Python Programming for the TI-84 Plus CE-T *Python Edition* Graphing Calculator

Version 5.7.0. Bundle 84CE-T

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

Important Information

Note: Actual screens may vary slightly from provided images.

Contents

What's New

What's New in Python App

TI-84 Plus CE-T *Python Edition*

Python Programming

- Access the Python App from [prgm] when the Python App is loaded. Python App is also listed in [2nd] [apps].
  - Stay up to date at education.ti.com/84cetupdate.
  - Find details for the Python App in the Python Programming guide at education.ti.com/eguide.
- Quick paste of import statements for Add-On modules. Add-On modules are available in Python activities posted on education.ti.com.
- New ti_draw and ti_image Add-On modules load with the CE Bundle.
  - Draw and use images in your Python programs.
- ti_system module menu now contains the wait_key() method for ease of use.
- ti_hub and ti_rover modules contain the latest TI-Innovator™ Hub sketch v 1.5 support.
  - Data Collection - collect multiple data samples in a single command
  - Compound Statements to synchronize multiple outputs
  - TI-RGB Array - control multiple LEDs
  - Sound - use single command to play repeated beeps
  - Ranger - return "time of flight"

Transferring Python Programs

When transferring Python programs from a non-TI platform to a TI platform OR from one TI product to another:

- Python programs that use core language features and standard libs (math, random etc.) can be ported without changes.
  Note: List length limit is 100 elements.
- Programs that use platform-specific libraries - matplotlib (for PC), ti_plotlib/ti_system/ti_hub/etc. for TI platforms, will require edits before they will run on a different platform.

This may be true even between TI platforms.

For more information about the new and updated functionality, go to education.ti.com/84cetupdate.

Python App

See the following for using, navigating, and running the Python App.

- Using Python App
- Python App Navigation
- Example Activity

Using Python App

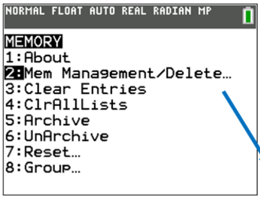
The Python App is available for the TI-84 Plus CE *Python*. The information in this eGuide is for use with the TI-84 Plus CE *Python* updated with the latest CE Bundle.

When you first run the Python App on your TI-84 Plus CE *Python*, the App may direct you to update to the latest CE Bundle for the latest Python App.
Please see at education.ti.com/84cetupdate to update your
TI-84 Plus CE *Python*.

Python program (PY AppVar) memory management

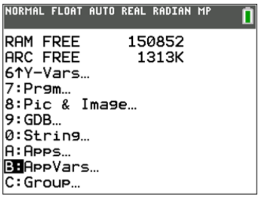
The Python App offers a File Manager, an Editor to create programs, and a Shell to run programs and interact with the Python interpreter. Python programs stored or created as Python AppVars will execute from RAM. You may store Python AppVars in Archive for memory management [2nd] [mem] 2:. If the Python App File Manger screen does not display one of your **PY AppVar** programs, you can move a **PY AppVar** calculator Python program between RAM or Archive memory as shown. The * denotes a file in Archive. Press [enter] to move file between RAM and Archive.

[2nd][mem] 2



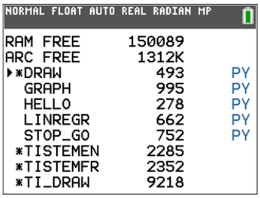
```
NORMAL FLOAT AUTO REAL RADIAN MP
MEMORY
1:About
2:Mem Management/Delete…
3:Clear Entries
4:ClrAllLists
5:Archive
6:UnArchive
7:Reset…
8:Group…
```

> B:AppVars …



```
NORMAL FLOAT AUTO REAL RADIAN MP
RAM FREE      150852
ARC FREE       1313K
6↑Y-Vars…
7:Prgm…
8:Pic & Image…
9:GDB…
0:String…
A:Apps…
B:AppVars…
C:Group…
```

[enter] RAM <> *ARC



```
NORMAL FLOAT AUTO REAL RADIAN MP
RAM FREE      150089
ARC FREE       1312K
▸*DRAW           493      PY
  GRAPH          995      PY
  HELLO          278      PY
  LINREGR        662      PY
  STOP_GO        752      PY
 *TISTEMEN      2285
 *TISTEMFR      2352
 *TI_DRAW       9218
```

Note: If your calculator is the TI-84 Plus CE *Python*, please see education.ti.com/84cetupdate to find the latest information for your CE.

Python App Navigation

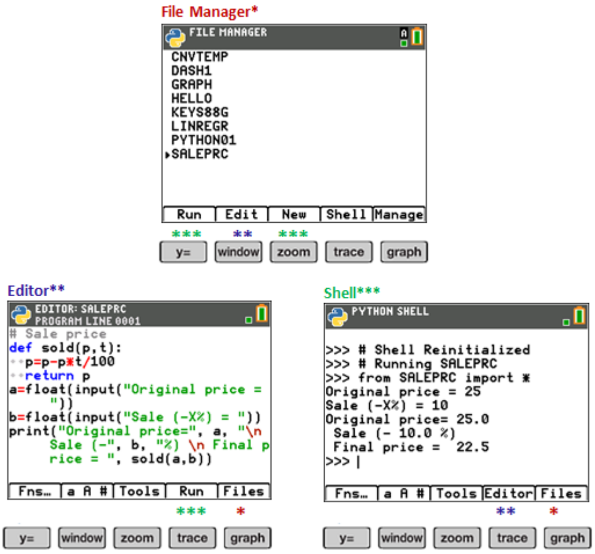
Use the shortcut keys on the screen in the App to navigate between workspaces in the Python App. In the image, the shortcut tab labels indicate:

* Navigation to the File Manager [Files]

** Navigation the Editor [Edit] or [Editor]

*** Navigation to the Shell [Shell]

Access shortcut tabs on the screen using the graphing key row immediately under the screen. Also, see Keypad. The Editor>Tools menu and Shell>Tools menu also contain navigation actions.


File Manager*
FILE MANAGER
Run | Edit | New | Shell | Manage
*** ** ***
y= window zoom trace graph
Editor**
EDITOR: SALEPRC
PROGRAM LINE 0001
Fns… | a A # | Tools | Run | Files
*** *
y= window zoom trace graph
Shell***
PYTHON SHELL
Fns… | a A # | Tools | Editor | Files
** *
y= window zoom trace graph

```
CNVTEMP
DASH1
GRAPH
HELLO
KEYS8G
LINREGR
PYTHON01
▸SALEPRC
```

```
# Sale price
def sold(p,t):
  p=p-p*t/100
  return p
a=float(input("Original price =
     "))
b=float(input("Sale (-X%) = "))
print("Original price=", a, "\n
     Sale (-", b, "%) \n Final p
     rice = ", sold(a,b))
```

```
>>> # Shell Reinitialized
>>> # Running SALEPRC
>>> from SALEPRC import *
Original price = 25
Sale (-X%) = 10
Original price= 25.0
 Sale (- 10.0 %)
 Final price =  22.5
>>> |
```

Example Activity

Use the example activity provided as an experience to become familiar with the workspaces in the Python App.

- Create a new program from the File Manager
- Write the program in the Editor
- Execute the program in the Shell in the Python App.

For more about Python programming on your CE, please see resources for TI-84 Plus CE-T *Python Edition*.

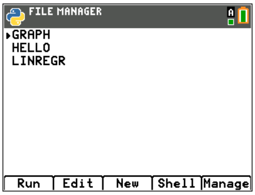
Getting Started:

- Run the Python App.

Note: Actual screens may vary slightly from provided images.

Enter new program name from File Manager.

- Press [zoom] ([New]) to create a new program.

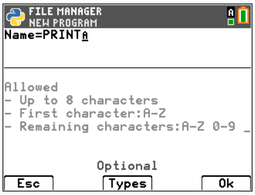


```
FILE MANAGER
▸GRAPH
 HELLO
 LINREGR
```

New File Name Entry

- The example program will be named "PRINT". Enter the program name and press [graph] ([Ok]).
- Notice the cursor is in ALPHA lock. Always enter a program name following the given requirements on the screen.

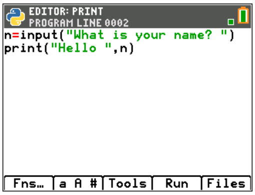
TIp: If the cursor is not in ALPHA lock, press [2nd] [alpha] [alpha] for upper case letters.



```
FILE MANAGER
NEW PROGRAM
Name=PRINTA

Allowed
- Up to 8 characters
- First character:A-Z
- Remaining characters:A-Z 0-9 _

Optional
```

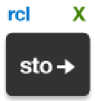
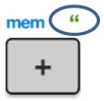
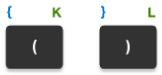
Enter program as shown.

