loop		Loop Commands This menu contains loop commands such as FOR … NEXT.
REPEAT		Repeat Loop Structure

Help Topics Tree				13217	Help Text
					Syntax: REPEAT commands UNTIL test; Executes commands UNTIL test is true. A:=5; REPEAT PRINT(A); A:= A-1; UNTIL A<1; will print 5 4 3 2 1 Examples: Demo_REPEAT GETSIDES
				BREAK	Break Loop Syntax: BREAK [n]; Exits from expression local loop structure. Example: FOR A FROM 1 TO 10 DO B:= (A+3) MOD 5 IF B==1 THEN BREAK; END; END; If n is specified, allow to exit n loop structures. Example: Demo_BREAK
				CONTINUE	Syntax: CONTINUE [n]; Transfers execution in a loop to the start of the next iteration of the nth upper loop (default current loop). Example: Demo_CONTINUE

				FOR FROM TO DO END	For Loop Structure
					Syntax:
					FOR var FROM start TO (or DOWNTO) finish [STEP increment] DO commands END;
					Sets variable var to start; then, for as long as this variable's value is less than or equal to (or more than for a DOWNTO) finish, executes commands and adds (or subtracts for DOWNTO) 1 (or increment) to var.
					Examples:
					//print 1 3 5 7 9
					FOR A FROM 1 TO 10 STEP 2
					  DO
					  PRINT(A);
					END;
					//print 10 8 6 4 2
					FOR A FROM 10 DOWNTO 1 STEP 2
					  DO
					  PRINT(A);
					END;
					Example:
					Demo_FOR

				WHILE	While Loop Structure Syntax: WHILE test DO commands END; Executes commands WHILE test is true. Example: A:=5; WHILE A>0 DO PRINT(A); A:= A-1; END; will print 5 4 3 2 1 Examples: Demo_WHILE ISPERFECT PERFECTNUMS
			KEY		Key Keyword Syntax: KEY name BEGIN END;

Help Topics Tree		13217	Help Text
			Declaring a function with the KEY keyword allows to redefine the appropriate key in the keyboard. The name of the function specifies the key. See user manual for the complete list. Note that a key function can also be exported and be a view.
		VIEW	View Keyword Syntax: VIEW "Text" Function() BEGIN END; Only works in an app program. Allows a programmer to customize the View menu. Causes "Text" to appear when the View is pressed and Function to be executed when the OK menu key (or ENTER key) is pressed. Note that a view function can also be exported.
	Debugging Environment		Once you tap the Debug menu key with a program selected, if a program hits a Debug statement or if you debug a program using the debug command, the debugging environment starts. This environment has three parts: • A title bar at the top, with the current name of the program and/or routine being debugged • The listing of the program being debugged (if available, else it displays the current and next instructions) • A variable watch listing in a two-column table In the variable list, type the names of the variables you want to observe during debugging in the left column. Their current values will be displayed in the right column. By typing a new value in the right column, you can change the value of a variable. The menu keys are: • Skip: executes a subroutine but does not debug it • Step: moves to the next step in the program; if the next step is a subroutine, then steps down into it and begins to debug it • Swap: switches to view the calculator display so you can see your program output (press any key to return to the debugger) • Stop: quits the debugging environment and returns to the Program Catalog • Cont: continues execution of the program without debugging If the variable list has the focus, Edit can be used to change the variable listed on a row or to change the variable value. If the program listing has the control, Edit will stop the program evaluation and jump to the program editor to allow you to modify the program. You can drag the program source code. You can drag the variable list header to see more or less of the program as needed.
Note Catalog			Press Shift 0 (Notes) to open the Note Catalog. Menu Buttons: • Edit: opens the highlighted note for editing in the Notes Editor • New: creates a new note • More: opens more menu options (Save, Rename, Sort, Delete, and Clear) for the highlighted note • Send: sends the highlighted note to another HP Prime Note: the Editor saves your changes automatically when you exit the editor. If you want to save the original version of your note before you make changes, be sure to use the More button in the Note Catalog and select Save.
	Note Editor		The Note Editor is where you create or modify a note. Menu Buttons: • Format: displays a menu of formatting options • Style: displays a menu of style options • ▲ Page ▼: moves from page to page in a multi-page note • •: cycles through bullet styles • Insert: tap to display a menu of items that can be inserted Press ALPHA twice to lock the alpha shift. Press it again to release the alpha shift. You can copy and paste text using Shift View (Copy) and Shift Menu (Paste) respectively.
Messaging			When connected to a PC either wirelessly or via USB, the PC can send messages to the calculator. You can also send a message back to the PC. See the HP Prime Connectivity Kit User Guide for more details. Either dismiss the message by tapping OK, or tap Reply, enter your message, and tap OK to send it.
Poll and Quiz			Poll and Quiz functionality enables communication between a computer and any number of other calculators. It enables teachers to wirelessly communicate with students' calculators, send them questions, and receive responses. The responses are aggregated and displayed. These features enable formative assessment as well as active participation of the students in classroom activities. Poll and Quiz functionality requires the HP Prime Connectivity Kit. See the User Guide that accompanies the HP Prime Connectivity Kit for instructions.

