TI-82 Advanced Edition Python Reference Guide

Catalog, Commands and Functions, Error Messages

Arithmetic Operations, Test Relations, and Symbols

Learn more about TI Technology through the online help at education.ti.com/eguide.

Important Information

Except as otherwise expressly stated in the License that accompanies a program, Texas Instruments makes no warranty, either expressed or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an “as-is” basis. In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the purchase price of this product. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

Contents

Introduction

In this Reference Guide you will find the following information:

- CATALOG, Strings, Hyperbolic Functions - Includes instructions on browsing, using, entering strings, and other functions in the CATALOG.
- Commands and Functions Listing - Includes an alphabetical listing of all CATALOG items, referencing:
  - Function or Instruction/Arguments
  - Results
  - Key or Keys/Menu or Screen/Item
- Arithmetic Operations, Test Relations, and Symbols - Items whose names are not alphabetic (such as +, !, and >).
- Error Messages - Includes a listing of error types with possible causes and suggested remedies.

CATALOG, Strings, Hyperbolic Functions

What Is the CATALOG?

The CATALOG is an alphabetical list of all functions and instructions on the TI-82 Advance Edition Python. You also can access each CATALOG item from a menu or the keyboard, except:

- The six string functions
- The six hyperbolic functions
- The **solve(** instruction without the equation solver editor
- The inferential stat functions without the inferential stat editors

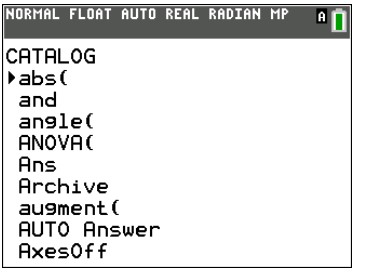
Note: The only CATALOG programming command you can execute from the home screen is **GetCalc(**.

Browsing the TI-82 Advanced Edition Python Catalog

Selecting an Item from the CATALOG

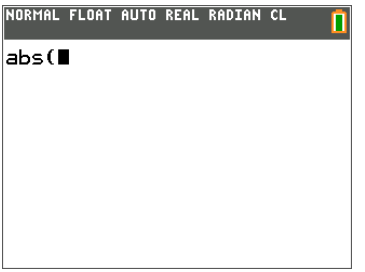
To browse and select a **CATALOG** item, follow these steps.

1. Press 2nde [catalog] to display the **CATALOG**.



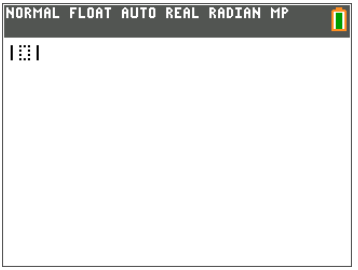
   The ▸ in the first column is the selection cursor.

2. Press ▼ or ▲ to scroll the **CATALOG** until the selection cursor points to the item you want.
   - To jump to the first item beginning with a particular letter, press that letter; alpha-lock is on.
   - Items that begin with a number are in alphabetical order according to the first letter after the number. For example, **2-PropZTest(** is among the items that begin with the letter **P**.
   - Functions that appear as symbols, such as **+**, $^{-1}$, **<**, and $\sqrt{}$**(**, follow the last item that begins with **Z**. To jump to the first symbol, **!**, press [θ].
3. Press entrer to paste the item to the current screen.

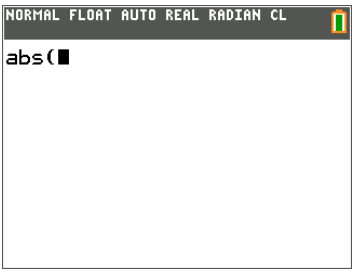


Note:

- From the top of the **CATALOG** menu, press ▲ to move to the bottom. From the bottom, press ▼ to move to the top.
- When your calculator is in MathPrint™ mode, many functions will paste the MathPrint™ template on the home screen. For example, **abs(** pastes the absolute value template on the home screen instead of **abs(**.



MathPrint™



Classic

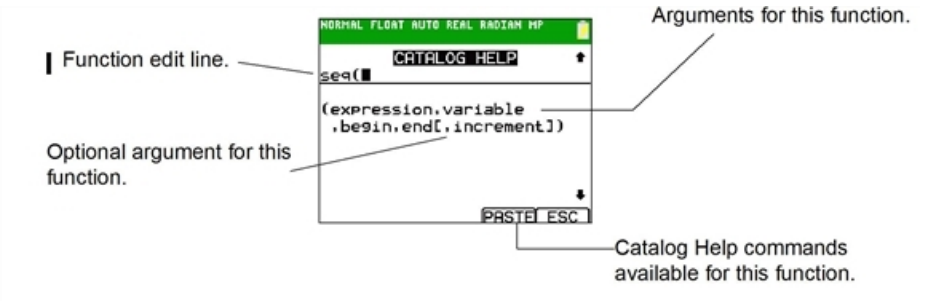
Using Catalog Help

Displaying Catalog Help

You can display Catalog Help arguments for functions in two ways:

- Using an alpha/numeric function listing in the catalog (e.g, 2nde [catalog]).
- Using the functions listed in certain menus (e.g, math).

Catalog Help lists the valid arguments for the function under the edit line. Arguments in brackets are optional.



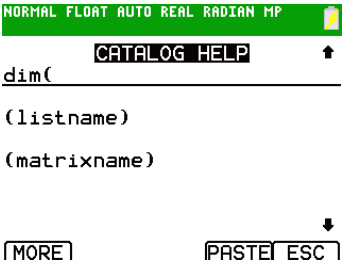
1. Display the menu that contains the function.
2. Use ▲ and/or ▼ to move the cursor to the function.
3. Press + to display arguments for the function. The cursor is on the function edit line.

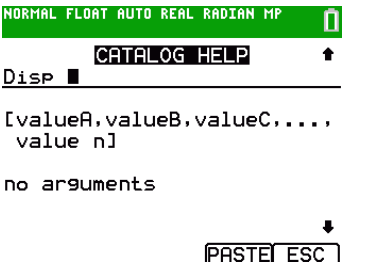
Note:

- The catalog (2nde [catalog]) is displayed in alphabetical order. When you display the catalog, the alpha-lock is turned on. Press the first letter of the function name to skip function names that come before it alphabetically. Use ▲ and/or ▼ to move the cursor to the function.
- Not all catalog functions have associated arguments. If the function does not require an argument, Catalog Help displays the message "***No arguments required for this item***."

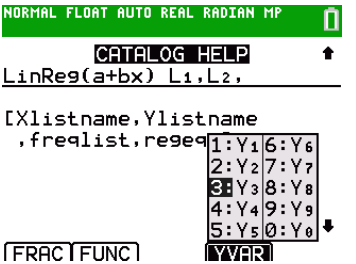
Catalog Help Commands

- Select **MORE** (if available) to display more arguments for the function.



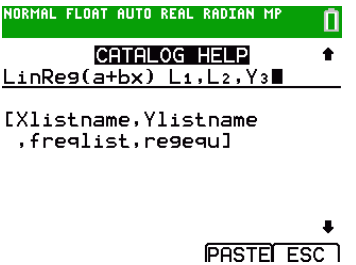


- Use shortcut menus alpha [f1] through [f4] through for argument values if available.



- Enter your argument values on the function edit line, and then select **PASTE** to paste the function and the argument values you entered.

  Note: You can paste to most cursor locations.



- Select **ESC** to exit the Catalog Help screen.

Entering and Using Strings

What Is a String?

A string is a sequence of characters that you enclose within quotation marks. On the TI-82 Advanced Edition Python, a string has two primary applications.

- It defines text to be displayed in a program.
- It accepts input from the keyboard in a program.

Characters are the units that you combine to form a string.

- Each number, letter, and space counts as one character.
- Each instruction or function name, such as **sin(** or **cos(**, counts as one character; the TI-82 Advanced Edition Python interprets each instruction or function name as one character.

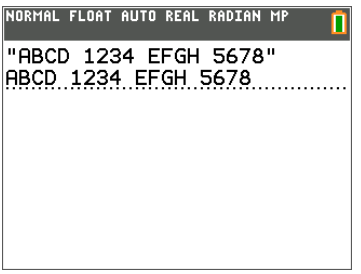
Entering a String

To enter a string on a blank line on the home screen or in a program, follow these steps.

1. Press alpha ["] to indicate the beginning of the string.
2. Enter the characters that comprise the string.
   - Use any combination of numbers, letters, function names, or instruction names to create the string.
   - To enter a blank space, press alpha [␣].
   - To enter several alpha characters in a row, press alpha [verr A] to activate alpha-lock.
3. Press alpha ["] to indicate the end of the string.

   "*string*"

4. Press entrer. On the home screen, the string is displayed on the next line without quotations. An ellipsis (**...**) indicates that the string continues beyond the screen. To scroll to see the entire string, press ▶ and ◀.



Note: A string must be enclosed in quotation marks. The quotation marks do not count as string characters.

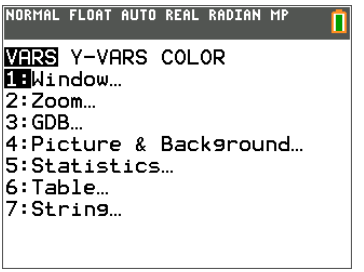
Storing Strings to String Variables

String Variables

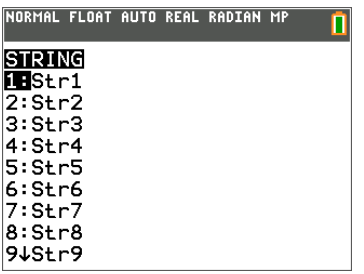
The TI-82 Advanced Edition Python, has 10 variables to which you can store strings. You can use string variables with string functions and instructions.

To display the **VARS STRING** menu, follow these steps.

1. Press [var] to display the **VARS** menu. Move the cursor to **7:String**.



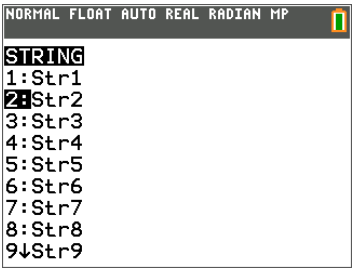
2. Press [entrer] to display the **STRING** secondary menu.



Storing a String to a String Variable

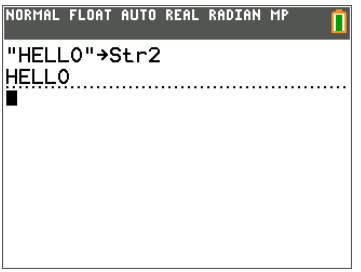
To store a string to a string variable, follow these steps.

1. Press [alpha] ["], enter the string, and press [alpha] ["].
2. Press [sto→].
3. Press [var] **7** to display the **VARS STRING** menu.
4. Select the string variable (from **Str1** to **Str9**, or **Str0**) to which you want to store the string.



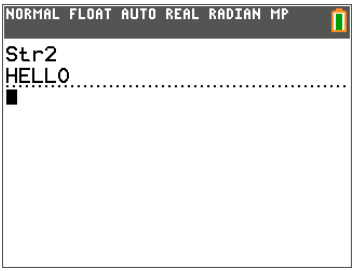
The string variable is pasted to the current cursor location, next to the store symbol (→).

5. Press entrer to store the string to the string variable. On the home screen, the stored string is displayed on the next line without quotation marks.



Displaying the Contents of a String Variable

To display the contents of a string variable on the home screen, select the string variable from the **VARS STRING** menu, and then press entrer. The string is displayed.



String Functions and Instructions in the CATALOG

Displaying String Functions and Instructions in the CATALOG

String functions and instructions are available only from the CATALOG. The table below lists the string functions and instructions in the order in which they appear among the other **CATALOG** menu items. The ellipses in the table indicate the presence of additional CATALOG items.

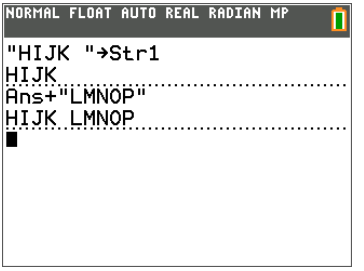
CATALOG	
...	
Equ▸String(	Converts an equation to a string.
...	
expr(	Converts a string to an expression.
...	
inString(	Returns a character's place number.
...	
length(	Returns a string's character length.
...	
String▸Equ(	Converts a string to an equation.
sub(	Returns a string subset as a string.
...	

Concatenation

To concatenate two or more strings, follow these steps.

1. Enter *string1*, which can be a string or string name.
2. Press +.
3. Enter *string2*, which can be a string or string name. If necessary, press + and enter *string3*, and so on.

   string1+*string2*+*string3*...
4. Press enter to display the strings as a single string.



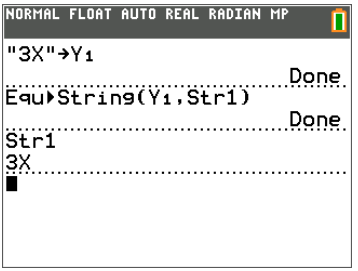
Selecting a String Function from the CATALOG

To select a string function or instruction and paste it to the current screen, follow the steps for selecting an item from the CATALOG.

Equ▸String(

Equ▸String(converts an equation to a string. The equation must be store in a VARS Y-VARS variable. **Y***n* contains the equation. **Str***n* (from **Str1** to **Str9**, or **Str0**) is the string variable to which you want the equation to be stored.

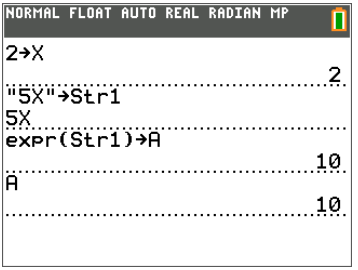
Equ▸String(Y*n*,**Str***n*)

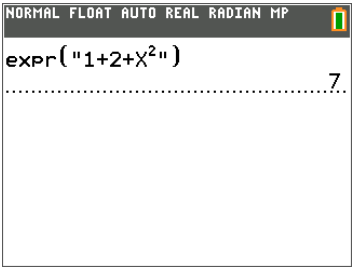


expr(

expr(converts the character string contained in *string* to an expression and executes the expression. *string* can be a string or a string variable.

expr(*string***)**

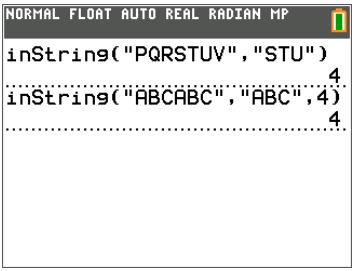




inString(

inString(returns the character position in *string* of the first character of *substring*. *string* can be a string or a string variable. *start* is an optional character position at which to start the search; the default is 1.

inString(*string*,*substring*[,*start*]**)**



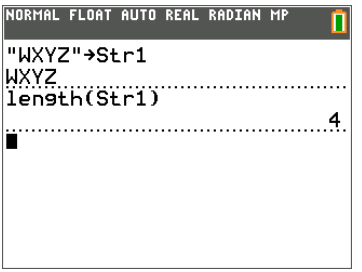
Note: If *string* does not contain *substring*, or *start* is greater than the length of *string*, **inString(** returns **0**.

length(

length(returns the number of characters in *string*. *string* can be a string or string variable.