TIp: The App provides a quick entry! Always watch the cursor state as you enter your program!



```
EDITOR: PRINT
PROGRAM LINE 0002
n=input("What is your name? ")
print("Hello ",n)
```

Alphabet characters on Keypad	[alpha] toggles the insert cursor state in the Editor and Shell. _ non-alpha a lower case alpha A upper case ALPHA
Where is the equal sign?	Press [sto→] when the

	cursor is _. 
Where are these located? input() print()	[Fns…] I/O 1:print() 2:input()
Where is double quote?	alpha ["] 
Where are (and)?	Use keypad when cursor is _. 

Try-It! [a A #] and 2nd [catalog] also are helpers for quick entry as needed!

Execute the program PRINT

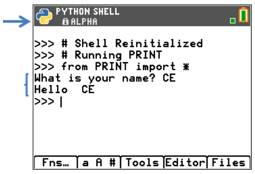
- From the Editor, press trace ([Run]) to execute your program in the Shell.
- Enter your name at the "What is your name?" prompt.
- Output displays "HELLO" with your name.

Note: At the Shell prompt >>>, you can execute a command, such as 2+3. If you use any method from math, random, or other available modules, be sure to execute an import module statement first as in any Python coding environment.

Shell cursor state indicator.

Input your name.

Output of PRINT displays.



```
>>> # Shell Reinitialized
>>> # Running PRINT
>>> from PRINT import *
What is your name? CE
Hello  CE
>>> |
```

Setting up a Python Session with your Programs

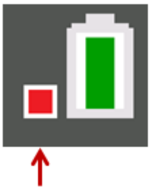
When the Python App is launched, the CE connection with the TI-Python experience will synchronize for your current Python session. You will see your list of programs in RAM and dynamic modules, as they synchronize to the Python experience.

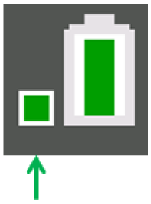
When the Python session is established, the status bar contains a green square indicator near the battery icon that signals the Python session is ready for use. In the event the indicator is red , wait for the indicator to change back to green when the Python experience is again available.

You may see an update of the Python distribution when launching the Python App along with program synchronization after the latest update for your TI-84 Plus CE-T *Python Edition* from education.ti.com/84cetupdate.

Disconnecting and Reconnecting the Python App

When the Python App is running, the status bar contains an indicator that signals whether Python is ready for use. Until the connection is established, the CE keypad may not respond. Best practice is to be aware of the status bar connection indicator while in your Python session.


Python Not Ready


Python Ready

Screen Captures

Using TI Connect™ CE at education.ti.com/84cetupdate, screen captures of any Python App screen is allowed.

Python Workspaces

The Python App contains three workspaces for your Python programming development.

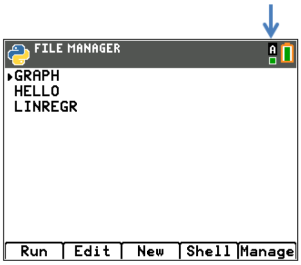
- File Manager
- Editor
- Shell

Python File Manager

The File Manager lists the Python AppVars available in RAM on your calculator. You can create, edit, and run programs as well as navigate to the Shell.

When in alpha state, press any letter on the keypad to jump to programs beginning with that letter.

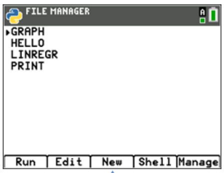
Press [alpha] if needed when **A** indicator is not in the status bar.

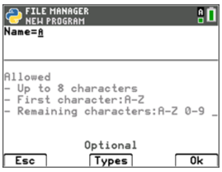


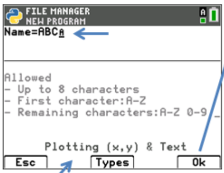
```
▸GRAPH
 HELLO
 LINREGR
```

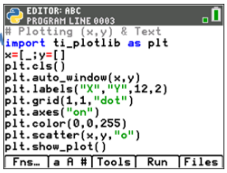
File Manager shortcut keys and menus		
Menus	**Keypress**	**Description**
[Run]	[y=]	Select a program using [▲]or [▼]. Next, select [Run] to execute your program.
[Edit]	[window]	Select a program using [▲]or [▼]. Next, select [Edit] to display the program in the Editor to edit your program.
[New]	[zoom]	Select [New] to enter a new program name and continue to the Editor to enter your new program. On the [New] screen, select [Types] (press [zoom]), to select a Type of program. By selecting a type of program, a template of import statements and frequently used functions and methods will be pasted to your new program for that activity.
[Shell]	[trace]	Select [Shell] to display the Shell prompt (Python interpreter). The Shell will be in the current state.
[Manage]	[graph]	Select [Manage] to: • View version number. • Replicate, delete or rename a selected program. • View the About screen. • Quit the App. Also use [2nd] [quit]

Create a New Program Using Program Type Templates

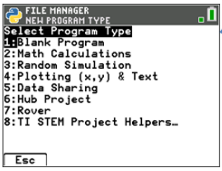

```
FILE MANAGER
▸GRAPH
 HELLO
 LINREGR
 PRINT
 Run | Edit | New | Shell |Manage
```


```
FILE MANAGER
NEW PROGRAM
Name=A

Allowed
- Up to 8 characters
- First character:A-Z
- Remaining characters:A-Z 0-9 _

            Optional
 Esc        Types           Ok
```


```
FILE MANAGER
NEW PROGRAM
Name=ABCA

Allowed
- Up to 8 characters
- First character:A-Z
- Remaining characters:A-Z 0-9 _

      Plotting (x,y) & Text
 Esc        Types           Ok
```


```
EDITOR: ABC
PROGRAM LINE 0003
# Plotting (x,y) & Text
import ti_plotlib as plt
x=[_;y=[]
plt.cls()
plt.auto_window(x,y)
plt.labels("X","Y",12,2)
plt.grid(1,1,"dot")
plt.axes("on")
plt.color(0,0,255)
plt.scatter(x,y,"o")
plt.show_plot()
 Fns… | a A # |Tools| Run |Files
```

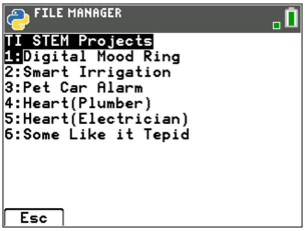
Selection pastes appropriate import statement(s) and progran lines to the new program.


```
FILE MANAGER
NEW PROGRAM TYPE
Select Program Type
1:Blank Program
2:Math Calculations
3:Random Simulation
4:Plotting (x,y) & Text
5:Data Sharing
6:Hub Project
7:Rover
8:TI STEM Project Helpers…

 Esc
```

- Select [Types] to display Select Program Type menu.
- Import(s) displays on status bar.

Create a New STEM Activity Program Using Templates

When the TISTEMEN AppVar is loaded in Archive, the “TI STEM Project Helpers...” menu item will display in the Select Program Type menu. Select the STEM activity template as needed to help begin a new STEM program.


```
FILE MANAGER
TI STEM Projects
1:Digital Mood Ring
2:Smart Irrigation
3:Pet Car Alarm
4:Heart(Plumber)
5:Heart(Electrician)
6:Some Like it Tepid

 Esc
```

Python Editor

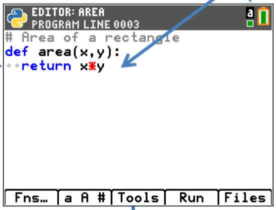
The Python Editor is displayed from a selected program in File Manager or from the Shell. The Editor displays keywords, operators, comments, strings and indents in color. Quick paste of common Python keywords and functions are available as well as direct keypad entry and [a A #] character entry . When pasting a code block such as if.. elif.. else, the Editor offers auto-indent which can be modified as needed as you write your program.

Cursor is always in insert mode. Use 2nd and alpha to toggle cursor. Cursor states are numeric, a, and A. [del] behaves as a backspace and delete of a character.

Program line location of the cursor.

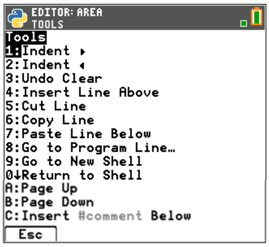
Auto indent code blocks.

Gray dots as visual indicator of indented lines.



```
EDITOR: AREA
PROGRAM LINE 0003
# Area of a rectangle
def area(x,y):
··return x*y
```

Useful tools for editing and working in the Shell. Full description below.



```
EDITOR: AREA
TOOLS
Tools
1:Indent ▸
2:Indent ◂
3:Undo Clear
4:Insert Line Above
5:Cut Line
6:Copy Line
7:Paste Line Below
8:Go to Program Line…
9:Go to New Shell
0↓Return to Shell
A:Page Up
B:Page Down
C:Insert #comment Below
```

Python Editor shortcut keys and menus

Menus	Keypress	Description
[Fns…]	y=	Select [Fns…] to access menus of commonly used functions, keywords, and operators. Also access selected contents of the math and random modules. **Note:** 2nd [catalog] is also helpful for quick paste.
[a A #]	window	Select [a A #] to access a character palette as an alternate way to enter many characters.