Help Topics Tree 13217	Help Text
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. 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. Exam Mode can be configured, and activated, on Home Settings Page 3. You can quickly access this page by holding down the ON key and pressing Esc. Basic and Custom Mode You can create your own exam mode configurations using the Custom Mode as a basis. Or you can use the Basic Mode. To use the Basic Mode, simply tap on the Configuration field and select Basic Mode. Then tap the Start menu key. The Basic Mode configuration is as follows: • The HP Prime memory is hidden and restored when the exam mode exits • The green LED at the top of the calculator is set to blink • The Basic Mode will end when the calculator is connected to either a PC or another HP Prime via the micro-USB cable Fields • Configuration: choose Basic Mode or create your own Custom Mode (see below) • Timeout: choose a duration for Exam Mode (that is, how long the disabled features will be disabled). • Default Angle: sets the angle measure (degrees or radians) that all client calculators must use, or allows the current calculator settings to be used • Password: a code that will deactivate Exam Mode before the time-out period. • Memory: Keep, Erase, Hide, or Keep and delete memory changes that occurred during exam mode • Blink LED: forces the LED on all client calculators to maintain a fixed sequential pattern • Security code: check to prompt for a security code to enter at the start of exam mode that drives the LED blink pattern Menu Buttons: • Config: view and edit the selected exam configuration (not available in Basic Mode) • Choose: select an option from the currently selected menu • ▲ Page ▼: return to the previous page (left side of button) or go to the next page (right side of button) • More: display options to copy the configuration or reset it (not available in Basic Mode) • Start or Send: activate exam mode or send it to another HP Prime Note that Send only appears if the HP Prime is connected to another HP Prime via the USB cable on HP Prime calculators that support unit-to-unit connectivity.
Configuring Exam Mode	On the Exam Mode settings screen (Home Settings Page 3), tap Config. The Exam Mode Configuration screen appears with a tree depicting the sets of features that can be disabled. Each entry has a check box to its left. Tap an entry to select it and then either tap its check box to disable it or tap on the ✓ menu button. Some of the entries have a plus sign (+) to the left of their check boxes. These entries represent categories; you can check it to disable all the features in the category or tap on the plus sign (+) to expand the category and disable certain features within the category. 1. Select those features you want disabled, and make sure that those features you don’t want disabled are not selected. 2. Tap OK to return to Exam Mode Settings.
Creating New Configurations	As well as modifying Custom Exam Mode configuration, you can create new configurations. You can then select the particular configuration you want before activating Exam Mode. 1. On Home Settings Page 3, choose the configuration you want to be the base for a new configuration. (If you haven't yet created a new configuration, the only configuration available will be Custom Mode.) 2. Tap More and select Copy The New Exam Mode Configuration screen appears. 3. Enter a name for the new configuration. 4. Tap OK twice. 5. Tap Config. 6. Select those functions you want disabled, and make sure that those functions you don’t want disabled are not selected. 7. Tap OK. The new configuration has now been added to the Configuration field drop-down box.
Activating Exam Mode	Exam Mode can be activated on Home Settings Page 3. You can quickly access this page by holding down the On key and pressing Esc. 1. On Home Settings Page 3, choose the configuration you want to activate. 2. Tap Start. A summary screen will appear notifying you that you are about to enter an exam mode. Swipe the lock to start exam mode, or tap Cancel to exit and return to Home Settings Page 3. 3, Depending on the exam mode configuration, you may be prompted to enter a security code provided by your proctor. 4. You can always exit exam mode by connecting the HP Prime to a PC via the USB cable or connecting it to another HP Prime via the micro-USB unit-to-unit cable.

Help Topics Tree		13217	Help Text
			Depending on the exam mode configuration, you may also be able to exit exam mode by: • entering the password on the Exam Mode settings screen and tapping OK • waiting for the timeout period to expire You can also activate the current Exam Mode configuration on an HP Prime attached to yours via the USB cable, if both units support unit-to-unit connectivity. In this case, the Start menu button is replaced with a Send menu button once the HP Primes are connected. Tap the Send menu button to install and start the current Exam Mode configuration on the attached HP Prime. Finally, you can install and start an Exam Mode configuration on one or more HP Prime calculators using the Connectivity Kit. The HP Prime calculators can communicate with the Connectivity Kit either via USB cable or using the optional wireless module(s) if your HP Prime supports wireless connectivity. Please see the Connectivity Kit User Guide for details.