Note: An instruction or function name, such as **sin(** or **cos(,** counts as one character.

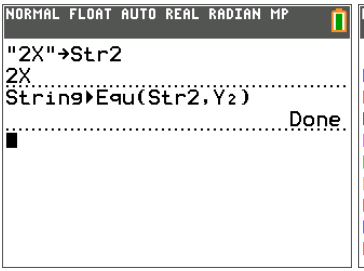
length(*string***)**

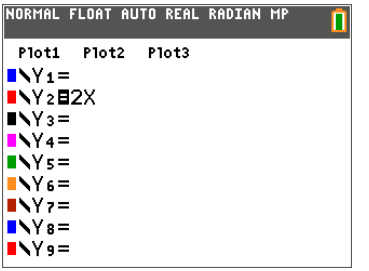


String▶Equ(

String▶Equ(converts *string* into an equation and stores the equation to **Y***n*. *string* can be a string or string variable. **String▶Equ(** is the inverse of **Equ▶String(**.

String▶Equ(*string*,**Y***n***)**

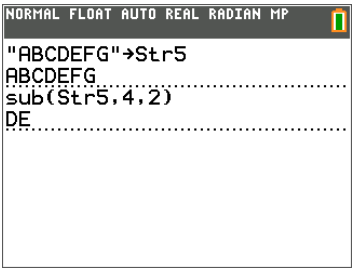




sub(

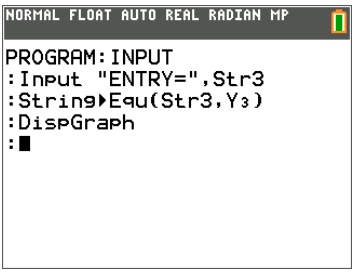
sub(returns a string that is a subset of an existing *string*. *string* can be a string or a string variable. *begin* is the position number of the first character of the subset. *length* is the number of characters in the subset.

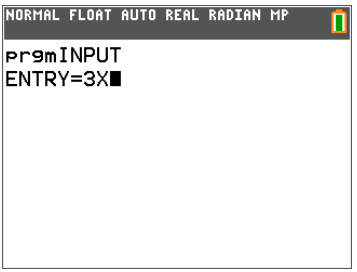
sub(*string*,*begin*,*length*)**

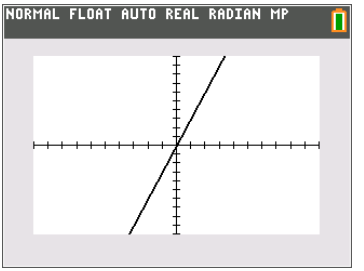


Entering a Function to Graph during Program Execution

In a program, you can enter a function to graph during program execution using these commands.







Note: When you execute this program, enter a function to store to **Y3** at the **ENTRY=** prompt.

Hyperbolic Functions in the CATALOG

Hyperbolic Functions

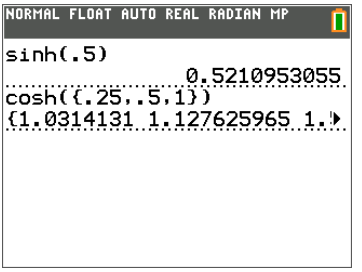
The hyperbolic functions are available only from the CATALOG. The table below lists the hyperbolic functions in the order in which they appear among the other **CATALOG** menu items. The ellipses in the table indicate the presence of additional CATALOG items.

CATALOG	
...	
cosh(	Hyperbolic cosine
$\cosh^{-1}($	Hyperbolic arccosine
...	
sinh(	Hyperbolic sine
$\sinh^{-1}($	Hyperbolic arcsine
...	
tanh(	Hyperbolic tangent
$\tanh^{-1}($	Hyperbolic arctangent
...	

sinh(, cosh(, tanh(

sinh(, **cosh(**, and **tanh(** are the hyperbolic functions. Each is valid for real numbers, expressions, and lists.

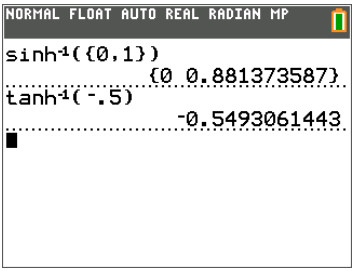
sinh(*value***)**
cosh(*value***)**
tanh(*value***)**



sinh^{-1}(, cosh^{-1}(, tanh^{-1}(

sinh^{-1}(is the hyperbolic arcsine function. **cosh^{-1}(** is the hyperbolic arccosine function. **tanh^{-1}(** is the hyperbolic arctangent function. Each is valid for real numbers, expressions, and lists.

sinh^{-1}(*value***)**
cosh^{-1}(*value***)**
tanh^{-1}(*value***)**


NORMAL FLOAT AUTO REAL RADIAN MP
sinh⁻¹({0,1})
{0 0.881373587}
tanh⁻¹(-.5)
-0.5493061443

Commands and Functions Listing

The purpose of this table of information is to provide a short description with syntax of command arguments as appropriate and menu locations for each command or function in the Catalog listing in the calculator.

This table is useful for executing commands when using the calculator or creating TI-Basic programs.

Items whose names are not alphabetic (such as +, !, and >) are listed in the Arithmetic Operations, Test Relations, and Symbols section. Unless otherwise specified, all examples in this section were performed in the default reset mode, and all variables are assumed to be the default value of 0.

From the **CATALOG**, you can paste any function or command to the home screen or to a command line in the program editor.

The same syntax information for function and command arguments below is available on the calculator and also in the TI Connect™ CE Program Editor. For the TI-82 Advanced Edition Python, please use the TI-83 Premium CE catalog in TI Connect™ CE v5.6.3 or higher. Please select only the TI-82 Advanced Python Edition commands and functions as described in this Reference Guide. Running unsupported commands and functions will result in an Invalid error on the TI-82 Advanced Edition Python.

- On the calculator, pressing [+] when a function or command is highlighted in the menu listing will display the Catalog Help syntax editor to assist your entries.
- Using TI Connect™ CE Program Editor, the catalog listing also displays the syntax of the arguments for functions and commands.

Note that some functions and commands are only valid when executed in a TI-Basic program and not from the home screen.

The items in this table appear in the same order as they appear in the **CATALOG** ([2nde] [catalog])

In the table below, the **†** symbol indicates either keystrokes or certain commands which are only available in the Program Editor mode on the calculator. Press [prgm] and select to **EDIT** an existing program or **NEW** to start a new program to set the calculator in the Program Edit mode.

Some arguments are optional. Optional arguments will be indicated within **[ ]** in the syntax help given in the table below. **[ ]** are not symbols on the calculator and are not to be typed in. They are used here only to indicate an optional argument.

On the calculator, functions and commands paste as "tokens." This means they paste as one character and not as individual letters, symbols and spaces. Do not attempt to type in any function or command on the calculator. Just paste the token from menu locations. Watch the cursor jump over tokens as you edit to get a better understanding of tokens.

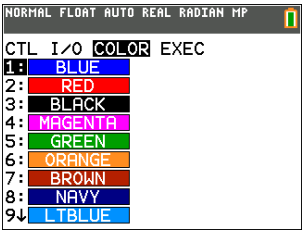
In TI Connect™ CE Program Editor, you can "feel" the same experience of pasting tokens when using the Catalog tree provided in that editor. You also can type in the functions and commands if you know the correct format and syntax. TI Connect™ CE

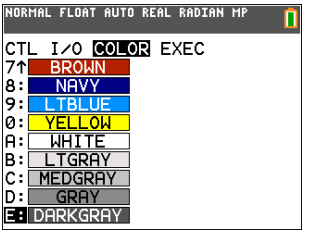
"tokenizes" the functions and commands when you send the program to the calculator. However, you must type in the functions and commands exactly as the tokens. Note that some commands will have spaces as part of the token which you might not see. **For example**, Pause command as a token has a space at the end. Once you send the program to the calculator, you can run the program and if there are any syntax errors, you can fix the issues on either the calculator or in TI Connect™ CE Program Editor.

CTL I/O COLOR EXEC

Color Numbers	Names
10	BLUE
11	RED
12	BLACK
13	MAGENTA
14	GREEN
15	ORANGE
16	BROWN
17	NAVY
18	LTBLUE
19	YELLOW
20	WHITE
21	LTGRAY
22	MEDGRAY
23	GRAY
24	DARKGRAY

You can also choose a name in the [var] menu (**COLOR** sub-menu).





GraphColor(*function#,color#***)**

For example, **GraphColor(2,4)** or **GraphColor(2,MAGENTA)**.

Alpha CATALOG Listing

A

abs()

abs(*value***)**

Returns the absolute value of a real number, expression, list, or matrix.

[math]
NUM
1:abs(

abs()

abs(*complex value***)**

Returns the magnitude of a complex number or list.

[math]
CMPLX
5:abs(

and

valueA **and** *valueB*

Returns 1 (true) when both *valueA* and *valueB* are true. Otherwise, return is 0 (false).

valueA and *valueB* can be real numbers, expressions, or lists.

TI Connect™ Program Editor Tip:

Notice the token is "_and_" where "_" is a space.

[2nde] [tests]
LOGIC
1:and

angle()

angle(*value***)**

Returns the polar angle of a complex number or list of complex numbers.

[math]
CMPLX
4:angle(

ANOVA()

ANOVA(*list1***,***list2***[,***list3***,...,***list20***])**

Performs a one-way analysis of variance for comparing the means of two to 20 populations.

[stats]
TESTS
H:ANOVA(

Ans

Ans

Returns the last answer.

[2nde]
[rép]

Archive

Archive *variables*

Moves the specified *variable* from RAM to the user data archive memory.

2nde [mém] **5:Archive**

augment()

augment(*matrixA* **,***matrixB* **)**

Returns a matrix, which is *matrixB* appended to *matrixA* as new columns.

matrice **MATH 7:augment(**

augment()

augment(*listA***,***listB***)**

Returns a list, which is *listB* concatenated to the end of *listA*.

2nde [listes] **OPS 9:augment(**

AUTO Answer

AUTO

Displays answers in a similar format as the input.

mode **Answers: AUTO**

AxesOff

AxesOff

Turns off the graph axes.

† 2nde [format] **AxesOff**

AxesOn

AxesOn[color#**]**

Turns on the graph axes with color. The *color* option allows the color of the axes to be specified.

Color#: 10 - 24 or color name pasted from [vars] COLOR..

† 2nde [format] **AxesOn**

a+b*i*

a+b*i*

Sets the mode to rectangular complex number format (a+b*i*).

† mode **a+b** *i*

B

BackgroundOff

BackgroundOff

Turns off background image in the graph area.

† 2nde [dessin] BACKGROUND 2:BackgroundOff:

BackgroundOn

BackgroundOn n

Displays a menu the Background Image Var n (Image#n) specified in the graph area.

† 2nde [dessin] BACKGROUND 1:BackgroundOn

bal(

bal(*npmt*[,*roundvalue*])

Computes the balance at *npmt* for an amortization schedule using stored values for **PV**, **I**%, and **PMT** and rounds the computation to *roundvalue*.

2nde [apps] 1:Finance CALC 9:bal(

binomcdf(

binomcdf(*numtrials*,*p*[,*x*])

Computes a cumulative probability at *x* for the discrete binomial distribution with the specified *numtrials* and probability *p* of success on each trial.

2nde [distrib] DISTR B:binomcdf(

binompdf(

binompdf(*numtrials*,*p*[,*x*])

Computes a probability at *x* for the discrete binomial distribution with the specified *numtrials* and probability *p* of success on each trial.

2nde [distrib] DISTR A:binompdf(

BorderColor

BorderColor[color#**]**

Turns on a border color surrounding the graph area with the specified color. Color #:1-4.

† 2nde [format] BorderColor

Boxplot

Boxplot Plot#(*type,Xlist,*[**,***freqlist,color#*]**)**

Defines Plot# (1, 2, or 3) of type

† 2nde [graph stats] **TYPE**

C

checkTmr(

checkTmr(*starttime***)**

Returns the number of seconds since you used **startTmr** to start the timer. The *starttime* is the value displayed by **startTmr**.

2nde [catalog] **checkTmr(**

χ^2cdf(

χ^2cdf(*lowerbound,upperbound,df***)**

Computes the χ^2distribution probability between *lowerbound* and *upperbound* for the specified degrees of freedom *df*.

2nde [distrib] **DISTR** **8: χ^2 cdf(**

χ^2pdf(

χ^2pdf(*x,df***)**

Computes the probability density function (pdf) for the χ^2 distribution at a specified *x* value for the specified degrees of freedom *df*.

2nde [distrib] **DISTR** **7: χ^2 pdf(**

χ^2-Test(

χ^2-Test(*observedmatrix,expectedmatrix* [**,***drawflag,color#*]**)**

Performs a chi-square test. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats **TESTS** **C: χ^2 - Test (**

χ^2GOF

χ^2GOF-Test(*observedlist,expectedlist,df* [**,***drawflag,color#*]**)**

Performs a test to confirm that sample data is from a population that conforms to a specified distribution.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats **TESTS** **D: χ^2 GOF - Test(**

Circle(

Circle(*X*,*Y*,*radius*[,*color#*,*linestyle#*])

Draws a circle with center (*X*,*Y*) and *radius* with specified

Color#: 10 - 24 or color name pasted from [vars] COLOR.

linestyle#: 1-2.

2nde [dessin]
DRAW
9:Circle(

CLASSIC

CLASSIC

Displays inputs and outputs on a single line, such as 1/2+3/4.

mode
CLASSIC

Clear Entries

Clear Entries

Clears the contents of the Last Entry storage area.

2nde [mém]
MEMORY
3:Clear Entries

ClockOff

ClockOff

Turns off the clock display in the mode screen.

2nde [catalog]
ClockOff

ClockOn

ClockOn

Turns on the clock display in the mode screen.

2nde [catalog]
ClockOn

ClrAllLists

ClrAllLists

Sets to **0** the dimension of all lists in memory.