Python Editor shortcut keys and menus			
Menus	**Keypress**	**Description**	
[Tools]	zoom	Select [Tools] to access features to assist in your editing or your interaction with the Shell.	
		1: Indent ▶	Indents the program line to the right cursor moves to first character of the line.
		2: Indent ◀	Reduces the indent of the program line to the left. Cursor moves to first character of the line.
		3: Undo Clear	Pastes the last cleared line to a new line below the program line containing the cursor. Cursor displays at the end of the pasted line.
		4: Insert Line Above	Inserts a line above the program line with the cursor. Line will indent and display indent dots when appropriate.
		5: Cut Line	Current program line with cursor is cut. Cursor displays on program line below the cut line.
		6: Copy Line	Copies current program line with cursor. A copied program line can be pasted to the Shell prompt. See Shell below.
		7: Paste Line Below	Pastes the last stored program line to the line below the cursor position.
		8: Go to Program Line…	Displays cursor at the beginning of the specified program line.
		9: Go to New Shell	Displays reinitialized Shell.
		0: Return to Shell	Displays Shell in current state.
		A: Page up	Displays 11 program lines above current cursor position as available.
		B: Page Down	Displays 11 program lines below current cursor position as available.
		C: Insert #comment Below	Inserts # on a new line below cursor position.

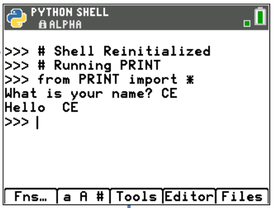
Python Editor shortcut keys and menus		
Menus	**Keypress**	**Description**
[Run]	trace	Select [Run] to execute your program.
[Files]	graph	Select [Files] to display the File Manager.

Python Shell

The Python Shell is the console where you can interact with the Python interpreter or run your Python programs. Quick paste of common Python keywords and functions is available as well as direct keypad entry and [a A #] character entry . The Shell prompt can be used to test one line of code pasted from the Editor. Multiple lines of code may also be entered and run at a Shell prompt >>>.

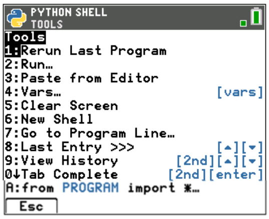
Shell cursor state indicator.

Shell reinitialize when a new program is executed.



```
PYTHON SHELL
ALPHA
>>> # Shell Reinitialized
>>> # Running PRINT
>>> from PRINT import *
What is your name? CE
Hello  CE
>>> |
```

Useful tools for working in the Shell. See details below.



```
PYTHON SHELL
TOOLS
Tools
1:Rerun Last Program
2:Run…
3:Paste from Editor
4:Vars…                      [vars]
5:Clear Screen
6:New Shell
7:Go to Program Line…
8:Last Entry >>>             [▲][▼]
9:View History          [2nd][▲][▼]
0↓Tab Complete          [2nd][enter]
A:from PROGRAM import *…
```

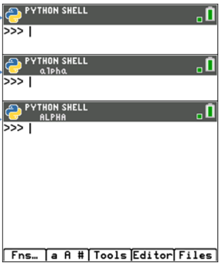
Shell Cursor States

non-alpha

[2nd] [alpha] if needed to toggle

[alpha] alpha

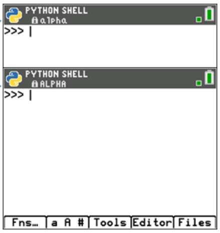
[alpha] again ALPHA



```
PYTHON SHELL
>>> |

PYTHON SHELL
alpha
>>> |

PYTHON SHELL
ALPHA
>>> |
```

[2nd] [alpha] lock alpha

[alpha] again lock ALPHA



```
PYTHON SHELL
alpha
>>> |

PYTHON SHELL
ALPHA
>>> |
```

Python Shell shortcut keys and menus		
Menus	**Keypress**	**Description**
[Fns…]	y=	Select [Fns…] to access menus of commonly used functions, keywords, and operators. Also access selected contents of the math and random modules. **Note:** 2nd [catalog] is also helpful for quick paste.
[a A #]	window	Select [a A #] to access a character palette as an alternate way to enter many characters.
[Tools]	zoom	Select [Tools] to display the following menu items.

1: Rerun last program	Reruns last program which was executed in the Shell.
2: Run…	Displays a list of the Python programs available to run in Shell.
3: Paste from Editor	Pastes the last copied program line from the Editor to the Shell prompt.
4: Vars…	Displays the vars from the last program which ran. Does not display program defined vars from an imported program.
5: Clear Screen	Clears the Shell screen. Does not reinitialize a new Shell.
6: New Shell	Reinitialize a new Shell.
7: Go to Program Line…	Displays the Editor from the Shell with cursor on the specified program line.
8: Last Entry>>> ▲ ▼	Displays up to the last 8 entries at the Shell prompt during a Shell session.
9: View History 2nd ▲ 2nd ▼	Scroll the Shell screen to view up to the last 60 lines of output in the Shell during a Shell session. After drawing to the Shell using ti_plotlib, ti_draw or ti_image, pressing [clear] will clear the drawing back to the Shell. The history will not be in display. Use [2nd] ▲ and [2nd] ▼ to view the history if needed.
0: Tab Complete 2nd [enter]	Displays the names of the variables and functions available for access in the current Shell session.

Python Shell shortcut keys and menus

Menus	Keypress	Description	
			When a letter of an available variable or function is entered, press [2nd] [enter] to auto-complete the name if a match is available in the current Shell session.
		A: from PROGRAM import *…	When first executed in a Shell session, PROGRAM will run and vars will only be viewable using Tab Complete. When executed again in the same Shell session, the execution will appear as no execution. This command can also be pasted from [2nd] [catalog].
[Editor]	[trace]	Select [Editor] to display the Editor with the last programs in Editor. If Editor is empty, you can display File Manager.	
[Files]	[graph]	Select [Files] to display the File Manager.	

Note:

- To break a running Python program, such as if a program is in a continuous loop, press [on]. Press **[Tools]** ([zoom]) > **6:New Shell** as an alternate method to halt a running program.
- When using ti_plotlib, ti_draw or ti_image modules to draw to the Shell, press [clear] to clear the draw and return to the Shell prompt at the top of the screen. To view the Shell history, use [2nd] [▲] and [2nd] [▼] to view the history as needed.

Execution Error: Go to Program Line using Shell >Tools

The TI-Python experience will display Python error messages in the Shell when code is executed. If an error is displayed when a program executes, a program line number will display. Use **Shell>Tools 7:Go to Program Line…** Enter the line number and press **[OK]**. The cursor will display on the first character of the appropriate program line in the Editor. The program line number is displayed in the second line of the Status bar in the Editor.

Fast Editing Support

When entering code in the Editor or in the Shell, use the following entry methods to quickly paste to the edit line.

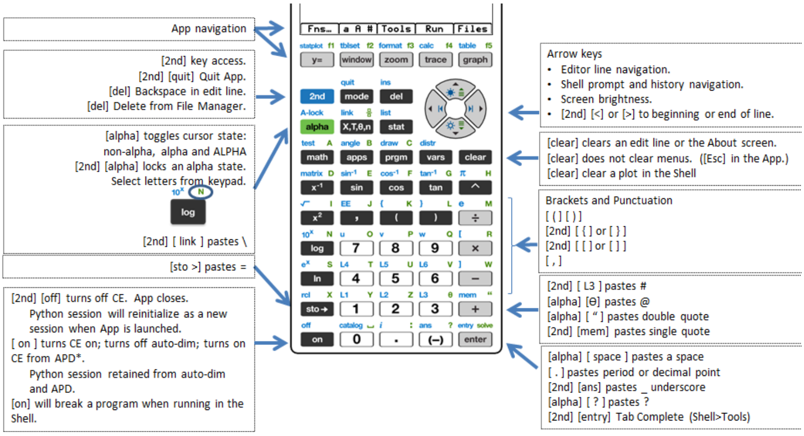
Tips for fast entry

- Using the Python Keypad
- Using the Python Catalog
- Using the [a A #] Character Map

Using the Python Keypad

When the Python App is running, the keypad is designed to paste the appropriate Python operations or open menus designed for easy entry of functions, keywords, methods, operators, etc. Pressing 2nd and alpha will access the second and third functions on a key as in the Operating System.

Python App Navigation, Editing, and Special Characters by Keypad Rows



App navigation

[2nd] key access.
[2nd] [quit] Quit App.
[del] Backspace in edit line.
[del] Delete from File Manager.

[alpha] toggles cursor state:
non-alpha, alpha and ALPHA
[2nd] [alpha] locks an alpha state.
Select letters from keypad.

[2nd] [link] pastes \

[sto >] pastes =

[2nd] [off] turns off CE. App closes.
Python session will reinitialize as a new session when App is launched.
[on] turns CE on; turns off auto-dim; turns on CE from APD*.
Python session retained from auto-dim and APD.
[on] will break a program when running in the Shell.

Arrow keys
- Editor line navigation.
- Shell prompt and history navigation.
- Screen brightness.
- [2nd] [<] or [>] to beginning or end of line.

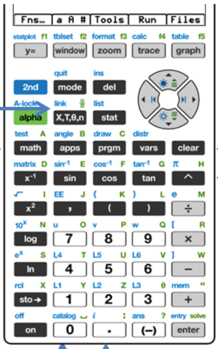
[clear] clears an edit line or the About screen.
[clear] does not clear menus. ([Esc] in the App.)
[clear] clear a plot in the Shell

Brackets and Punctuation
[( ] [ )]
[2nd] [{] or [}]
[2nd] [[ ] or [ ]]
[,]

[2nd] [L3] pastes #
[alpha] [θ] pastes @
[alpha] ["] pastes double quote
[2nd] [mem] pastes single quote

[alpha] [space] pastes a space
[.] pastes period or decimal point
[2nd] [ans] pastes _ underscore
[alpha] [?] pastes ?
[2nd] [entry] Tab Complete (Shell>Tools)

Python App Specific Key Presses for Menus and Functions by Keypad Rows



[X,T,θ,n] X or x
[2nd] [list] List Menu

[math] Modules Menu
[2nd] [test] Operators Menu

[x^-1] Paste **-1

[2nd] [rcl] ti_system menu
import ti_system module

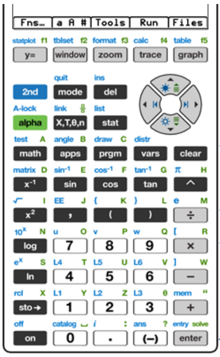
[2nd] [catalog]
Python specific catalog

[2nd] [i]
complex type imaginary j
a+bj

[var] displays available variables in Shell after a program is executed.

[2nd] [π]
[sin]; [cos]; [tan];
[2nd][sin];[2nd][cos];[2nd][tan]
displays Trig menu; import math module

Python App Specific Key Presses for Menus and Functions by Keypad Rows (Continued)



[x^2] pastes **2
[2nd] [√] pastes sqrt()
[2nd] [EE] pastes E

[log] pastes log(,10)
[2nd] [10^x] pastes 10**()

[ln] pastes log( ) base e
[2nd] [e^x] pastes exp()

[sto >] pastes =

[var] displays available variables in Shell after a program is executed.
[clear] Clears plotting area in Shell for ti_plotlib plotting methods.

[^] pastes **

[÷] pastes /
[2nd] [e] pastes e

[*] pastes *

[-] pastes -

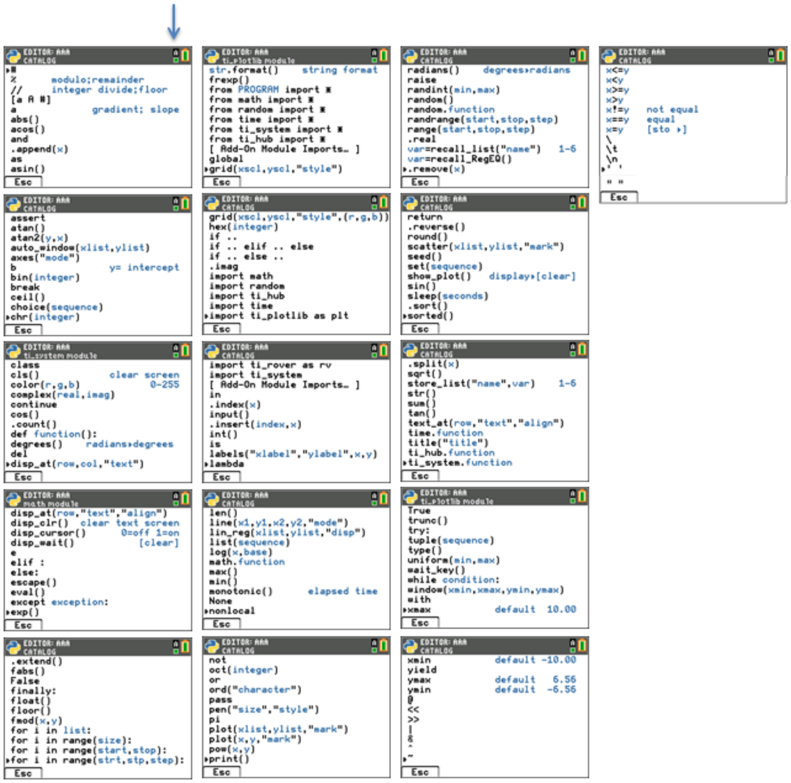
[+] pastes +

[enter]
- In File Manager, runs the selected program.
- In Editor, splits a program line.
- Use [2nd] [enter] to insert a line below.

Using the Python Catalog

When the Python App is running, [2nd] [catalog] will display a list of frequently used delimiters, keywords, functions and operators to quickly paste to an edit line. [2nd] [catalog] is available in Editor and Shell only. For a more detailed description of each Catalog item, please see the Reference Guide. From the top of the catalog menu, use [▲] for circular navigation of the catalog.

When in catalog, select [alpha] and a letter key to display the listing starting at that letter.



```
EDITOR: AAA
CATALOG
▸#
 %        modulo;remainder
 //       integer divide;floor
 [a A #]
 a                  gradient; slope
 abs()
 acos()
 and
 .append(x)
 as
 asin()
Esc
```

```
EDITOR: AAA
CATALOG
 assert
 atan()
 atan2(y,x)
 auto_window(xlist,ylist)
 axes("mode")
 b                 y= intercept
 bin(integer)
 break
 ceil()
 choice(sequence)
▸chr(integer)
Esc
```

```
EDITOR: AAA
ti_system module
 class
 cls()             clear screen
 color(r,g,b)             0-255
 complex(real,imag)
 continue
 cos()
 .count()
 def function():
 degrees()     radians▸degrees
 del
▸disp_at(row,col,"text")
Esc
```

```
EDITOR: AAA
math module
 disp_at(row,"text","align")
 disp_clr()  clear text screen
 disp_cursor()      0=off 1=on
 disp_wait()           [clear]
 e
 elif :
 else:
 escape()
 eval()
 except exception:
▸exp()
Esc
```

```
EDITOR: AAA
CATALOG
 .extend()
 fabs()
 False
 finally:
 float()
 floor()
 fmod(x,y)
 for i in list:
 for i in range(size):
 for i in range(start,stop):
▸for i in range(strt,stp,step):
Esc
```

```
EDITOR: AAA
ti_plotlib module
 str.format()    string format
 frexp()
 from PROGRAM import *
 from math import *
 from random import *
 from time import *
 from ti_system import *
 from ti_hub import *
 [ Add-On Module Imports… ]
 global
▸grid(xscl,yscl,"style")
Esc
```

```
EDITOR: AAA
CATALOG
 grid(xscl,yscl,"style",(r,g,b))
 hex(integer)
 if ..
 if .. elif .. else
 if .. else ..
 .imag
 import math
 import random
 import ti_hub
 import time
▸import ti_plotlib as plt
Esc
```

```
EDITOR: AAA
CATALOG
 import ti_rover as rv
 import ti_system
 [ Add-On Module Imports… ]
 in
 .index(x)
 input()
 .insert(index,x)
 int()
 is
 labels("xlabel","ylabel",x,y)
▸lambda
Esc
```

```
EDITOR: AAA
CATALOG
 len()
 line(x1,y1,x2,y2,"mode")
 lin_reg(xlist,ylist,"disp")
 list(sequence)
 log(x,base)
 math.function
 max()
 min()
 monotonic()      elapsed time
 None
▸nonlocal
Esc
```

```
EDITOR: AAA
CATALOG
 not
 oct(integer)
 or
 ord("character")
 pass
 pen("size","style")
 pi
 plot(xlist,ylist,"mark")
 plot(x,y,"mark")
 pow(x,y)
▸print()
Esc
```

```
EDITOR: AAA
CATALOG
 radians()    degrees▸radians
 raise
 randint(min,max)
 random()
 random.function
 randrange(start,stop,step)
 range(start,stop,step)
 .real
 var=recall_list("name")   1-6
 var=recall_RegEQ()
▸.remove(x)
Esc
```

```
EDITOR: AAA
CATALOG
 return
 .reverse()
 round()
 scatter(xlist,ylist,"mark")
 seed()
 set(sequence)
 show_plot()   display▸[clear]
 sin()
 sleep(seconds)
 .sort()
▸sorted()
Esc
```

```
EDITOR: AAA
CATALOG
 .split(x)
 sqrt()
 store_list("name",var)    1-6
 str()
 sum()
 tan()
 text_at(row,"text","align")
 time.function
 title("title")
 ti_hub.function
▸ti_system.function
Esc
```