2nde [mém]
MEMORY
4:ClrAllLists

ClrDraw

ClrDraw

Clears all drawn elements from a graph or drawing.

2nde [dessin]
DRAW
1:ClrDraw

ClrHome

ClrHome

Clears the home screen.

† prgm
I/O
8:ClrHome

ClrList

ClrList*listname1***[,***listname2***, ...,***listname n***]**

Sets the dimension of one or more listnames to 0.

stats
EDIT
4:ClrList

ClrTable

ClrTable

Clears all values from the table.

† prgm
I/O
9:ClrTable

conj(

conj(*value***)**

Returns the complex conjugate of a complex number or list of complex numbers.

math
CMPLX
1:conj(

CoordOff

CoordOff

Turns off cursor coordinate value display.

† 2nde [format]
CoordOff

CoordOn

CoordOn

Turns on cursor coordinate value display.

† 2nde [format]
CoordOn

cos(

cos(*value***)**

Returns cosine of a real number, expression, or list.

2nde [catalog]

cos^{-1}(

cos^{-1}(*value*)

Returns arccosine of a real number, expression, or list.

2nde [catalog]

cosh(

cosh(*value*)

Returns hyperbolic cosine of a real number, expression, or list.

2nde [catalog] **cosh(**

cosh^{-1}(

cosh^{-1} (*value*)

Returns hyperbolic arccosine of a real number, expression, or list.

2nde [catalog] **cosh^{-1} (**

CubicReg

CubicReg [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a cubic regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats **CALC 6:CubicReg**

cumSum(

cumSum(*list*)

Returns a list of the cumulative sums of the elements in *list*, starting with the first element.

2nde [listes] **OPS 6:cumSum(**

cumSum(

cumSum(*matrix*)

Returns a matrix of the cumulative sums of *matrix* elements. Each element in the returned matrix is a cumulative sum of a *matrix* column from top to bottom.

matrice **MATH 0:cumSum(**

dayOfWk(

dayOfWk(*year*,*month*,*day*)

Returns an integer from 1 to 7, with each integer representing a day of the week. Use **dayOfWk(** to determine on which day of the week a particular date would occur. The *year* must be 4 digits; *month* and *day* can be 1 or 2 digits.

2nde [catalog]
dayOfWk(
1:Sunday
2:Monday
3:Tuesday...

dbd(

dbd(*date1*,*date2*)

Calculates the number of days between *date1* and *date2* using the actual-day-count method.

2nde [apps]
1:Finance
CALC
D:dbd(

DEC Answers

DEC

Displays answers as integers or decimal numbers.

mode
Answers:
DEC

▶Dec

value▶**Dec**

Displays a real or complex number, expression, list, or matrix in decimal format.

math
MATH
2:▶Dec

Degree

Degree

Sets degree angle mode.

† mode
Degree

DelVar

DelVar *variable*

Deletes from memory the contents of *variable*.

† prgm
CTL
G:DelVar

DependAsk

DependAsk

Sets table to ask for dependent-variable values.

† 2nde [déf table]
Depend: Ask

DependAuto

DependAuto

Sets table to generate dependent-variable values automatically.

† 2nde [déf table]
Depend: Auto

det(

det(*matrix*)

Returns determinant of *matrix*.

matrice
MATH
1:det(

DetectAsymOff

DetectAsymOff

Turns off checks for rational function asymptotes when graphing. Impacts graph speed. Does not perform extra calculations to detect asymptotes pixel to pixel while graphing. Pixels will connect across the screen even across an asymptote.

† 2nde [format]
DetectAsymOff

DetectAsymOn

DetectAsymOn

Turns on checks for rational function asymptotes when graphing. Impacts graph speed. Performs more calculations and will not connect pixels across an asymptote on a graph.

† 2nde [format]
DetectAsymOn

DiagnosticOff

DiagnosticOff

Sets diagnostics-off mode; **r**, $\mathbf{r}^2$, and $\mathbf{R}^2$ are not displayed as regression model results.

2nde [catalog]
DiagnosticOff

DiagnosticOn

DiagnosticOn

Sets diagnostics-on mode; **r**, $\mathbf{r}^2$, and $\mathbf{R}^2$ are displayed as regression model results.

2nde [catalog]
DiagnosticOn

dim(

dim(*listname*)

Returns the dimension of *listname*.

2nde [listes]
OPS

dim(

3:dim(

dim(

dim(*matrixname***)**

Returns the dimension of *matrixname* as a list.

matrice
MATH
3:dim(

dim(

length→**dim(***listname***)**

Assigns a new dimension (*length*) to a new or existing *listname*.

2nde [listes]
OPS
3:dim(

dim(

{*rows*,*columns***}**→**dim(***matrixname***)**

Assigns new dimensions to a new or existing *matrixname*.

matrice
MATH
3:dim(

Disp

Disp

Displays the home screen.

† prgm
I/O
3:Disp

Disp

Disp [*valueA*,*valueB*,*valueC*,...,*value n***]**

Displays each value.

† prgm
I/O
3:Disp

DispGraph

DispGraph

Displays the graph.

† prgm
I/O
4:DispGraph

DispTable

DispTable

Displays the table.

† prgm
I/O
5:DispTable

▸DMS

value▸**DMS**

Displays *value* in DMS format.

2nde [angle]
ANGLE
4: ▸ DMS

Dot-Thick

Dot-Thick

Sets dot plotting mode; resets all Y=editor graph-style settings to Dot-Thick.

† mode
Dot-Thick

Dot-Thin

Dot-Thin

Sets dot plotting mode; resets all Y=editor graph-style settings to Dot-Thin.

† mode
Dot-Thin

DrawF

DrawF*expression*[,*color#*]

Draws *expression* (in terms of **X**) on the graph with specified

Color#:10 - 24 or color name pasted from [vars] COLOR.

2nde [dessin]
DRAW
6:DrawF

DrawInv

DrawInv*expression*[,*color#*]

Draws the inverse of *expression* by plotting **X** values on the y-axis and **Y** values on the x-axis with specified

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [dessin]
DRAW
8:DrawInv

DS<(

DS<(*variable*,*value*):*commandA*:*commands*

Decrements *variable* by 1; skips *commandA* if *variable* < *value*.

† prgm
CTL
B:DS<(

E

e

e

Returns decimal approximation of the constant **e**.

2nde [e]

e^(

e^(*power***)** 2nde [e^x]

Returns **e** raised to *power*.

e^(

e^(*list***)** 2nde [e^x]

Returns a list of **e** raised to a *list* of powers.

E

Exponent: 2nde [EE]
*value***E***exponent*

Returns *value* times 10 to the *exponent*.

E

Exponent: 2nde [EE]
*list***E***exponent*

Returns *list* elements times 10 to the *exponent*.

E

Exponent: 2nde [EE]
*matrix***E***exponent*

Returns *matrix* elements times 10 to the *exponent*.

▶Eff(

▶Eff(*nominal rate***,**
*compounding periods***)** 2nde [apps]
1:Finance
CALC
C: ▶ Eff(

Computes the effective interest rate.

Else

Else

See **If:Then:Else**

End

End

Identifies end of **For(**, **If**-**Then**-**Else**, **Repeat**, or **While** loop.

† prgm
CTL
7:End

Eng

Eng

Sets engineering display mode.

† mode
Eng

Equ▶String(

Equ▶String(Y= *var*,**Str***n*)

Converts the contents of a **Y=** *var* to a string and stores it in **Str***n*

2nde [catalog]
Equ ▶ String
(

eval(

eval(*expression*)

Returns an evaluated expression as a string with 8 significant digits. The expression must be real.

† prgm
I/O
A:eval(

ExecLib

ExecLib

Extends TI-Basic (not available)

† prgm
CTL
K:ExecLib

expr(

expr(*string*)

Converts the character string contained in *string* to an expression and executes the expression. *string* can be a string or a string variable.

† prgm
I/O
B:expr(

ExpReg

ExpReg [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits an exponential regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
0:ExpReg

ExprOff

ExprOff

Turns off the expression display during **TRACE**.

† 2nde [format]
ExprOff

ExprOn

ExprOn

Turns on the expression display during **TRACE**.

† 2nde [format]
ExprOn

F

Fcdf(

Fcdf(*lowerbound*,*upperbound*,*numerator df*,*denominator df***)**

Computes the F distribution probability between *lowerbound* and *upperbound* for the specified *numerator df* (degrees of freedom) and *denominator df*.

2nde [distrib]
DISTR
0: Fcdf(

▶F◀▶D

▶F◀▶D

Converts an answer from a fraction to a decimal or from a decimal to a fraction. Fraction and or decimal may be an approximation.

alpha [f1]
4: ▶F◀▶D
or
math
NUM
B: ▶F◀▶D
math
FRAC
3: ▶F◀▶D

Fill(

Fill(*value*,*matrixname***)**

Stores *value* to each element in *matrixname*.

matrice
MATH
4:Fill(

Fill(

Fill(*value*,*listname***)**

2nde [listes]

Fill(

Stores *value* to each element in *listname*.

OPS
4:Fill(

Fix

Fix #

Sets fixed-decimal mode for # of decimal places.

† mode
0123456789
(select one)

Float

Float

Sets floating decimal mode.

† mode
Float

fMax(

fMax(*expression*,*variable*,*lower*,*upper*[,*tolerance*])

Returns the value of *variable* where the local maximum of *expression* occurs, between *lower* and *upper*, with specified *tolerance*.

math
MATH
7:fMax(

fMin(

fMin(*expression*,*variable*,*lower*,*upper*[,*tolerance*])

Returns the value of *variable* where the local minimum of *expression* occurs, between *lower* and *upper*, with specified *tolerance*.

math
MATH
6:fMin(

fnInt(

fnInt(*expression*,*variable*,*lower*,*upper*[,*tolerance*])

Returns the function integral of *expression* with respect to *variable*, between *lower* and *upper*, with specified *tolerance*.

math
MATH
9:fnInt(

FnOff

FnOff [*function#*,*function#*,...,*function n*]

Deselects all **Y=** functions or specified **Y=** functions.

var
Y-VARS
4:On/Off
2:FnOff

FnOn

FnOn [*function#,function#,...,function n***]**

Selects all **Y=** functions or specified **Y=** functions.

[var]
Y-VARS
4:On/Off
1:FnOn

For(

:For(*variable,begin,end*
[*,increment***]):***commands***:End:***commands*

Executes *commands* through **End**, incrementing *variable* from *begin* by *increment* until *variable>end*.

† [prgm]
CTL
4:For(

fPart(

fPart(*value***)**

Returns the fractional part or parts of a real or complex number, expression, list, or matrix.

[math]
NUM
4:fPart(

Fpdf(

Fpdf(*x,numerator df,denominator df***)**

Computes the F distribution probability between *lowerbound* and *upperbound* for the specified *numerator df* (degrees of freedom) and *denominator df*.

[2nde] [distrib]
DISTR
9: Fpdf(

▶Frac

value▶**Frac**

Displays a real or complex number, expression, list, or matrix as a fraction simplified to its simplest terms.

[math]
MATH
1: ▶Frac

Full

Full

Sets full screen mode.

† [mode]
Full

Func

Func

Sets function graphing mode.

† [mode]
Func

G

GarbageCollect

GarbageCollect

Displays the garbage collection menu to allow cleanup of unused archive memory.

2nde [catalog]
GarbageCollect

gcd(

gcd(*valueA*,*valueB*)

Returns the greatest common divisor of *valueA* and *valueB*, which can be real numbers or lists.

math
NUM
9:gcd(

geometcdf(

geometcdf(*p*,*x*)

Computes a cumulative probability at *x*, the number of the trial on which the first success occurs, for the discrete geometric distribution with the specified probability of success *p*.

2nde [distrib]
DISTR
F:geometcdf(

geometpdf(

geometpdf(*p*,*x*)

Computes a probability at *x*, the number of the trial on which the first success occurs, for the discrete geometric distribution with the specified probability of success *p*.

2nde [distrib]
DISTR
E:geometpdf(

GetCalc(

GetCalc(*variable*[,*portflag*])

Gets contents of *variable* on another TI-82 Adv Edition Python and stores it to *variable* on the receiving TI-82 Adv Edition Python. By default, the TI-82 Adv Edition Python uses the USB port if it is connected. If the USB cable is not connected, it uses the I/O port.
portflag=0 use USB port if connected;
portflag=1 use USB port;
portflag=2 use I/O port.

† prgm
I/O
0:GetCalc(

getDate

getDate

Returns a list giving the date according to the current value of the clock. The list is in *{year,month,day}* format.

2nde [catalog]
getDate

getDtFmt

getDtFmt

2nde [catalog] **getDtFmt**

Returns an integer representing the date format that is currently set on the device.

1 = M/D/Y
2 = D/M/Y
3 = Y/M/D

getDtStr(

getDtStr(*integer*)

2nde [catalog] **getDtStr(**

Returns a string of the current date in the format specified by *integer*, where:

1 = M/D/Y
2 = D/M/Y
3 = Y/M/D

getTime

getTime

2nde [catalog] **getTime**

Returns a list giving the time according to the current value of the clock. The list is in *{hour,minute,second}* format. The time is returned in the 24 hour format.

getTmFmt

getTmFmt

2nde [catalog] **getTmFmt**

Returns an integer representing the clock time format that is currently set on the device.

12 = 12 hour format
24 = 24 hour format

getTmStr(

getTmStr(*integer*)

2nde [catalog] **getTmStr(**

Returns a string of the current clock time in the format specified by *integer*, where:

12 = 12 hour format
24 = 24 hour format

getKey

getKey

† prgm **I/O 7:getKey**

Returns the key code for the current keystroke, or **0**, if no key is pressed.

Goto

Goto*label*

Transfers control to *label*.

† prgm
CTL
0:Goto

GraphColor(

GraphColor(*function#*,*color#*)

Sets the color for *function#*.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† prgm
CTL
H:GraphColor(

GraphStyle(

GraphStyle(*function#*,*graphstyle#*)

Sets a *graphstyle* for *function#*.

† prgm
CTL
H:GraphStyle(

GridDot

GridDot [*color#*]

Turns on grid dots in the graph area in the specified color.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† 2nde [format]
GridDot

GridLine

GridLine [*color#*]

Turns on grid lines in the graph area in the specified color.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† 2nde [format]
GridLine

GridOff

GridOff

Turns off grid format.

† 2nde [format]
GridOff

G-T

G-T

Sets graph-table vertical split-screen mode.

† mode
GRAPH-
TABLE

H

Histogram

Histogram Plot#*(type,Xlist,*[*,freqlist,color#*])

Used as the "type" argument in the command

Where # gives Plot1, Plot2 or Plot3.