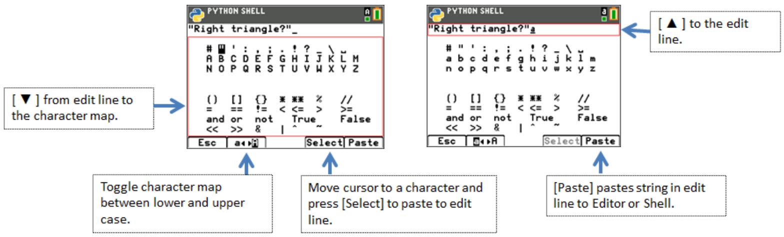
```
EDITOR: AAA
ti_plotlib module
 True
 trunc()
 try:
 tuple(sequence)
 type()
 uniform(min,max)
 wait_key()
 while condition:
 window(xmin,xmax,ymin,ymax)
 with
▸xmax             default  10.00
Esc
```

```
EDITOR: AAA
CATALOG
 xmin             default -10.00
 yield
 ymax             default   6.56
 ymin             default  -6.56
 @
 <<
 >>
 |
 &
▸~
Esc
```

```
EDITOR: AAA
CATALOG
 x<=y
 x<y
 x>=y
 x>y
 x!=y   not equal
 x==y   equal
 x=y    [sto ▸]
 \
 \t
 \n
▸' '
 " "
Esc
```

Using the [a A #] Character Map

[a A #] shortcut tab to a character palette is a convenient feature to enter strings when in Editor or Shell.



[▼] from edit line to the character map.

[▲] to the edit line.

Toggle character map between lower and upper case.

Move cursor to a character and press [Select] to paste to edit line.

[Paste] pastes string in edit line to Editor or Shell.

Note: When the cursor focus is in the [a A #] edit line, selected keypad keys are not available. When focus is in the character map, the keypad is restricted.

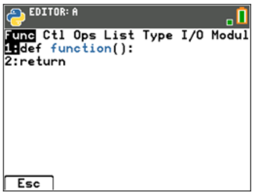
[Fns] Menus, Modules and Add-On Modules

- [Fns…] Menus
- [Fns…] Built-in, Operators and Keywords
- [Fns…] Modules
- [Fns…] Add-On Modules

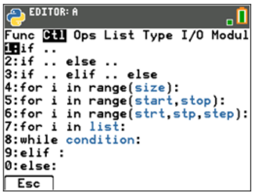
[Fns…] Menus

[Fns…] shortcut tab displays menus containing frequently used Python functions, keywords, and operators. The menus also provide access to the selected functions and constants from the Modules and Add-On Modules. While you can enter character by character from the keypad, these menus provide a quick way to paste in Editor or Shell. Press [Fns…] when in Editor or Shell. See also Using the Python Catalog and Using the Python Keypad for alternate entry methods.

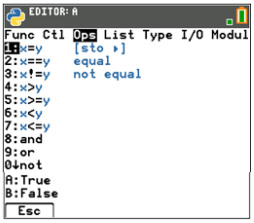
[Fns...] Built-in, Operators and Keywords



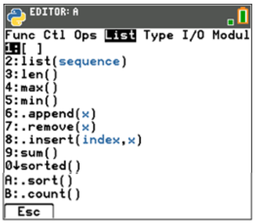
```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:def function():
2:return
Esc
```



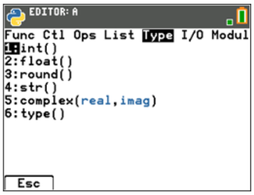
```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:if ..
2:if .. else ..
3:if .. elif .. else
4:for i in range(size):
5:for i in range(start,stop):
6:for i in range(strt,stp,step):
7:for i in list:
8:while condition:
9:elif :
0:else:
Esc
```



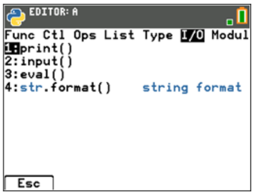
```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:x=y      [sto ▸]
2:x==y     equal
3:x!=y     not equal
4:x>y
5:x>=y
6:x<y
7:x<=y
8:and
9:or
0↓not
A:True
B:False
Esc
```



```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:[ ]
2:list(sequence)
3:len()
4:max()
5:min()
6:.append(x)
7:.remove(x)
8:.insert(index,x)
9:sum()
0↓sorted()
A:.sort()
B:.count()
Esc
```



```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:int()
2:float()
3:round()
4:str()
5:complex(real,imag)
6:type()
Esc
```



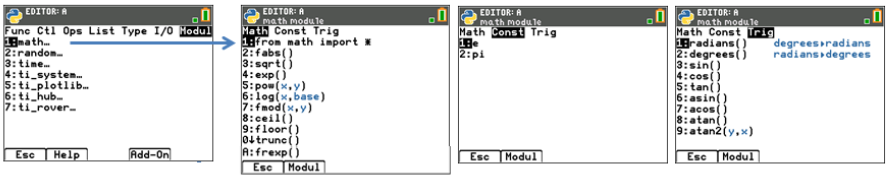
```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:print()
2:input()
3:eval()
4:str.format()     string format
Esc
```

[Fns...] Modules

When using a Python function or constant from a module, always use an import statement to indicate the module location of the function, method or constant.

See What is the Python programming experience?

[Fns…]>Modul: math module



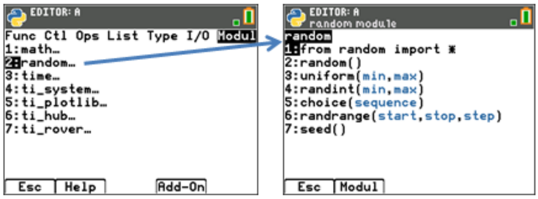
```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
Esc  Help        Add-On
```

```
EDITOR: A
math module
Math Const Trig
1:from math import *
2:fabs()
3:sqrt()
4:exp()
5:pow(x,y)
6:log(x,base)
7:fmod(x,y)
8:ceil()
9:floor()
0↓trunc()
A:frexp()
Esc  Modul
```

```
EDITOR: A
math module
Math Const Trig
1:e
2:pi
Esc  Modul
```

```
EDITOR: A
math module
Math Const Trig
1:radians()      degrees▸radians
2:degrees()      radians▸degrees
3:sin()
4:cos()
5:tan()
6:asin()
7:acos()
8:atan()
9:atan2(y,x)
Esc  Modul
```

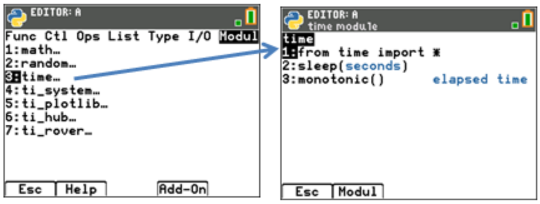
[Fns…]>Modul: random module



```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
Esc  Help        Add-On
```

```
EDITOR: A
random module
random
1:from random import *
2:random()
3:uniform(min,max)
4:randint(min,max)
5:choice(sequence)
6:randrange(start,stop,step)
7:seed()
Esc  Modul
```

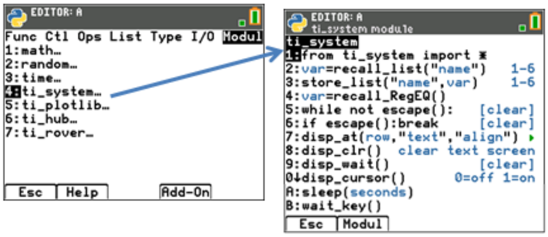
[Fns…]>Modul: time module



```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
Esc  Help        Add-On
```

```
EDITOR: A
time module
time
1:from time import *
2:sleep(seconds)
3:monotonic()       elapsed time
Esc  Modul
```

[Fns…]>Modul: ti_system module

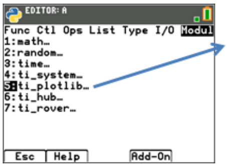


```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
Esc  Help        Add-On
```

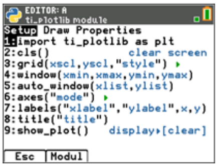
```
EDITOR: A
ti_system module
ti_system
1:from ti_system import *
2:var=recall_list("name")    1-6
3:store_list("name",var)     1-6
4:var=recall_RegEQ()
5:while not escape():    [clear]
6:if escape():break      [clear]
7:disp_at(row,"text","align") ▸
8:disp_clr()  clear text screen
9:disp_wait()            [clear]
0↓disp_cursor()      0=off 1=on
A:sleep(seconds)
B:wait_key()
Esc  Modul
```

See: Keypad mapping for wait_key()

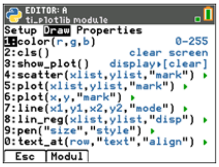
[Fns…]>Modul: ti_plotlib



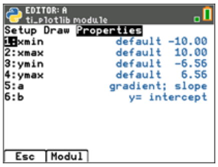
```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
Esc | Help |        | Add-On
```



```
EDITOR: A
ti_plotlib module
Setup Draw Properties
1:import ti_plotlib as plt
2:cls()               clear screen
3:grid(xscl,yscl,"style") ▸
4:window(xmin,xmax,ymin,ymax)
5:auto_window(xlist,ylist)
6:axes("mode") ▸
7:labels("xlabel","ylabel",x,y)
8:title("title")
9:show_plot()    display▸[clear]
Esc | Modul
```



```
EDITOR: A
ti_plotlib module
Setup Draw Properties
1:color(r,g,b)            0-255
2:cls()              clear screen
3:show_plot()    display▸[clear]
4:scatter(xlist,ylist,"mark") ▸
5:plot(xlist,ylist,"mark") ▸
6:plot(x,y,"mark") ▸
7:line(x1,y1,x2,y2,"mode") ▸
8:lin_reg(xlist,ylist,"disp") ▸
9:pen("size","style") ▸
0:text_at(row,"text","align") ▸
Esc | Modul
```



```
EDITOR: A
ti_plotlib module
Setup Draw Properties
1:xmin          default -10.00
2:xmax          default  10.00
3:ymin          default  -6.56
4:ymax          default   6.56
5:a            gradient; slope
6:b               y= intercept
Esc | Modul
```

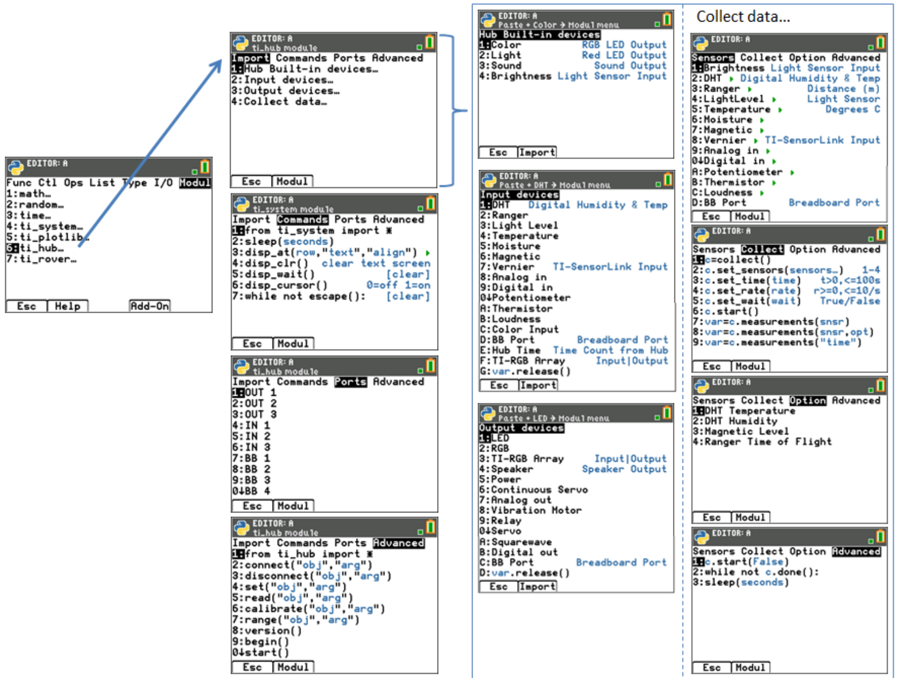
Important Plotting Note:

- The order of program lines for plotting must follow the order as in the Setup menu to ensure expected results.
- Plotting displays when plt.show_plot() is executed at the end of the plotting objects in a program. To clear the plotting area in the Shell, press [clear]. To view the Shell history, press [2nd] [▲] and [2nd] [▼].
- Running a second program that assumes the default values are set within the same Shell environment, will generally result in unexpected behavior such as color or other default argument settings. Edit programs with expected argument values or Reinitialize the Shell before running another plotting program.

[Fns…]>Modul: ti_hub module

ti_hub methods are not listed in Catalog and thus, not listed in the Reference Guide. Please use the screen information in the menus for arguments and argument default or allowed value details. More information on Python programming for TI-Innovator™ Hub and TI-Innovator™ Rover will be available at education.ti.com.

Note: TI-Innovator™ Hub should be connected when you run your Python programs.



```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
 Esc | Help |          |Add-On
```

```
EDITOR: A
ti_hub module
Import Commands Ports Advanced
1:Hub Built-in devices…
2:Input devices…
3:Output devices…
4:Collect data…
 Esc |Modul
```

```
EDITOR: A
ti_system module
Import Commands Ports Advanced
1:from ti_system import *
2:sleep(seconds)
3:disp_at(row,"text","align") ▸
4:disp_clr()  clear text screen
5:disp_wait()             [clear]
6:disp_cursor()       0=off 1=on
7:while not escape():     [clear]
 Esc |Modul
```

```
EDITOR: A
ti_hub module
Import Commands Ports Advanced
1:OUT 1
2:OUT 2
3:OUT 3
4:IN 1
5:IN 2
6:IN 3
7:BB 1
8:BB 2
9:BB 3
0↓BB 4
 Esc |Modul
```

```
EDITOR: A
ti_hub module
Import Commands Ports Advanced
1:from ti_hub import *
2:connect("obj","arg")
3:disconnect("obj","arg")
4:set("obj","arg")
5:read("obj","arg")
6:calibrate("obj","arg")
7:range("obj","arg")
8:version()
9:begin()
0↓start()
 Esc |Modul
```

```
EDITOR: A
Paste ♦ Color → Modul menu
Hub Built-in devices
1:Color           RGB LED Output
2:Light           Red LED Output
3:Sound             Sound Output
4:Brightness Light Sensor Input
 Esc |Import
```

```
EDITOR: A
Paste ♦ DHT → Modul menu
Input devices
1:DHT   Digital Humidity & Temp
2:Ranger
3:Light Level
4:Temperature
5:Moisture
6:Magnetic
7:Vernier    TI-SensorLink Input
8:Analog in
9:Digital in
0↓Potentiometer
A:Thermistor
B:Loudness
C:Color Input
D:BB Port        Breadboard Port
E:Hub Time   Time Count from Hub
F:TI-RGB Array      Input|Output
G:var.release()
 Esc |Import
```

```
EDITOR: A
Paste ♦ LED → Modul menu
Output devices
1:LED
2:RGB
3:TI-RGB Array      Input|Output
4:Speaker         Speaker Output
5:Power
6:Continuous Servo
7:Analog out
8:Vibration Motor
9:Relay
0↓Servo
A:Squarewave
B:Digital out
C:BB Port        Breadboard Port
D:var.release()
 Esc |Import
```

Collect data…

```
EDITOR: A
Sensors Collect Option Advanced
1:Brightness Light Sensor Input
2:DHT ▸ Digital Humidity & Temp
3:Ranger ▸          Distance (m)
4:LightLevel ▸      Light Sensor
5:Temperature ▸        Degrees C
6:Moisture ▸
7:Magnetic ▸
8:Vernier ▸ TI-SensorLink Input
9:Analog in ▸
0↓Digital in ▸
A:Potentiometer ▸
B:Thermistor ▸
C:Loudness ▸
D:BB Port        Breadboard Port
 Esc |Modul
```

```
EDITOR: A
Sensors Collect Option Advanced
1:c=collect()
2:c.set_sensors(sensors…)    1-4
3:c.set_time(time)   t>0,<=100s
4:c.set_rate(rate)  r>=0,<=10/s
5:c.set_wait(wait)   True/False
6:c.start()
7:var=c.measurements(snsr)
8:var=c.measurements(snsr,opt)
9:var=c.measurements("time")
 Esc |Modul
```

```
EDITOR: A
Sensors Collect Option Advanced
1:DHT Temperature
2:DHT Humidity
3:Magnetic Level
4:Ranger Time of Flight
 Esc |Modul
```

```
EDITOR: A
Sensors Collect Option Advanced
1:c.start(False)
2:while not c.done():
3:sleep(seconds)
 Esc |Modul
```