† 2nde [graph stats]
TYPE

Horiz

Horiz

Sets horizontal split-screen mode.

† mode
Horiz

Horizontal

Horizontal *y*[,*color#,linestyle#*]

Draws a horizontal line at *y* in a specified

Color#: 10 - 24 or color name pasted from [vars] COLOR.

line style #: 1-4.

2nde [dessin]
DRAW
3:Horizontal

I

i

i

Returns the complex number *i*.

2nde [*i*]

identity(

identity(*dimension***)**

Returns the identity matrix of *dimension* rows x *dimension* columns.

matrice
MATH
5:identity(

If

If *condition*:*commandA*:*commands*

If *condition* = 0 (false), skips *commandA*.

† prgm
CTL
1:If

If
Then
End

If:*condition***Then:***commands***End:***commands*

Executes *commands* from **Then** to **End** if *condition* = 1 (true).

† prgm
CTL
2:Then

If
Then
Else
End

If:
*condition***Then:***commands***Else:***commands***End:***commands*

Executes *commands* from **Then** to **Else** if *condition* = 1 (true); from **Else** to **End** if *condition* = 0 (false).

† prgm
CTL
3:Else

imag(

imag(*value***)**

Returns the imaginary (non-real) part of a complex number or list of complex numbers.

math
CMPLX
3:imag(

invBinom(

invBinom(*area,trial,p***)**

The inverse binomial cumulative distribution function results in the minimum number of successes, such that the cumulative probability for that minimum number of successes ≥ the given cumulative probability (area). If more information is needed, also find the binomcdf for the result from invBinom(as shown below for a full analysis.

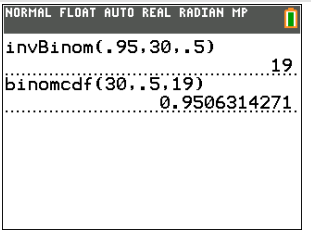
Details:

Assume the toss of a fair coin 30 times. What is the minimum number of heads you must observe such that the cumulative probability for that number of observed heads is at least 0.95?

The results on the screen first show that the minimum number of successes to obtain at least the given cumulative probability of 0.95 is 19. Next, the cumulative probability for up to 19 is computed using binomcdf(and is approximately 0.9506314271 which meets the criteria of 0.9506314271≥0.95

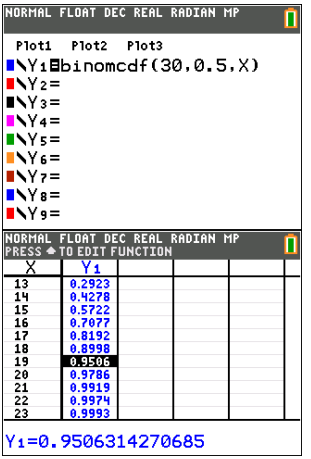
2nde distrib
DISTR
C:invBinom(

invBinom(



Alternate Method:

Set Y1=binomcdf(30,0.5,X) and use the table of values (starting at 0 and increment by 1) to find when the cumulative probability is at or just above the given cumulative probability. This gives you a view of all values to make decisions. For this example, search in the table to find the cumulative probability just larger than 0.95. Again, the number of successes is 19.



IndpntAsk

IndpntAsk

Sets table to ask for independent-variable values.

† 2nde [déf table]
Indpnt: Ask

IndpntAuto

IndpntAuto

Sets table to generate independent-variable values automatically.

† 2nde [déf table]
Indpnt:

IndpntAuto

Auto

Input

Input

Displays graph.

† prgm
I/O
2:Input

Input

Input [*variable***]**

Input ["*text***",***variable***]**

Prompts for value to store to *variable*.

† prgm
I/O
2:Input

Input

Input [Str*n***,***variable***]**

Displays **Str***n* and stores entered value to *variable*.

† prgm
I/O
2:Input

inString(

inString(*string***,***substring***[,***start***])**

Returns the character position in *string* of the first character of *substring* beginning at *start*.

2nde [catalog]
inString(

int(

int(*value***)**

Returns the largest integer a real or complex number, expression, list, or matrix.

math
NUM
5:int(

ΣInt(

ΣInt(*pmt1***,***pmt2***[,***roundvalue***])**

Computes the sum, rounded to *roundvalue*, of the interest amount between *pmt1* and *pmt2* for an amortization schedule.

2nde [apps]
1:Finance
CALC
A: Σ Int(

invNorm(

invNorm(*area*[,μ,σ,tail])

2nde [distrib]
DISTR
3:invNorm(

tail [catalog]: **LEFT**, **CENTER**, **RIGHT**

Computes the inverse cumulative normal distribution function for a given area under the normal distribution curve specified by μ and σ. The optional argument tail can be **LEFT** (-∞,-a), **CENTER** [-a,a] or **RIGHT** (a, ∞) for Real a.

The tokens **LEFT**, **CENTER** and **RIGHT** can be found in [catalog].

LEFT

LEFT

2nde [catalog]
LEFT

LEFT is a tail argument for the **invNorm(** command where the optional argument tail can be **LEFT** (-∞,-a), **CENTER** [-a,a] or **RIGHT** (a, ∞) for Real a.

See also invNorm(.

RIGHT

RIGHT

2nde [catalog]
RIGHT

RIGHT is a tail argument for the **invNorm(** command where the optional argument tail can be **LEFT** (-∞,-a), **CENTER** [-a,a] or **RIGHT** (a, ∞) for Real a.

See also invNorm(.

CENTER

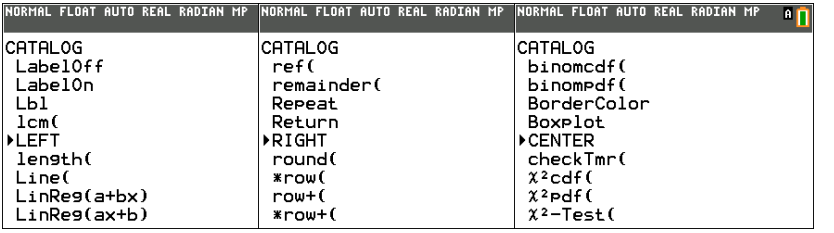
CENTER

2nde [catalog]
CENTER

CENTER is a tail argument for the **invNorm(** command where the optional argument tail can be **LEFT** (-∞,-a), **CENTER** [-a,a] or **RIGHT** (a, ∞) for Real a.

See also invNorm(.

LEFT **RIGHT** **CENTER**



invT(

invT(*area,df***)**

Computes the inverse cumulative student-t probability function specified by degree of freedom, df for a given area under the curve.

2nde [distrib]
DISTR
4:invT(

iPart(

iPart(*value***)**

Returns the integer part of a real or complex number, expression, list, or matrix.

math
NUM
3:iPart(

irr(

irr(*CF0,CFList*[**,***CFFreq*]**)**

Returns the interest rate at which the net present value of the cash flow is equal to zero.

2nde [apps]
1:Finance
CALC
8:irr(

isClockOn

isClockOn

Identifies if clock is ON or OFF. Returns 1 if the clock is ON. Returns 0 if the clock is OFF.

2nde
[catalog]
isClockOn

IS>(

:IS>(*variable***,***value***)**
:*commandA*
:*commands*

Increments *variable* by 1; skips *commandA* if *variable*>*value*.

† prgm
CTL
A:IS>(

L

L

L *listname*

Identifies the next one to five characters as a user-created list name.

2nde [listes]
OPS
B: L

LabelOff

LabelOff

Turns off axes labels.

† 2nde [format]
LabelOff

LabelOn

LabelOn

Turns on axes labels.

† 2nde [format]
LabelOn

Lbl

Lbl *label*

Creates a *label* of one or two characters.

† prgm
CTL
9:Lbl

lcm(

lcm(*valueA*,*valueB*)

Returns the least common multiple of *valueA* and *valueB*, which can be real numbers or lists.

math
NUM
8:lcm(

length(

length(*string*)

Returns the number of characters in *string*.

2nde
[catalog]
length(

Line(

Line(*X1*,*Y1*,*X2*,*Y2*[,*erase#*,*color#*,*linestyle#*])

Draws a line from (*X1*,*Y1*) to (*X2*,*Y2*) with the following options: erase #: 1,0, color #: 10-24, and line style #: 1-4.

2nde [dessin]
DRAW
2:Line(

Line(

Line(*X1*,*Y1*,*X2*,*Y2*,**0**[,*line#*])

Erases a line (erase #: 1,0) from (*X1*,*Y1*) to (*X2*,*Y2*).

2nde
[dessin]
DRAW
2:Line(

LinReg(a+bx)

LinReg(a+bx) [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a linear regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
8:LinReg
(a+bx)

LinReg(ax+b)

LinReg(ax+b) [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a linear regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
4:LinReg
(ax+b)

LinRegTInt

LinRegTInt [*Xlistname*,*Ylistname*,*freqlist*,*confidence level*, *regequ*]

Performs a linear regression and computes the t confidence interval for the slope coefficient b.

† stats
TESTS
G:LinRegTInt

LinRegTTest

LinRegTTest
[*Xlistname*,*Ylistname*,*freqlist*,*alternative*,*regequ*]

Performs a linear regression and a *t*-test. *alternative*=**−1** is <; *alternative*=**0** is ≠; *alternative*=**1** is >.

† stats
TESTS
F:LinRegTTest

ΔList(

ΔList(*list*)

Returns a list containing the differences between consecutive elements in *list*.

2nde [listes]
OPS
7: Δ List(

List▶matr(

List▶matr(*listname1*,...,*listname n*,*matrixname*)

Fills *matrixname* column by column with the elements from each specified *listname*.

2nde [listes]
OPS
0:List ▶ matr
(

ln(

ln(*value*)

Returns the natural logarithm of a real or complex number, expression, or list.

ln

LnReg

LnReg [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a logarithmic regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
9:LnReg

log(

log(*value***)** log

Returns logarithm of a real or complex number, expression, or list.

logBASE(

logBASE(*value, base***)** math

Returns the logarithm of a specifed value determined from a specified base: logBASE(value, base). **A: logBASE**

Logistic

Logistic [*Xlistname***,***Ylistname***,***freqlist***,***regequ***]** stats **CALC B:Logistic**

Fits a logistic regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

Manual-Fit

Manual-Fit[*equname,color#,line style#*]

Fits a linear equation to a scatter plot with specified color and line style.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

line style #: 1-4.

stats
CALC
D:Manual-Fit

MATHPRINT

MATHPRINT

Displays most entries and answers the way they are displayed in textbooks, such as $\frac{1}{2}+\frac{3}{4}$.

mode
MATHPRINT

Matr▸list(

Matr▸list(*matrix,listnameA,...,listname n***)**

Fills each *listname* with elements from each column in *matrix*.

2nde [listes]
OPS
A:Matr▸list(

Matr▸list(

Matr▸list(*matrix,column#,listname***)**

Fills a *listname* with elements from a specified *column#* in *matrix*.

2nde [listes]
OPS
A:Matr▸list(

max(

max(*valueA,valueB***)**

Returns the larger of *valueA* and *valueB*.

math
NUM
7:max(

max(

max(*list***)**

Returns the larger of *valueA* and *valueB*.

math
NUM
7:max(

max(

max(*list*)

Returns largest real or complex element in *list*.

2nde [listes]
MATH
2:max(

max(

max(*listA*,*listB*)

Returns a real or complex list of the larger of each pair of elements in *listA* and *listB*.

2nde [listes]
MATH
2:max(

max(

max(*value*,*list*)

Returns a real or complex list of the larger of *value* or each *list* element.

2nde [listes]
MATH
2:max(

mean(

mean(*list*[,*freqlist*])

Returns the mean of *list* with frequency *freqlist*.

2nde [listes]
MATH
3:mean(

median(

median(*list*[,*freqlist*])

Returns the median of *list* with frequency *freqlist*.

2nde [listes]
MATH
4:median(

Med-Med

Med-Med [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a median-median model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
3:Med-Med

Menu(

Menu("*title*","*text1*",*label1*[,...,"*text7*",*label7*])

Generates a menu of up to seven items during program execution.

† prgm
CTL
C:Menu(

min(

min(*valueA*,*valueB*)

Returns smaller of *valueA* and *valueB*.

[math]
NUM
6:min(

min(

min(*list*)

Returns smallest real or complex element in *list*.

[2nde] [listes]
MATH
1:min(

min(

min(*listA*,*listB*)

Returns real or complex list of the smaller of each pair of elements in *listA* and *listB*.

[2nde] [listes]
MATH
1:min(

min(

min(*value*,*list*)

Returns a real or complex list of the smaller of *value* or each *list* element.

[2nde] [listes]
MATH
1:min(

ModBoxplot

ModBoxplot Plot#(*type*,*Xlist*,[,*freqlist*,*color#*])

Used as the "type" argument in the command.

Where # gives Plot1, Plot2 or Plot3.

† [2nde]
[graph stats]
TYPE

N

nCr

valueA **nCr** *valueB*

Returns the number of combinations of *valueA* taken *valueB* at a time.

[math]
PRB
3:nCr

nCr

value **nCr** *list*

Returns a list of the combinations of *value* taken each element in *list* at a time.

[math]
PRB
3:nCr

nCr

list **nCr** *value*

Returns a list of the combinations of each element in *list* taken *value* at a time.

[math]
PRB
3:nCr

nCr

listA **nCr** *listB*

Returns a list of the combinations of each element in *listA* taken each element in *listB* at a time.

[math]
PRB
3:nCr

n/d

n/d

Displays results as a simple fraction.

[alpha] [f1]
1: n/d
or

[math]
NUM
D: n/d
or

[math]
FRAC
1:n/d

nDeriv(

nDeriv(*expression*,*variable*,*value*[,ε])

When command is used in Classic mode, returns approximate numerical derivative of *expression* with respect to *variable* at *value*, with specific tolerance ε.

In MathPrint mode, numeric derivative template pastes and uses default tolerance ε.

[math]
MATH
8:nDeriv(

▸ n/d ◂▸ Un/d

▸ n/d ◂▸ Un/d

Converts the results from a fraction to mixed number or from a mixed number to a fraction, if applicable.

[alpha] [f1]
3: ▸ n/d ◂▸ Un/d

or

▸ n/d ◂▸ Un/d

math
NUM
A: ▸ n/d◂▸
Un/d

or

math
FRAC
4: ▸ n/d ◂
▸Un/d

▸Nom(

▸Nom(*effective rate*, *compounding periods*)

Computes the nominal interest rate.