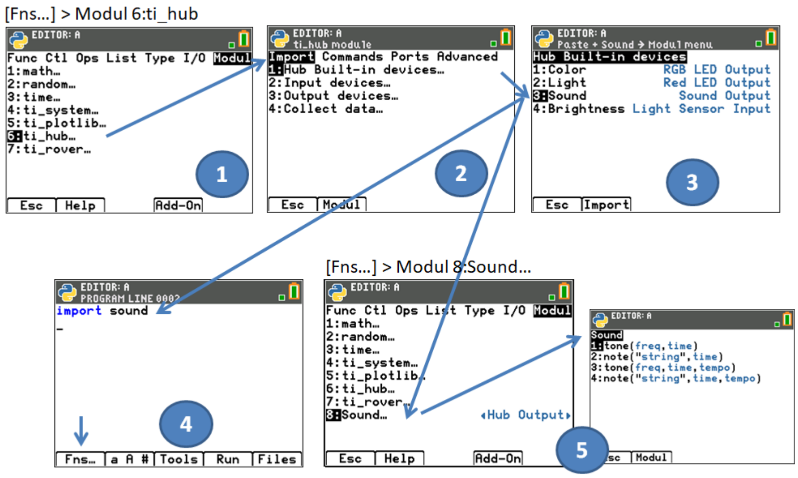
ti_hub module – Add import to Editor and add ti_hub sensor module to the Modul menu

Screen Example: Import sound

To import TI-Innovator™ sensor methods to your Python program, from the Editor,

1. Select **[Fns…] > Modul 6:ti_hub**
2. Select the ti_hub Import menu. Select a sensor type from Built-in, Input and Output.
3. Select a sensor.
4. An import statement will paste to the Editor and the sensor module will be available in **[Fns...] > Modul** when you return to that menu from your program.
5. Select **[Fns…] > Modul 8:Sound...** to paste appropriate methods for this sensor.

[Fns…]>Modul 6:ti_hub



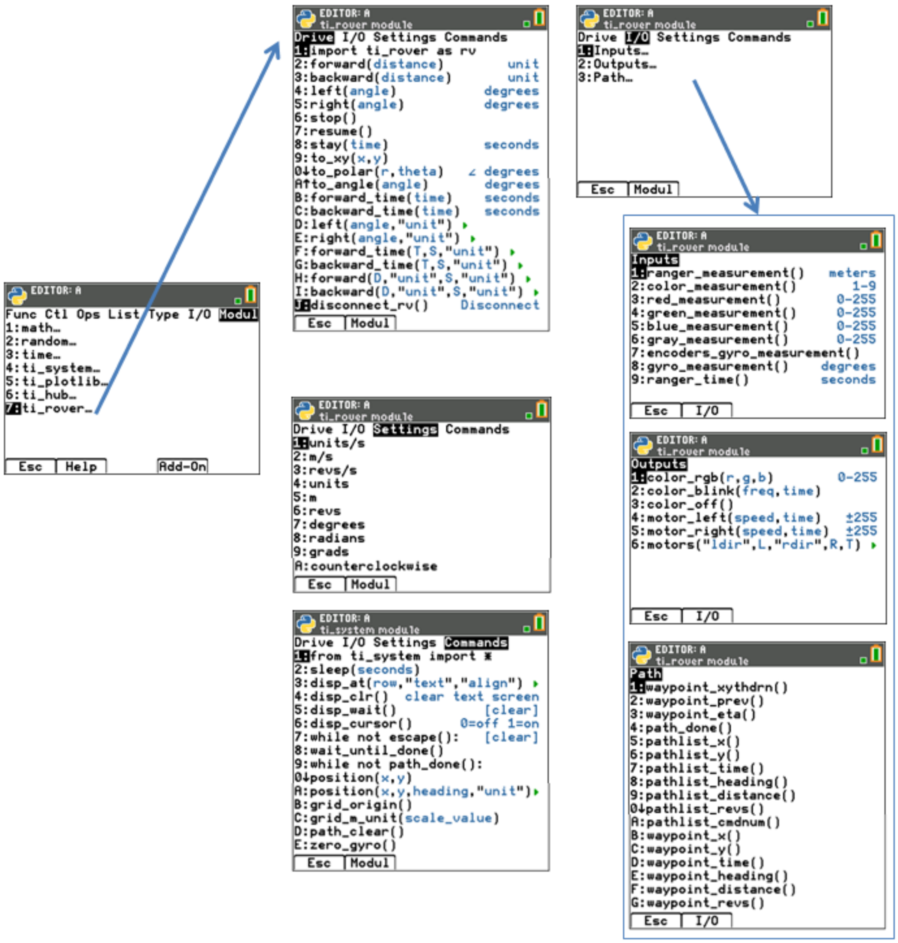
Note: Brightns is a "built-in" object on TI-Innovator Hub.
When using the 'import brightns' statement, enter 'brightns.range(0,100)' to ensure the correct default range at the start of the program execution.

Example:

import brightns
brightns.range(0,100)
b=brightns.measurement()
print(b)

[Fns…]>Modul ti_rover module

ti_rover methods are not listed in Catalog and thus, not listed in the Reference Guide. Please use the screen information in the menus for arguments and argument default or allowed value details. More information on Python programming for TI-Innovator™ Hub and TI-Innovator™ Rover will be available at education.ti.com.



```
EDITOR: A
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
Esc  Help           Add-On
```

```
EDITOR: A
ti_rover module
Drive I/O Settings Commands
1:import ti_rover as rv
2:forward(distance)           unit
3:backward(distance)          unit
4:left(angle)              degrees
5:right(angle)             degrees
6:stop()
7:resume()
8:stay(time)               seconds
9:to_xy(x,y)
0↓to_polar(r,theta)      ∠ degrees
A↑to_angle(angle)          degrees
B:forward_time(time)       seconds
C:backward_time(time)      seconds
D:left(angle,"unit") ▸
E:right(angle,"unit") ▸
F:forward_time(T,S,"unit") ▸
G:backward_time(T,S,"unit") ▸
H:forward(D,"unit",S,"unit") ▸
I:backward(D,"unit",S,"unit") ▸
J:disconnect_rv()       Disconnect
Esc  Modul
```

```
EDITOR: A
ti_rover module
Drive I/O Settings Commands
1:Inputs…
2:Outputs…
3:Path…
Esc  Modul
```

```
EDITOR: A
ti_rover module
Inputs
1:ranger_measurement()      meters
2:color_measurement()          1-9
3:red_measurement()          0-255
4:green_measurement()        0-255
5:blue_measurement()         0-255
6:gray_measurement()         0-255
7:encoders_gyro_measurement()
8:gyro_measurement()       degrees
9:ranger_time()            seconds
Esc  I/O
```

```
EDITOR: A
ti_rover module
Drive I/O Settings Commands
1:units/s
2:m/s
3:revs/s
4:units
5:m
6:revs
7:degrees
8:radians
9:grads
A:counterclockwise
Esc  Modul
```

```
EDITOR: A
ti_rover module
Outputs
1:color_rgb(r,g,b)           0-255
2:color_blink(freq,time)
3:color_off()
4:motor_left(speed,time)      ±255
5:motor_right(speed,time)     ±255
6:motors("ldir",L,"rdir",R,T) ▸
Esc  I/O
```

```
EDITOR: A
ti_system module
Drive I/O Settings Commands
1:from ti_system import *
2:sleep(seconds)
3:disp_at(row,"text","align") ▸
4:disp_clr()  clear text screen
5:disp_wait()          [clear]
6:disp_cursor()      0=off 1=on
7:while not escape():  [clear]
8:wait_until_done()
9:while not path_done():
0↓position(x,y)
A:position(x,y,heading,"unit")▸
B:grid_origin()
C:grid_m_unit(scale_value)
D:path_clear()
E:zero_gyro()
Esc  Modul
```

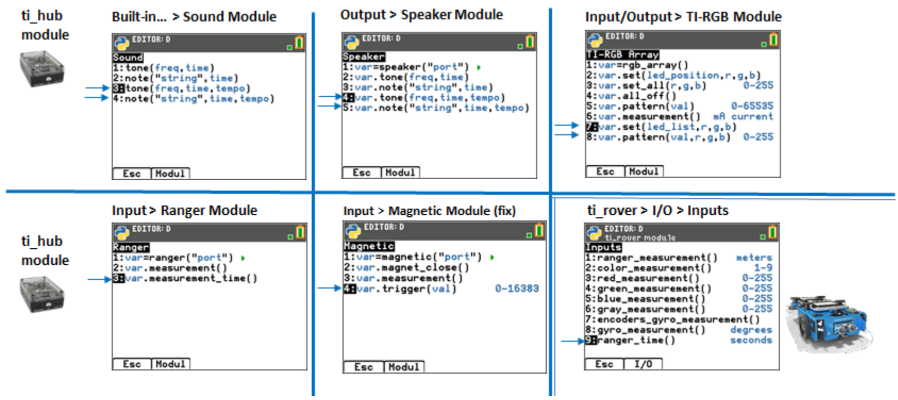
```
EDITOR: A
ti_rover module
Path
1:waypoint_xythdrn()
2:waypoint_prev()
3:waypoint_eta()
4:path_done()
5:pathlist_x()
6:pathlist_y()
7:pathlist_time()
8:pathlist_heading()
9:pathlist_distance()
0↓pathlist_revs()
A:pathlist_cmdnum()
B:waypoint_x()
C:waypoint_y()
D:waypoint_time()
E:waypoint_heading()
F:waypoint_distance()
G:waypoint_revs()
Esc  I/O
```

Notes:

- In TI-Python programming, you do not need to include methods to connect and disconnect TI-Innovator™ Rover. The TI-Innovator™ Rover Python methods handle connect and disconnect with no additional methods. This is a bit different than programming TI-Innovator™ Rover in TI-Basic.