2nde [apps]
1:Finance
CALC
B: ▸ Nom(

Normal

Normal

Sets normal display mode.

† mode
Normal

normalcdf(

normalcdf(*lowerbound*,*upperbound*[,μ,σ])

Computes the normal distribution probability between *lowerbound* and *upperbound* for the specified μ and σ.

2nde [distrib]
DISTR
2:normalcdf(

normalpdf(

normalpdf(*x*[,μ,σ])

Computes the probability density function for the normal distribution at a specified *x* value for the specified μ and σ.

2nde [distrib]
DISTR
1:normalpdf(

NormProbPlot

NormProbPlot Plot#(*type*,*Xlist*,[,*freqlist*,*color#*])

Used as the "type" argument in the command

Where # gives Plot1, Plot2 or Plot3.

† 2nde
[graph stats]

NormProbPlot

TYPE

not(

not(*value***)**

Returns **0** if *value* is 0. *value* can be a real number, expression, or list.

2nde [tests]
LOGIC
4:not(

nPr

valueA **nPr** *valueB*

Returns the number of permutations of *valueA* taken *valueB* at a time.

math
PRB
2:nPr

nPr

value **nPr** *list*

Returns a list of the permutations of *value* taken each element in *list* at a time.

math
PRB
2:nPr

nPr

list **nPr** *value*

Returns a list of the permutations of each element in *list* taken *value* at a time.

math
PRB
2:nPr

nPr

listA **nPr** *listB*

Returns a list of the permutations of each element in *listA* taken each element in *listB* at a time.

math
PRB
2:nPr

npv(

npv(*interest rate***,***CF0***,***CFList***[,***CFFreq***])**

Computes the sum of the present values for cash inflows and outflows.

2nde [apps]
1:Finance
CALC
7:npv(

O

OpenLib(

OpenLib(

Extends TI-Basic. (Not available.)

† prgm
CTL
J:OpenLib(

or

valueA **or** *valueB*

Returns 1 if *valueA* or *valueB* is 0. *valueA* and *valueB* can be real numbers, expressions, or lists.

2nde [tests]
LOGIC
2:or

Output(

Output(*row*,*column*,"*text*"**)**

Displays *text* beginning at specified *row* and *column* of the home screen.

† prgm
I/O
6:Output(

Output(

Output(*row*,*column*,*value***)**

Displays *value* beginning at specified *row* and *column* of the home screen.

† prgm
I/O
6:Output(

P

Param

Param

Sets parametric graphing mode.

† mode
Par

Pause

Pause

Suspends program execution until you press ENTER.

† prgm
CTL
8:Pause

Pause

Pause [*value***]**

Displays *value*; suspends program execution until you press ENTER.

† prgm
CTL

Pause

8:Pause

Pause

Pause [*value, time*]

Displays value on the current home screen and execution of the program continues after the time period specified. For time only, use Pause *"",time* where the value is a blank string. Time is in seconds.

Pause *value,time.*

† prgm
CTL
8:Pause

piecewise

piecewise(

New piecewise function to support entry of functions as they are seen in textbook. This command can be found in math MATH **B:piecewise(**

math
▲ or ▼ to scroll to **B:piecewise(**

Plot1(Plot2(Plot3(

Plot#(*type***,***Xlist***,***Ylist*[**,***mark,color#*])

Defines **Plot***#* (**1**, **2**, or **3**) of *type* **Scatter** or **xyLine** for *Xlist* and *Ylist* using *mark* and *color.*

Color#: 10 - 24 or color name pasted from [vars] COLOR.

Note: *Xlist* and *Ylist* represent the Xlist and Ylist names.

† 2nde [graph stats]
STAT PLOTS
1:Plot1
2:Plot2
3:Plot3

Plot1(Plot2(Plot3(

Plot#(*type***,***Xlist***,**[,*freqlist,color#*])

Defines **Plot***#* (**1**, **2**, or **3**) of *type* **Histogram** or **Boxplot** for *Xlist* with frequency *freqlist* and color **#**.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

Note: *Xlist* represents the Xlist name.

† 2nde [graph stats]
STAT PLOTS
1:Plot1
2:Plot2
3:Plot3

Plot1(Plot2(Plot3(

Plot#(*type***,***Xlist***,**[,*freqlist,mark,color#*])

Defines **Plot***#* (**1**, **2**, or **3**) of *type* **ModBoxplot** for *Xlist* with frequency *freqlist* using *mark* and *color #.*

† 2nde [graph stats]
STAT PLOTS
1:Plot1

Plot1(Plot2(Plot3(

Color#: 10 - 24 or color name pasted from [vars] COLOR.

Note: *Xlist* represents the Xlist name.

2:Plot2
3:Plot3

Plot1(Plot2(Plot3(

Plot#(*type*,*datalist*,[,*data axis*,*mark*,*color#*])

Defines **Plot#** (**1**, **2**, or **3**) of *type* **NormProbPlot** for *datalist* on *data axis* using *mark* and *color* # *data axis* can be **X** or **Y**.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

Note: *datalist* represents the datalist name.

† 2nde [graph stats]
STAT PLOTS
1:Plot1
2:Plot2
3:Plot3

PlotsOff

PlotsOff [1,2,3]

Deselects all stat plots or one or more specified stat plots (**1**, **2**, or **3**).

2nde [graph stats]
STAT
PLOTS
4:PlotsOff

PlotsOn

PlotsOn [1,2,3]

Selects all stat plots or one or more specified stat plots (**1**, **2**, or **3**).

2nde [graph stats]
STAT
PLOTS
5:PlotsOn

Pmt_Bgn

Pmt_Bgn

Specifies an annuity due, where payments occur at the beginning of each payment period.

2nde [apps]
1:Finance
CALC
F:Pmt_Bgn

Pmt_End

Pmt_End

Specifies an ordinary annuity, where payments occur at the end of each payment period.

2nde [apps]
1:Finance
CALC
E:Pmt_End

poissoncdf(

poissoncdf(μ**,**x**)**

Computes a cumulative probability at x for the discrete Poisson distribution with specified mean μ.

2nde [distrib]
DISTR

D:poissoncdf(

poissonpdf(

poissonpdf(μ**,**x**)**

Computes a probability at x for the discrete Poisson distribution with the specified mean μ.

2nde [distrib]
DISTR

C:poissonpdf(

Polar

Polar

Sets polar graphing mode.

† mode
Polar

▶Polar

complex value **▶Polar**

Displays *complex value* in polar format.

math
CMPLX
7: ▶ Polar

PolarGC

PolarGC

Sets polar graphing coordinates format.

† 2nde [format]
PolarGC

prgm

prgm*name*

Executes the program *name*.

† prgm
CTRL
D:prgm

ΣPrn(

ΣPrn(*pmt1*,*pmt2*[,*roundvalue*]**)**

Computes the sum, rounded to *roundvalue*, of the principal amount between *pmt1* and *pmt2* for an amortization schedule.

2nde [apps]
1:Finance
CALC
0: Σ Prn(

prod(

prod(*list*[**,***start*,*end*]**)**

Returns product of *list* elements between *start* and *end*

2nde [listes]
MATH
6:prod(

Prompt

Prompt *variableA*[**,***variableB*,...,*variable n*]

Prompts for value for *variableA*, then *variableB*, and so on.

† prgm
I/O
2:Prompt

1-PropZInt(

1-PropZInt(*x*,*n*[**,***confidence level*]**)**

Computes a one-proportion *z* confidence interval.

† stats
TESTS
A:1-PropZInt(

2-PropZInt(

2-PropZInt(*x1*,*n1*,*x2*,*n2*[**,***confidence level*]**)**

Computes a two-proportion *z* confidence interval.

† stats
TESTS
B:2-PropZInt(

1-PropZTest(

1-PropZTest(*p0*,*x*,*n*[**,***alternative*,*drawflag, color#*]**)**

Computes a one-proportion *z* test. *alternative*=**-1** is <; *alternative*=**0** is ; *alternative*=**1** is >. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats
TESTS
5:1-PropZTest
(

2-PropZTest(

2-PropZTest(*x1*,*n1*,*x2*,*n2*[,*alternative*,*drawflag*, *color#*])

Computes a two-proportion *z* test. *alternative*=**-1** is <; *alternative*=**0** is ; *alternative*=**1** is >. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† [stats]
TESTS
6:2-PropZTest
(

Pt-Change(

Pt-Change(*x*,*y*[,*color#*])

Toggles a point on or off at (*x*,*y*) on the graph area. Off will be in the Background color and On will be the specified

Color#: 10 - 24 or color name pasted from [vars] COLOR.

[2nde] [dessin]
POINTS
3:Pt-Change(

Pt-Off(

Pt-Off(*x*,*y*[,*mark*])

Erases a point at (*x*,*y*) on the graph area using *mark*. The Off state may be the background color determined by the *ImageVar* or *color* setting.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

[2nde] [dessin]
POINTS
2:Pt-Off(

Pt-On(

Pt-On(*x*,*y*[,*mark*,*color#*])

Draws a point at (*x*,*y*) on the graph area using *mark* and the specified *color#*.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

[2nde] [dessin]
POINTS
1:Pt-On(

PwrReg

PwrReg [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a power regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

[stats]
CALC
A:PwrReg

Pxl-Change(

Pxl-Change(*row*,*column*[,*color#*]**)**

Toggles Off to On in the graph area: with specified *color#*
Toggles On to Off in the graph area: Off will display the set Background Image Var or Color.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [dessin]
POINTS
6:Pxl-Change(

Pxl-Off(

Pxl-Off(*row*,*column*)

The Off state will display the set Background Image Var or COLOR.

2nde [dessin]
POINTS
5:Pxl-Off(

Pxl-On(

Pxl-On(*row*,*column*[,*color#*]**)**

Draws pixel on the graph area at (*row*,*column*) in the specified color.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [dessin]
POINTS
4:Pxl-On(

pxl-Test(

pxl-Test(*row*,*column*)

Returns 1 if pixel (*row*, *column*) is on, 0 if it is off;

2nde [dessin]
POINTS
7:pxl-Test(

P▶Rx(

P▶Rx(*r*,θ**)**

Returns **X**, given polar coordinates *r* and θ or a list of polar coordinates.

2nde [angle]
ANGLE
7:P▶Rx(

P▶Ry(

P▶Ry(*r*,θ**)**

Returns **Y**, given polar coordinates *r* and θ or a list of polar coordinates.

2nde [angle]
ANGLE
8:P▶Ry(

Q

QuadReg

QuadReg [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a quadratic regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
5:QuadReg

QuartReg

QuartReg [*Xlistname*,*Ylistname*,*freqlist*,*regequ*]

Fits a quartic regression model to *Xlistname* and *Ylistname* with frequency *freqlist*, and stores the regression equation to *regequ*.

stats
CALC
7:QuartReg

R

Radian

Radian

Sets radian angle mode.

† mode
Radian

rand

rand[(*numtrials***)]**

Returns a random number between 0 and 1 for a specified number of trials *numtrials*.

math
PRB
1:rand

randBin(

randBin(*numtrials*,*prob*[,*numsimulations*])

Generates and displays a random real number from a specified Binomial distribution.

math
PRB
7:randBin(

randInt(

randInt(*lower*,*upper* **[**,*numtrials*]**)**

Generates and displays a random integer within a range specified by *lower* and *upper* integer bounds for a specified number of trials *numtrials*.

math
PRB
5:randInt(

randIntNoRep(

randIntNoRep(*lowerint*,*upperint* [,*numelements*])

Returns a random ordered list of integers from a lower integer to an upper integer which may include the lower integer and upper integer. If the optional argument *numelements* is specified, the first *numelements* are listed. The first *numelements* term in the list of random integers are displayed.

math
PRB
8:randIntNoRep(

randM(

randM(*rows*,*columns***)**

Returns a random matrix of *rows* × *columns*.

Max rows x columns = 400 matrix elements.

matrice
MATH
6:randM(

randNorm(

randNorm(μ,σ**[**,*numtrials*]**)**

Generates and displays a random real number from a specified Normal distribution specified by μ and σ for a specified number of trials *numtrials*.

math
PRB
6:randNorm(

re^θi

re^θi

Sets the mode to polar complex number mode (**re^θi**).

† mode
r *e* **^** θ *i*

Real

Real

Sets mode to display complex results only when you enter complex numbers.

† mode
Real

real(

real(*value***)**

Returns the real part of a complex number or list of complex numbers.

[math]
CPLX
2:real(

RecallGDB

RecallGDB *n*

Restores all settings stored in the graph database variable **GDB***n*.

[2nde] [dessin]
STO

4:RecallGDB

RecallPic

RecallPic *n*

Displays the graph and adds the picture stored in **Pic***n*.

[2nde]
[dessin]
STO
2:RecallPic

▶Rect

complex value **▶Rect**

Displays *complex value* or list in rectangular format.

[math]
CMPLX
6: ▶ Rect

RectGC

RectGC

Sets rectangular graphing coordinates format.

† [2nde]
[format]
RectGC

ref(

ref(*matrix***)**

Returns the row-echelon form of a *matrix*.

[matrice]
MATH
A:ref(

remainder(

remainder(*dividend, divisor***)**

Reports the remainder as a whole number from a division of two whole numbers where the divisor is not zero.

[math]
NUM
0:remainder(

remainder(

remainder(*list, divisor***)**

Reports the remainder as a whole number from a division of two lists where the divisor is not zero.

math
NUM
0:remainder(

remainder(

remainder(*dividend, list***)**

Reports the remainder as a whole number from a division of two whole numbers where the divisor is a list.

math
NUM
0:remainder(

remainder(

remainder(*list, list***)**

Reports the remainder as a whole number from a division of two lists.

math
NUM
0:remainder
(

Repeat

Repeat*condition***:***commands***:End:***commands*

Executes *commands* until *condition* is true.

† prgm
CTL
6:Repeat

Return

Return

Returns to the calling program.

† prgm
CTL
E:Return

round(

round(*value***[,***#decimals***])**

Returns a number, expression, list, or matrix rounded to *#decimals* (9).

math
NUM
2:round(

***row(**

row(value***,***matrix***,***row***)**

Returns a matrix with *row* of *matrix* multiplied by *value* and stored in *row*.

matrice
MATH
E: * row(

row+(

row+(*matrix*,*rowA*,*rowB*)

Returns a matrix with *rowA* of *matrix* added to *rowB* and stored in *rowB*.

matrice
MATH
D:row+(

*row+(

row+(value*,*matrix*,*rowA*,*rowB*)

Returns a matrix with *rowA* of *matrix* multiplied by *value*, added to *rowB*, and stored in *rowB*.

matrice
MATH
F: * row+(

rowSwap(

rowSwap(*matrix*,*rowA*,*rowB*)

Returns a matrix with *rowA* of *matrix* swapped with *rowB*.

matrice
MATH
C:rowSwap(

rref(

rref(*matrix*)

Returns the reduced row-echelon form of a *matrix*.

matrice
MATH
B:rref(

R▶Pr(

R▶Pr(*x*,*y*)

Returns **R**, given rectangular coordinates *x* and *y* or a list of rectangular coordinates.

2nde [angle]
ANGLE
5:R▶Pr(

R▶Pθ(

R▶Pθ(*x*,*y*)

Returns θ, given rectangular coordinates *x* and *y* or a list of rectangular coordinates.

2nde [angle]
ANGLE
6:R▶Pθ(

2-SampFTest

2-SampFTest
[*listname1* **,** *listname2* **,***freqlist1***,***freqlist2***,***alternative***,***drawflag***,***color#***]**

† stats
TESTS
E:2-Samp F Test

Performs a two-sample F test. *alternative*=**-1** is **<**; *alternative*=**0** is ≠; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2-SampFTest

2-SampFTest*Sx1***,***n1***,***Sx2***,***n2*
[,*alternative***,***drawflag***,***color#***]**

† stats
TESTS
E:2-Samp F Test

Performs a two-sample F test. *alternative*=**-1** is **<**; *alternative*=**0** is ≠; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2-SampTInt

2-SampTInt
[*listname1***,***listname2***,***freqlist1***,***freqlist2***,***confidence level***,***pooled***]**
(Data list input)

† stats
TESTS
0:2-SampTInt

Computes a two-sample *t* confidence interval. *pooled*=**1** pools variances; *pooled*=**0** does not pool variances.

2-SampTInt

2-SampTInt$\bar{x}$*1***,***Sx1***,***n1***,**$\bar{x}$*2***,***Sx2***,***n2***[,***confidence level***,***pooled***]**
(Summary stats input)

† stats
TESTS
0:2-SampTInt

Computes a two-sample *t* confidence interval. *pooled*=**1** pools variances; *pooled*=**0** does not pool variances.

2-SampTTest

2-SampTTest
[
listname1
,
listname2
,
freqlist1
,*freqlist2*,*alternative*,*pooled*,*drawflag*,*color#*]**)**

† stats
TESTS
4:2-SampTTest

Computes a two-sample *t* test. *alternative*=**-1** is **<**; *alternative*=**0** is ; *alternative*=**1** is **>**. *pooled*=**1** pools variances; *pooled*=**0** does not pool variances. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2-SampTTest

2-SampTTest $\bar{x}$*1*,*Sx1*,*n1*,*v2*,*Sx2*,*n2*
[,*alternative*,*pooled*,*drawflag*,*color#*]**)**

† stats
TESTS
4:2-SampTTest

Computes a two-sample *t* test. *alternative*=**-1** is **<**; *alternative*=**0** is ; *alternative*=**1** is **>**. *pooled*=**1** pools variances; *pooled*=**0** does not pool variances. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2-SampZInt(

2-SampZInt(σ_1,σ_2
[,*listname1*,*listname2*,*freqlist1*,*freqlist2*,*confidence level*]**)**
(Data list input)

† stats
TESTS
9:2-SampZInt(

Computes a two-sample *z* confidence interval.

2-SampZInt(

2-SampZInt(σ_1,σ_2,$\bar{x}$*1*,*n1*,$\bar{x}$*2*,*n2***[**,*confidence level*]**)**
(Summary stats input)

† stats
TESTS
9:2-SampZInt(

Computes a two-sample *z* confidence interval.

2-SampZTest(

2-SampZTest(σ_1,σ_2
[,
listname1
,
listname2
,freqlist1,freqlist2,alternative,drawflag,color#
])

Computes a two-sample *z* test. *alternative*=**-1** is **<**; *alternative*=**0** is **≠**; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† [stats]
TESTS
3:2-SampZTest(

2-SampZTest(

2-SampZTest(σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*
[,*alternative*,*drawflag*,*color#*]**)**

Computes a two-sample *z* test. *alternative*=**-1** is **<**; *alternative*=**0** is **≠**; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† [stats]
TESTS
3:2-SampZTest(

Scatter

Scatter Plot#*(type,Xlist*,[,*freqlist,color#*])

Used as the "type" argument in the command

Where # gives Plot1, Plot2 or Plot3.

† [2nde] [graph stats]
TYPE

Sci

Sci

Sets scientific notation display mode.

† [mode]
Sci

Select(

Select(*Xlistname*,*Ylistname*)

Selects one or more specific data points from a scatter plot or xyLine plot (only), and then store's the selected data points to two new lists, *Xlistname* and *Ylistname*.

[2nde] [listes]
OPS
8:Select(

seq(

seq(*expression*,*variable*,*begin*,*end*[,*increment*]**)**

2nde [listes]
OPS
5:seq(

Returns list created by evaluating *expression* with regard to *variable*, from *begin* to *end* by *increment*.

SEQ(*n*)

Seq(*n*)

† mode

In sequence mode, **SEQ(*n*)** sets the sequence editor type to enter sequence functions, u, v, or w, as a function of the independent variable *n*. Can also be set from the Y= editor in **SEQ mode**.

SEQ(*n*)

SEQ(*n*+1)

Seq(*n*+1)

† mode

In sequence mode, **SEQ(*n*+1)** sets the sequence editor type to enter sequence functions, u, v, or w, as a function of the independent variable *n*+1. Can also be set from the Y= editor in **SEQ mode**.

SEQ(*n*+1)

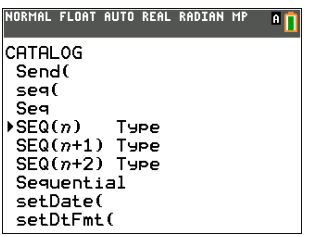
SEQ(*n*+2)

Seq(*n*+2)

† mode

In sequence mode, **SEQ(*n*+2)** sets the sequence editor type to enter sequence functions, u, v, or w, as a function of the independent variable *n*+2. Can also be set from the Y= editor in **SEQ mode**.

SEQ(*n*+2)



Note: "Type" will NOT be included in the TIC CE PE syntax

On the device, "Type" does not paste and is similar to how the device displays, for example, DEC Answers where Answers appears in [catalog] but does not paste.

.

Seq

Seq

† mode
Seq

Sets sequence graphing mode.

Sequential

Sequential

† mode
Sequential

Sets mode to graph functions sequentially.

setDate(

setDate(*year*,*month*,*day*)

2nde [catalog]
setDate(

Sets the date using a year, month, day format. The *year* must be 4 digits; *month* and *day* can be 1 or 2 digit.

setDtFmt(

setDtFmt(*integer*)

2nde
[catalog]
setDtFmt(

Sets the date format.

1 = M/D/Y
2 = D/M/Y
3 = Y/M/D

setTime(

setTime(*hour,minute, second***)**

Sets the time using an hour, minute, second format. The *hour* must be in 24 hour format, in which 13 = 1 p.m.

[2nde] [catalog]
setTime(

setTmFmt(

setTmFmt(*integer***)**

Sets the time format.

12 = 12 hour format
24 = 24 hour format

[2nde] [catalog]
setTmFmt(

SetUpEditor

SetUpEditor

Removes all list names from the stat list editor, and then restores list names **L1** through **L6** to columns **1** through **6**.

[stats]
EDIT
5:SetUpEditor

SetUpEditor

SetUpEditor *listname1*[**,***listname2*,...,*listname20*]

Removes all list names from the stat list editor, then sets it up to display one or more *listnames* in the specified order, starting with column **1**.

[stats]
EDIT
5:SetUpEditor

Shade(

Shade(*lowerfunc*,*upperfunc*
[,*Xleft*,*Xright*,*pattern*,*patres*,*color#*])

Draws *lowerfunc* and *upperfunc* in terms of **X** on the current graph and uses *pattern* and *patres* to shade and color the area bounded by *lowerfunc*, *upperfunc*, *Xleft*, and *Xright*. *lowerfunc* and *upperfunc* are shaded in the same specified color.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

[2nde] [dessin]
DRAW
7:Shade(

Shadeχ^2(

Shadeχ^2(*lowerbound*,*upperbound*,*df*[,*color#*])

Draws the density function for the χ^2 distribution specified by degrees of freedom *df*, and shades and colors the area between *lowerbound* and *upperbound*.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [distrib]
DRAW
3:Shade χ^2(

ShadeF(

ShadeF(*lowerbound*,*upperbound*,*numerator df*,*denominator df*[,*color#*])

Draws the density function for the F distribution specified by *numerator df* and *denominator df* and shades and colors the area between *lowerbound* and *upperbound*.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [distrib]
DRAW
4:ShadeF(

ShadeNorm(

ShadeNorm(*lowerbound*,*upperbound*[,μ,σ,*color#*])

Draws the normal density function specified by μ and σ and shades and colors the area between *lowerbound* and *upperbound*.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [distrib]
DRAW
1:ShadeNorm(

Shade_t(

Shade_t(*lowerbound*,*upperbound*,*df*[,*color#*])

Draws the density function for the Student-t distribution specified by degrees of freedom df, and shades or colors the area between *lowerbound* and *upperbound*.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [distrib]
DRAW
2:Shade_t(

Simul

Simul

Sets mode to graph functions simultaneously.

† mode
Simul

sin(

sin(*value***)**

Returns the sine of a real number, expression, or list.

2nde [catalog]
sin(

$\sin^{-1}$(

$\sin^{-1}$(*value***)**

Returns the arcsine of a real number, expression, or list.

2nde [catalog]
sin-1(

sinh(

sinh(*value***)**

Returns the hyperbolic sine of a real number, expression, or list.

2nde [catalog]
sinh(

$\sinh^{-1}$ (

$\sinh^{-1}$ (*value***)**

Returns the hyperbolic arcsine of a real number, expression, or list.

2nde [catalog]
sinh $^{-1}$ (

SinReg

SinReg
[*iterations*,*Xlistname*,*Ylistname*,*period*,*regequ***]**

Attempts *iterations* times to fit a sinusoidal regression model to *Xlistname* and *Ylistname* using a *period* guess, and stores the regression equation to *regequ*.

stats
CALC
C:SinReg

solve(

solve(*expression*,*variable*,*guess*,{*lower*,*upper*})

Solves *expression* for *variable*, given an initial *guess* and *lower* and *upper* bounds within which the solution is sought.

† [math]
MATH
0:solve(

SortA(

SortA(*listname*)

Sorts elements of *listname* in ascending order.

[2nde] [listes]
OPS
1:SortA(

SortA(

SortA(*keylistname*,*dependlist1*
[,*dependlist2*,...,*dependlist n*]**)**

Sorts elements of *keylistname* in ascending order, then sorts each *dependlist* as a dependent list.

[2nde] [listes]
OPS
1:SortA(

SortD(

SortD(*listname*)

Sorts elements of *listname* in descending order.

[2nde] [listes]
OPS
2:SortD(

SortD(

SortD(*keylistname*,*dependlist1***[**,*dependlist2*,...,
dependlist n]**)**

Sorts elements of *keylistname* in descending order, then sorts each *dependlist* as a dependent list.

[2nde] [listes]
OPS
2:SortD(

startTmr

startTmr

Starts the clock timer. Store or note the displayed value, and use it as the argument for **checkTmr( )** to check the elapsed time.

[2nde] [catalog]
startTmr

STATWIZARD OFF

STATWIZARD OFF

Disables wizard syntax help for statistical commands, distributions, and seq(.

[2nde] [catalog]
STATWIZARD OFF

STATWIZARD ON

STATWIZARD ON

Enables wizard syntax help for statistical commands, distributions, and seq(.

[2nde] [catalog]
STATWIZARD ON(

stdDev(

stdDev(*list*[**,***freqlist*]**)**

Returns the standard deviation of the elements in *list* with frequency *freqlist*.

[2nde] [listes]
MATH
7:stdDev(

Stop

Stop

Ends program execution; returns to home screen.

† [prgm]
CTL
F:Stop

Store →

Store: *value*→*variable*

Stores *value* in *variable*.

[sto→]

StoreGDB

StoreGDB *n*

Stores current graph in database **GDB***n*.

[2nde] [dessin]
STO
3:StoreGDB

StorePic

StorePic *n*

Stores current picture in picture **Pic***n*.

2nde [dessin]
STO
1:StorePic

String▶Equ(

String▶Equ(*string*,**Y=** *var*)

Converts *string* into an equation and stores it in **Y=** *var*.

string can be a string or string variable.

String▶Equ(is the inverse of **Equ▶String(**.

† [prgm]
I/O
D:String>Equ(

sub(

sub(*string*,*begin*,*length*)

Returns a string that is a subset of another *string*, from *begin* to *length*.

2nde [catalog]
sub(

sum(

sum(*list*[,*start*,*end*])

Returns the sum of elements of *list* from *start* to *end*.

2nde [listes]
MATH
5:sum(

summation Σ(

Σ(*expression*[,*start*,*end*])

Classic command as shown.

In MathPrint™ the summation entry template displays and returns the sum of elements of *list* from *start* to *end*, where *start* <= *end*.

[math]
NUM
0: summation Σ(

T

tan(

tan(*value*)

Returns the tangent of a real number, expression, or list.

2nde
[catalog]
tan(

tan^{-1}(

tan^{-1}(*value*)

Returns the arctangent of a real number, expression, or list.

2nde [catalog]
tan-1(

Tangent(

Tangent(*expression*,*value*[,*color#*,*linestyle#*])

Draws a line tangent to *expression* at **X**=*value* with specified *color#*: 10-24 and line style *linestyle #:* 1-2.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

2nde [dessin]
DRAW
5:Tangent(

tanh(

tanh(*value*)

Returns hyperbolic tangent of a real number, expression, or list.

2nde [catalog]
tanh(

tanh^{-1}(

tanh^{-1}(*value*)

Returns the hyperbolic arctangent of a real number, expression, or list.

2nd [catalog]
tanh^{-1}(

tcdf(

tcdf(*lowerbound*,*upperbound*,*df*)

Computes the Student-*t* distribution probability between *lowerbound* and *upperbound* for the specified degrees of freedom *df*.

2nde [distrib]
DISTR
6:tcdf(

Text(

Text(*row*,*column*,*text1*,*text2*,...,*text n*)

Writes *text* on graph beginning at pixel (*row*,*column*), where 0 *row* 164 and 0 *column* 264.

Full mode, row must be <=148; column must be 256

Horiz mode, row must be row<=66 and column must be <=256

G-T mode, row must be row <=126; column must be 176

2nde [dessin]
DRAW
0:Text(

TextColor(

TextColor([*color#*])

† 2nde [dessin]

TextColor(

Set text color prior to using the **Text(** command.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

DRAW
A:TextColor(

Then

Then
See **If:Then**

Thick

Thick

Resets all Y=editor line-style settings to Thick.

† mode
Thick

Thin

Thin

Resets all Y=editor line-style settings to Thin.

† mode
Thin

Time

Time

Sets sequence graphs to plot with respect to time.

† 2nde [format]
Time

timeCnv(

timeCnv(*seconds***)**

Converts seconds to units of time that can be more easily understood for evaluation. The list is in *{days,hours,minutes,seconds}* format.

2nde [catalog]
timeCnv

TInterval

TInterval [*listname,freqlist,confidence level***]**
(Data list input)

Computes a *t* confidence interval.

† stats
TESTS
8:TInterval

TInterval

TInterval $\bar{x}$,*Sx*,*n*[,*confidence level*]

† stats

TInterval

(Summary stats input)

Computes a *t* confidence interval.

TESTS
8:TInterval

toString(

toString((*value***[,***format***])**

Converts value to a string where *value* can be real, complex, an evaluated expression, list, or matrix. String *value* displays in classic *format* (0) following the mode setting AUTO/DEC or in decimal *format* (1).

† prgm
I/O
C:toString(

tpdf(

tpdf(*x***,***df***)**

Computes the probability density function (pdf) for the Student-*t* distribution at a specified *x* value with specified degrees of freedom *df*.

2nde [distrib]
DISTR
5:tpdf(

Trace

Trace

Displays the graph and enters **TRACE** mode.

trace

T-Test

T-Test μ*0*
[,*listname***,***freqlist***,***alternative***,***drawflag***,***color#***])**
(Data list input)

Performs a *t* test with frequency *freqlist*. *alternative*=**-1** is **<**; *alternative*=**0** is **≠**; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats
TESTS
2:T-Test

T-Test

T-Test μ*0*,$\bar{x}$,*Sx*,*n*[,*alternative*,*drawflag*,*color#*])

Performs a *t* test with frequency *freqlist*. *alternative*=**-1** is **<**; *alternative*=**0** is **≠**; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats
TESTS
2:T-Test

tvm_FV

tvm_FV[(N,I%,*PV*,*PMT*,*P/Y*,*C/Y*)]

Computes the future value.

2nde [apps]
1:Finance
CALC
6:tvm_FV

tvm_I%

tvm_I%[(N,*PV*,*PMT*,*FV*,*P/Y*,*C/Y*)]

Computes the annual interest rate.

2nde [apps]
1:Finance
CALC
3:tvm_ I%

tvm_N

tvm_N[(I%,*PV*,*PMT*,*FV*,*P/Y*,*C/Y*)]

Computes the number of payment periods.

2nde [apps]
1:Finance
CALC
5:tvm_ N

tvm_Pmt

tvm_Pmt[(N,I%,*PV*,*FV*,*P/Y*,*C/Y*)]

Computes the amount of each payment.

2nde [apps]
1:Finance
CALC
2:tvm_Pmt

tvm_PV

tvm_PV[(N,I%,*PMT*,*FV*,*P/Y*,*C/Y*)]

Computes the present value.

2nde [apps]
1:Finance
CALC
4:tvm_PV

U

UnArchive

UnArchive *variable*

Moves the specified variables from the user data archive memory to RAM.

To archive variables, use **Archive**.

2nde [mém]
6:UnArchive

Un/d

Un/d

Displays results as a mixed number, if applicable.

math
NUM
C: Un/d

or

math
FRAC
2:Un/d

uvAxes

uvAxes

Sets sequence graphs to plot **u(***n***)** on the x-axis and **v(***n***)** on the y-axis.

† 2nde [format]
uv

uwAxes

uwAxes

Sets sequence graphs to plot **u(***n***)** on the x-axis and **w(***n***)** on the y-axis.

† 2nde [format]
uw

V

1-VarStats

1-VarStats [*Xlistname*,*freqlist***]**

Performs one-variable analysis on the data in *Xlistname* with frequency *freqlist*.

stats
CALC
1:1-Var Stats

2-VarStats

2-VarStats [*Xlistname*,*Ylistname*,*freqlist***]**

Performs two-variable analysis on the data in *Xlistname* and *Ylistname* with frequency *freqlist*.

stats
CALC
2:2-Var Stats

variance(

variance(*list***[,***freqlist***])**

Returns the variance of the elements in *list* with frequency *freqlist*.

2nde [listes]
MATH
8:variance(

Vertical

Vertical *x*[,*color#*,*linestyle#*]

Draws a vertical line at *x* with specified color and line style.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

line style #: 1-4.

2nde [dessin]
DRAW
4:Vertical

vwAxes

vwAxes

Sets sequence graphs to plot **v(***n***)** on the x-axis and **w(***n***)** on the y-axis.

† 2nde [format]
vw

W

Wait

Wait*time*

Suspends execution of a program for a given time. Maximum time is 100 seconds.

† prgm
CTL
A:Wait

Web

Web

Sets sequence graphs to trace as webs.

† 2nde [format]
Web

:While

:While*condition*:*commands*
:End:*command*

Executes *commands* while *condition* is true.

† prgm
CTL
5:While

X

xor

valueA **xor** *valueB*

Returns 1 if only *valueA* or *valueB* = 0. *valueA* and *valueB* can be real numbers, expressions, or lists.

2nde [tests]
LOGIC
3:xor

xyLine

xyLine Plot#(*type*,*Xlist*,[,*freqlist*,*color#*])

† 2nde

xyLine	
Used as the "type" argument in the command Where # gives Plot1, Plot2 or Plot3.	[graph stats] **TYPE**

Z

ZBox	
ZBox Displays a graph, lets you draw a box that defines a new viewing window, and updates the window.	† [zoom] **ZOOM** **1:ZBox**

ZDecimal	
ZDecimal Adjusts the viewing window so that **TraceStep=0.1**, Δ**X=0.5** and Δ**Y=0.5**, and displays the graph screen with the origin centered on the screen.	† [zoom] **ZOOM** **4:ZDecimal**

ZFrac1/2	
ZFrac1/2 Sets the window variables so that you can trace in increments of $\frac{1}{2}$, if possible. Sets **TraceStep** to $\frac{1}{2}$ and Δ**X** and Δ**Y** to $\frac{1}{4}$.	[ZOOM] **ZOOM** **B:ZFrac1/2**

ZFrac1/3	
ZFrac1/3 Sets the window variables so that you can trace in increments of $\frac{1}{3}$, if possible. Sets **TraceStep** to $\frac{1}{3}$ and Δ**X** and Δ**Y** to $\frac{1}{6}$.	[zoom] **ZOOM** **C:ZFrac1/3**

ZFrac1/4	
ZFrac1/4 Sets the window variables so that you can trace in increments of $\frac{1}{4}$, if possible. Sets **TraceStep** to $\frac{1}{4}$ and Δ**X** and Δ**Y** to $\frac{1}{8}$.	[zoom] **ZOOM** **D:ZFrac1/4**

ZFrac1/5	
ZFrac1/5	[zoom] **ZOOM**

ZFrac1/5

Sets the window variables so that you can trace in increments of $\frac{1}{5}$, if possible. Sets **TraceStep** to $\frac{1}{5}$ and Δ**X** and Δ**Y** to $\frac{1}{10}$.

E:ZFrac1/5

ZFrac1/8

ZFrac1/8

Sets the window variables so that you can trace in increments of $\frac{1}{8}$, if possible. Sets **TraceStep** to $\frac{1}{8}$ and Δ**X** and Δ**Y** to $\frac{1}{16}$.

zoom
ZOOM
F:ZFrac1/8

ZFrac1/10

ZFrac1/10

Sets the window variables so that you can trace in increments of $\frac{1}{10}$, if possible. Sets **TraceStep** to $\frac{1}{10}$ and Δ**X** and Δ**Y** to $\frac{1}{20}$.

zoom
ZOOM
G:ZFrac1/10

ZInteger

ZInteger

Redefines the viewing window using the following dimensions:
TraceStep=1, ΔX=0.5, Xscl=10, ΔY=1, Yscl=10.

† zoom
ZOOM
8:ZInteger

ZInterval

ZIntervalσ[,*listname*,*freqlist*,*confidence level*]
(Data list input)

Computes a *z* confidence interval.

† stats
TESTS
7:ZInterval

ZInterval

ZIntervalσ,$\bar{x}$,*n*[,*confidence level***]**
(Summary stats input)

Computes a *z* confidence interval.

† stats
TESTS
7:ZInterval

Zoom In

Zoom In

Magnifies the part of the graph that surrounds the cursor location.

† zoom
ZOOM
2:Zoom In

Zoom Out

Zoom Out

Displays a greater portion of the graph, centered on the cursor location.

† zoom
ZOOM
3:Zoom Out

ZoomFit

ZoomFit

Recalculates **Ymin** and **Ymax** to include the minimum and maximum **Y** values, between **Xmin** and **Xmax**, of the selected functions and replots the functions.

† zoom
ZOOM
0:ZoomFit

ZoomRcl

ZoomRcl

Graphs the selected functions in a user-defined viewing window.

† zoom
MEMORY
3:ZoomRcl

ZoomStat

ZoomStat

Redefines the viewing window so that all statistical data points are displayed.

† zoom
ZOOM
9:ZoomStat

ZoomSto

ZoomSto

Immediately stores the current viewing window.

† zoom
MEMORY
2:ZoomSto

ZPrevious

ZPrevious

Replots the graph using the window variables of the graph that was displayed before you executed the last **ZOOM** instruction.

† zoom
MEMORY
1:ZPrevious

ZQuadrant1

ZQuadrant1

Displays the portion of the graph that is in quadrant 1.

zoom
ZOOM
A:ZQuadrant1

ZSquare

ZSquare

Adjusts the **X** or **Y** window settings so that each pixel represents an equal width and height in the coordinate system, and updates the viewing window.

† zoom
ZOOM
5:ZSquare

ZStandard

ZStandard

Replots the functions immediately, updating the window variables to the default values.

† zoom
ZOOM
6:ZStandard

Z-Test(

Z-Test($\mu 0$,σ **[**,*listname*,*freqlist*,*alternative*,*drawflag*,*color#*]**)**
(Data list input)

Performs a *z* test with frequency *freqlist*. *alternative*=**−1** is **<**; *alternative*=**0** is ≠; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats
TESTS
1:Z-Test(

Z-Test(

Z-Test($\mu 0$,σ,$\bar{x}$,*n***[**,*alternative*,*drawflag*,*color#*]**)**
(Summary stats input)

Performs a *z* test. *alternative*=**−1** is **<**; *alternative*=**0** is ≠; *alternative*=**1** is **>**. *drawflag*=**1** draws results; *drawflag*=**0** calculates results.

Color#: 10 - 24 or color name pasted from [vars] COLOR.

† stats
TESTS
1:Z-Test(

ZTrig

ZTrig

Replots the functions immediately, updating the window variables to preset values for plotting trig functions.

† zoom
ZOOM
7:ZTrig

Arithmetic Operations, Test Relations, and Symbols

! (factorial)

Factorial: *value*!

Returns factorial of *value*.

math
PRB
4:!

! (factorial)

Factorial: *list*!

Returns factorial of *list* elements.

math
PRB
4:!

° (degrees notation)

Degrees notation: *value*°

Interprets *value* as degrees; designates degrees in DMS format.

2nde [angle]
ANGLE
1: °

ʳ (radian)

Radian: *angle*ʳ

Interprets *angle* as radians.

2nde [angle]
ANGLE
3:ʳ

T (transpose)

Transpose: *matrix*T

Returns a matrix in which each element (row, column) is swapped with the corresponding element (column, row) of *matrix*.

matrice
MATH
2: T

x√

$x^{th}root\sqrt[x]{}value$

Returns $x^{th}root$ of *value*.

math
MATH
5: $\sqrt[x]{}$

x√(

$x^{th}root\sqrt[x]{}list$

math

$\sqrt[x]{}$(

Returns x^{th} *root* of *list* elements.

MATH
5: $\sqrt[x]{}$

$\sqrt[x]{}$(

list $\sqrt[x]{}$ *value*

Returns *list* roots of *value*.

math
MATH
5: $\sqrt[x]{}$

$\sqrt[x]{}$(

listA $\sqrt[x]{}$ *listB*

Returns *listA* roots of *listB*.

math
MATH
5: $\sqrt[x]{}$

3 (cube)

Cube: *value*^{3}

Returns the cube of a real or complex number, expression, list, or square matrix.

math
MATH
3: ^{3}

$\sqrt[3]{}$((cube root)

Cube root: $\sqrt[3]{}$(*value*)

Returns the cube root of a real or complex number, expression, or list.

math
MATH
4: $\sqrt[3]{}$ (

= (equal)

Equal:
valueA=*valueB*

Returns 1 if *valueA* = *valueB*. Returns 0 if *valueA* *valueB*. *valueA* and *valueB* can be real or complex numbers, expressions, lists, or matrices.

2nde [tests]
TEST
1:=

≠ (not equal)

Not equal:
valueA≠*valueB*

Returns 1 if *valueA* ≠ *valueB*. Returns 0 if *valueA* = *valueB*. *valueA* and *valueB* can be real or complex numbers, expressions, lists, or matrices.

2nde [tests]
TEST
2: ≠

< (less than)

Less than:
valueA<*valueB*

Returns 1 if *valueA* < *valueB*. Returns 0 if *valueA* ≥ *valueB*. *valueA* and *valueB* can be real or complex numbers, expressions, or lists.

2nde [tests]
TEST
5:<

> (greater than)

Greater than:
valueA>*valueB*

Returns 1 if *valueA* > *valueB*. Returns 0 if *valueA* ≤ *valueB*. *valueA* and *valueB* can be real or complex numbers, expressions, or lists.

2nde [tests]
TEST
3:>

≤ (less or equal)

Less than or equal:
valueA≤*valueB*

Returns 1 if *valueA* ≤ *valueB*. Returns 0 if *valueA* > *valueB*. *valueA* and *valueB* can be real or complex numbers, expressions, or lists.

2nde [tests]
TEST
6: ≤

≥ (greater or equal)

Greater than or equal:
valueA≥*valueB*

Returns 1 if *valueA* ≥ *valueB*. Returns 0 if *valueA* < *valueB*. *valueA* and *valueB* can be real or complex numbers, expressions, or lists.

2nde [tests]
TEST
4: ≥

$^{-1}$ (inverse)

Inverse: *value*$^{-1}$

Returns 1 divided by a real or complex number or expression.

[x^{-1}]

$^{-1}$ (inverse)

Inverse: $list^{-1}$ 

Returns 1 divided by *list* elements.

$^{-1}$ (inverse)

Inverse: $matrix^{-1}$ 

Returns *matrix* inverted.

^{2} (square)

Square: $value^{2}$ 

Returns *value* multiplied by itself. *value* can be a real or complex number or expression.

^{2} (square)

Square: $list^{2}$ 

Returns *list* elements squared.

^{2} (square)

Square: $matrix^{2}$ 

Returns *matrix* multiplied by itself.

^ (power)

Powers: *value*^*power* 

Returns *value* raised to *power*. *value* can be a real or complex number or expression.

^ (power)

Powers: *list*^*power* 

Returns *list* elements raised to *power*.

^ (power)

Powers: *value*^*list* [^]

Returns *value* raised to *list* elements.

^ (power)

Powers: *matrix*^*power* [^]

Returns *matrix* elements raised to *power*.

⁻ (negation)

Negation: ⁻*value* [(-)]

Returns the negative of a real or complex number, expression, list, or matrix.

10^((power of ten)

Power of ten: 10^(*value*) [2nde] [10^x]

Returns 10 raised to the *value* power. *value* can be a real or complex number or expression.

10^((power of ten)

Power of ten: 10^(*list*) [2nde] [10^x]

Returns a list of 10 raised to the *list*

√((square root)

Square root: √(*value*) [2nde] [√]

Returns square root of a real or complex number, expression, or list.

✱ (multiply)

Multiplication: [×]
valueA✱*valueB*

Returns *valueA* times *valueB*.

✱ (multiply)

Multiplication: [×]

✱ (multiply)

value✱*list*

Returns *value* times each *list* element.

✱ (multiply)

Multiplication: ⊠
list✱*value*

Returns each *list* element times *value*.

✱ (multiply)

Multiplication: ⊠
listA✱*listB*

Returns *listA* elements times *listB* elements.

✱ (multiply)

Multiplication: ⊠
value✱*matrix*

Returns value times *matrix* elements.

✱ (multiply)

Multiplication: ⊠
matrixA✱*matrixB*

Returns *matrixA* times *matrixB*.

/ (divide)

Division: *valueA*/*valueB* ÷

Returns *valueA* divided by *valueB*

/ (divide)

Division: *list*/*value* ÷

Returns *list* elements divided by value.

/ (divide)

Division: *value*/*list* ÷

/ (divide)

Returns value divided by *list* elements.

/ (divide)

Division: *listA*/*listB*

Returns *listA* elements divided by *listB* elements.

+ (add)

Addition: *valueA*+*valueB*

Returns *valueA* plus *valueB*.

+ (add)

Addition: *list*+*value*

Returns list in which *value* is added to each *list* element.

+ (add)

Addition: *listA*+*listB*

Returns *listA* elements plus *listB* elements.

+ (add)

Addition:
matrixA+*matrixB*

Returns *matrixA* elements plus *matrixB* elements.

+ (concatenation)

Concatenation:
string1+*string2*

Concatenates two or more strings.

− (subtract)

Subtraction:
valueA−*valueB*

− (subtract)

Subtracts *valueB* from *valueA*.

− (subtract)

Subtraction:
value−*list*

Subtracts *list* elements from *value*

− (subtract)

Subtraction:
list−*value*

Subtracts *value* from *list* elements.

− (subtract)

Subtraction:
listA−*listB*

Subtracts *listB* elements from *listA* elements.

− (subtract)

Subtraction:
matrixA−*matrixB*

Subtracts *matrixB* elements from *matrixA* elements.

' (minutes notation)

Minutes notation:*degrees*°*minutes*'
seconds"

2nde [angle]
ANGLE
2:'

Interprets *minutes* angle measurement as minutes.

" (seconds notation)

Seconds notation:
degrees°*minutes*'*seconds*"

alpha [''']

Interprets *seconds* angle measurement as seconds.

Error Messages

When the TI-82 Advance Edition Python detects an error, it returns an error message as a menu title, such as **ERR:SYNTAX** or **ERR:DOMAIN**. This table contains each error type, possible causes, and suggestions for correction. The error types listed in this table are each preceded by **ERR:** on your graphing calculator display. For example, you will see **ERR:ARCHIVED** as a menu title when your graphing calculator detects an **ARCHIVED** error type.

ERROR TYPE	Possible Causes and Suggested Remedies
ARCHIVED	You have attempted to use, edit, or delete an archived variable. For example, the expression dim(L1) produces an error if L1 is archived.
ARCHIVE FULL	You have attempted to archive a variable and there is not enough space in archive to receive it.
ARGUMENT	A function or instruction does not have the correct number of arguments. The arguments are shown in italics. The arguments in brackets are optional and you need not type them. You must also be sure to separate multiple arguments with a comma (,). For example, **stdDev(***list*[,*freqlist*]**)** might be entered as stdDev(L1) or stdDev(L1,L2) since the frequency list or *freqlist* is optional.
BAD ADDRESS	You have attempted to send or receive an application and an error (e.g. electrical interference) has occurred in the transmission.
BAD GUESS	In a **CALC** operation, you specified a **Guess** that is not between **Left Bound** and **Right Bound**. For the **solve(** function or the equation solver, you specified a *guess* that is not between *lower* and *upper*. Your guess and several points around it are undefined. Examine a graph of the function. If the equation has a solution, change the bounds and/or the initial guess.
BOUND	In a **CALC** operation or with **Select(**, you defined **Left Bound > Right Bound**. In **fMin(**, **fMax(**, **solve(**, or the equation solver, you entered *lower upper*.
BREAK	You pressed the [on] key to break execution of a program, to halt a **DRAW** instruction, or to stop evaluation of an expression.
DATA TYPE	You entered a value or variable that is the wrong data type. For a function (including implied multiplication) or an instruction, you entered an argument that is an invalid data type, such as a complex number where a real number is

ERROR TYPE	Possible Causes and Suggested Remedies
	required. In an editor, you entered a type that is not allowed, such as a matrix entered as an element in the stat list editor. You attempted to store an incorrect data type, such as a matrix, to a list. You attempted to enter complex numbers into the n/d MathPrint™ template.
DIMENSION MISMATCH	Your calculator displays the **ERR:DIMENSION MISMATCH** error if you are trying to perform an operation that references one or more lists or matrices whose dimensions do not match. For example, multiplying L1*L2, where L1={1,2,3,4,5} and L2={1,2} produces an **ERR:DIMENSION MISMATCH** error because the number of elements in L1 and L2 do not match. You may need to turn Plots Off to continue.
DIVIDE BY 0	You attempted to divide by zero. This error is not returned during graphing. The TI-82 Advance Edition Python allows for undefined values on a graph. • You attempted a linear regression with a vertical line.
DOMAIN	You specified an argument to a function or instruction outside the valid range. The TI-82 Advance Edition Python allows for undefined values on a graph. You attempted a logarithmic or power regression with a $^{-}$**X** or an exponential or power regression with a $^{-}$**Y**. You attempted to compute Σ**Prn(** or Σ**Int(** with $pmt2 < pmt1$.
DUPLICATE	You attempted to create a duplicate group name.
Duplicate Name	A variable you attempted to transmit cannot be transmitted because a variable with that name already exists in the receiving unit.
EXPIRED	You have attempted to run an application with a limited trial period which has expired.
Error in Xmit	The TI-82 Advance Edition Python was unable to transmit an item. Check to see that the cable is firmly connected to both units and that the receiving unit is in receive mode. You pressed [on] to break during transmission. Setup RECEIVE first and then SEND, when sending files ([échanger]) between graphing calculators.
ID NOT FOUND	This error occurs when the SendID command is executed but the proper graphing calculator ID cannot be found.
ILLEGAL	You attempted to use an invalid function in an argument to

ERROR TYPE	Possible Causes and Suggested Remedies
NEST	a function, such as **seq(** within *expression* for **seq(**.
INCREMENT	The increment, step, in **seq(** is 0 or has the wrong sign. . The TI-82 Advance Edition Python allows for undefined values on a graph. The increment in a **For(** loop is 0.
INVALID	You attempted to reference a variable or use a function where it is not valid. For example, **Y***n* cannot reference **Y**, **Xmin**, **ΔX**, or **TblStart**. In **Seq** mode, you attempted to graph a phase plot without defining both equations of the phase plot. In **Seq** mode, you attempted to graph a recursive sequence without having input the correct number of initial conditions. In **Seq** mode, you attempted to reference terms other than **(***n***−1)** or **(***n***−2)**. You attempted to designate a graph style that is invalid within the current graph mode. You attempted to use **Select(** without having selected (turned on) at least one xyLine or scatter plot.
INVALID DIMENSION	The **ERR:INVALID DIMENSION** error message may occur if you are trying to graph a function that does not involve the stat plot features. The error can be corrected by turning off the stat plots. To turn the stat plots off, press 2nde [graph stats] and then select **4:PlotsOff**. You specified a list dimension as something other than an integer between 1 and 999. You specified a matrix dimension as something other than an integer between 1 and 99. You attempted to invert a matrix that is not square.
ITERATIONS	The **solve(** function or the equation solver has exceeded the maximum number of permitted iterations. Examine a graph of the function. If the equation has a solution, change the bounds, or the initial guess, or both. **irr(** has exceeded the maximum number of permitted iterations. When computing **I**%, the maximum number of iterations was exceeded.
LABEL	The label in the **Goto** instruction is not defined with a **Lbl** instruction in the program.
LINK L1 (or any other file) to	The calculator has been disabled for testing. To restore full functionality, use TI Connect™ CE software to download a

ERROR TYPE	Possible Causes and Suggested Remedies
Restore	file to your calculator from your computer, or transfer any file to your calculator from another TI-82 Advance Edition Python.
MEMORY	Memory is insufficient to perform the instruction or function. You must delete items from memory before executing the instruction or function. Recursive problems return this error; for example, graphing the equation **Y1=Y1**. Branching out of an **If**/**Then**, **For(**, **While**, or **Repeat** loop with a **Goto** also can return this error because the **End** statement that terminates the loop is never reached.
MemoryFull	You are unable to transmit an item because the receiving unit's available memory is insufficient. You may skip the item or exit receive mode. During a memory backup, the receiving unit's available memory is insufficient to receive all items in the sending unit's memory. A message indicates the number of bytes the sending unit must delete to do the memory backup. Delete items and try again.
MODE	You attempted to store to a window variable in another graphing mode or to perform an instruction while in the wrong mode; for example, **DrawInv** in a graphing mode other than **Func**.
NO SIGN CHANGE	The **solve(** function or the equation solver did not detect a sign change. You attempted to compute **I**% when **FV**, (**N** **PMT**), and **PV** are all 0, or when **FV**, (**N** **PMT**), and **PV** are all 0. You attempted to compute **irr(** when neither $CFList$ nor CFO is > 0, or when neither $CFList$ nor CFO is < 0.
NONREAL ANSWERS	In **Real** mode, the result of a calculation yielded a complex result. . The TI-82 Advance Edition Python allows for undefined values on a graph.
OVERFLOW	You attempted to enter, or you have calculated, a number that is beyond the range of the graphing calculator. The TI-82 Advance Edition Python allows for undefined values on a graph.
RESERVED	You attempted to use a system variable inappropriately.
SINGULAR MATRIX	A singular matrix (determinant = 0) is not valid as the argument for $^{-}$**1**. The **SinReg** instruction or a polynomial regression generated a singular matrix (determinant = 0) because the algorithm could not find a solution, or a solution does not

ERROR TYPE	Possible Causes and Suggested Remedies
	exist. The TI-82 Advance Edition Python allows for undefined values on a graph.
SINGULARITY	*expression* in the **solve(** function or the equation solver contains a singularity (a point at which the function is not defined). Examine a graph of the function. If the equation has a solution, change the bounds or the initial guess or both.
STAT	You attempted a stat calculation with lists that are not appropriate. Statistical analyses must have at least two data points. **Med-Med** must have at least three points in each partition. When you use a frequency list, its elements must be 0. (**Xmax** − **Xmin**) ⁄ **Xscl** must be between 0 and 131 for a histogram.
STAT PLOT	You attempted to display a graph when a stat plot that uses an undefined list is turned on.
SYNTAX	The command contains a syntax error. Look for misplaced functions, arguments, parentheses, or commas. For example, **stdDev(***list*[,*freqlist*]**)** is a function of the TI-82 Advance Edition Python. The arguments are shown in italics. The arguments in brackets are optional and you need not type them. You must also be sure to separate multiple arguments with a comma (,). For example **stdDev (***list*[,*freqlist*]**)** might be entered as stdDev(L1) or stdDev (L1,L2) since the frequency list or *freqlist* is optional.
TOLERANCE NOT MET	You requested a tolerance to which the algorithm cannot return an accurate result.
UNDEFINED	You referenced a variable that is not currently defined. For example, you referenced a stat variable when there is no current calculation because a list has been edited, or you referenced a variable when the variable is not valid for the current calculation, such as **a** after **Med-Med**.
VALIDATION	Electrical interference caused a link to fail or this graphing calculator is not authorized to run the application.
VARIABLE	You have tried to archive a variable that cannot be archived or you have tried to unarchive an application or group. Examples of variables that cannot be archived include: Real numbers **LRESID, R, T, X, Y**, **Theta**, Statistic variables under **Vars**, **STATISTICS** menu, **Yvars**, and the **AppIdList**.
VERSION	You have attempted to receive an incompatible variable

ERROR TYPE	Possible Causes and Suggested Remedies
	version from another graphing calculator. A program may contain commands not supported in the OS version on your graphing calculator. Always use the latest OS.
WINDOW RANGE	A problem exists with the window variables. You defined **Xmax Xmin** or **Ymax Ymin**. You defined θ**max** θ**min** and θ**step** > **0** (or vice versa). You attempted to define **Tstep=0**. You defined **Tmax Tmin** and **Tstep** > **0** (or vice versa). Window variables are too small or too large to graph correctly. You may have attempted to zoom to a point that exceeds the TI-82 Advance Edition Python's numerical range.
ZOOM	A point or a line, instead of a box, is defined in **ZBox.** A **ZOOM** operation returned a math error.

General Information

Online Help

education.ti.com/eguide

Select your country for more product information.

Contact TI Support

education.ti.com/ti-cares

Select your country for technical and other support resources.

Service and Warranty Information

education.ti.com/warranty

Select your country for information about the length and terms of the warranty or about product service.

Limited Warranty. This warranty does not affect your statutory rights.