- rv.stop() executes as a pause and then resume continues with the Rover movements in the queue. If another movement command is executed after rv.stop(), then movement queue is cleared. This again is a bit different than programming TI-Innovator™ Rover in TI-Basic.

Python Support for TI-Innovator Sketch v1.5



```
EDITOR: D
Sound
1:tone(freq,time)
2:note("string",time)
3:tone(freq,time,tempo)
4:note("string",time,tempo)
Esc  Modul
```

```
EDITOR: D
Speaker
1:var=speaker("port")
2:var.tone(freq,time)
3:var.note("string",time)
4:var.tone(freq,time,tempo)
5:var.note("string",time,tempo)
Esc  Modul
```

```
EDITOR: D
TI-RGB Array
1:var=rgb_array()
2:var.set(led_position,r,g,b)
3:var.set_all(r,g,b)         0-255
4:var.all_off()
5:var.pattern(val)         0-65535
6:var.measurement()  mA current
7:var.set(led_list,r,g,b)
8:var.pattern(val,r,g,b)  0-255
Esc  Modul
```

```
EDITOR: D
Ranger
1:var=ranger("port")
2:var.measurement()
3:var.measurement_time()
Esc  Modul
```

```
EDITOR: D
Magnetic
1:var=magnetic("port")
2:var.magnet_close()
3:var.measurement()
4:var.trigger(val)          0-16383
Esc  Modul
```

```
EDITOR: D
ti_rover module
Inputs
1:ranger_measurement()    meters
2:color_measurement()        1-9
3:red_measurement()        0-255
4:green_measurement()      0-255
5:blue_measurement()       0-255
6:gray_measurement()       0-255
7:encoders_gyro_measurement()
8:gyro_measurement()     degrees
9:ranger_time()          seconds
Esc   I/O
```

[Fns...] Add-On Modules

Add-On modules enhance Python App module experience with additional functionality and easy access to the additional Python methods from menus in the Python App.

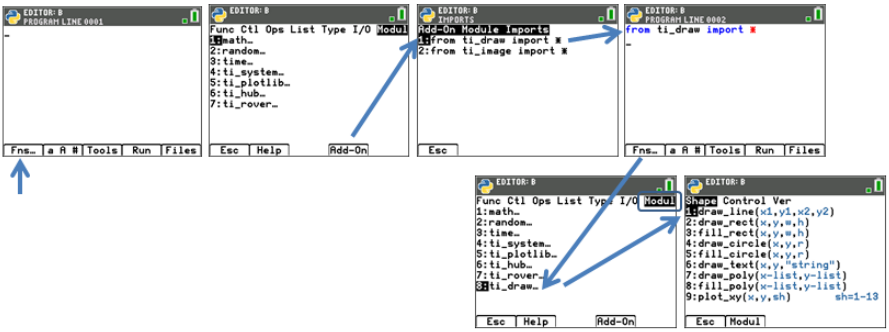
You may notice an Add-On module to load, using TI Connect™ CE, as part of a Python activity posted on education.ti.com such as ce_turtl, ce_chart, ce_box, ce_quivr, and microbit depending on your region. You will need to have the latest version of currently posted Add-On modules Some Add-On modules will load to your calculator, such as ti_draw and ti_image, when you update with the latest CE Bundle.

The Python App will display the Add-On module menus in the [Fns…] > Modul menu only if your program in the Editor starts with an appropriate import statement.

Pasting an Add-On Module import statement to the Editor

Steps:

1. Create a new program.
2. In Editor, select [Fns…] > Modul.
3. Select [Add-On] and when an Add-On module is loaded on the calculator, an import statement menu for loaded modules will display.
4. Select the import statement to paste to the Editor.
5. Select [Fns…] > Modul to locate the menus for the Add-On imported module.



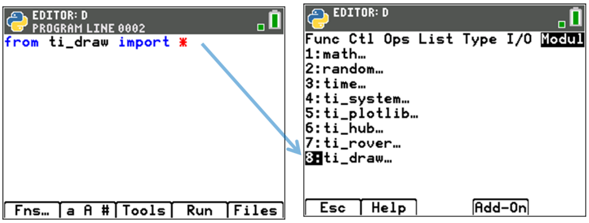
```
from ti_draw import *
```

Facts:

- [Add-On Modules Imports…] is also listed in [2nd] [catalog].
- Add-On modules are calculator "AppVar" files stored in Archive and appear in [mem] as an AppVar. It is recommended to keep these files in Archive memory for the enhanced Python App module experience.
- A Python program runs in the Python App from File Manager or Editor when the "PY AppVar" program is in RAM. If a PY AppVar Python program is placed in Archive memory, it will not be available to Run or Edit in the Python App.

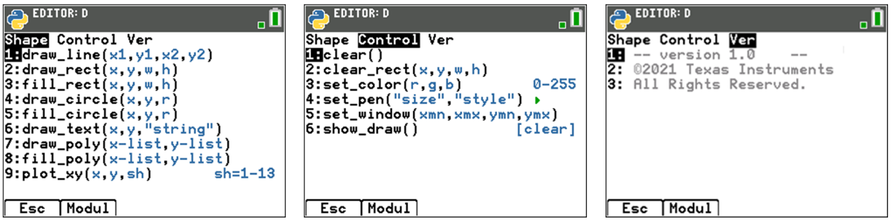
[Fns...] ti_draw Add-On Module

The ti_draw module is included in the latest CE Bundle. Use the [Fns…] > Modul [Add-On] feature to paste the import statement to your program. The ti_draw menu will then display in the [Fns…] > Modul menu as shown here.



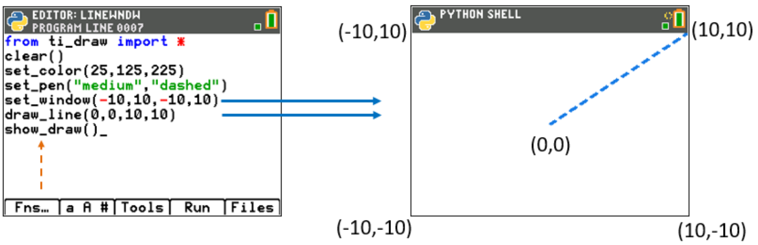
```
from ti_draw import *
```

ti_draw menus



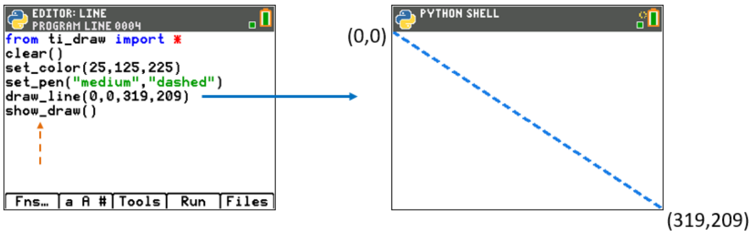
Program information when using ti_draw:

- After the import statement, use the clear() method to clear the Shell drawing area if needed.
- Programs must contain the show_draw() command to display the draw when running the program.
- Using draw_rect(), draw_circle(), or draw_poly() methods draw the border of the construction whereas fill_rect(), fill_circle(), and fill_poly() methods fill the interior of the specified shape (dependent on pen size).
- Press [clear] to clear the draw and return to the Shell prompt. Please note: Shell history can be viewed using [2nd] ▲ and [2nd] ▼.
- Please read through the Shape and Control menu information in the table below. Your drawings created with the methods in the Shape menu will depend on Control menu methods such as set_color() and set_ pen().
- **Coordinate arguments** are either screen pixel coordinates or set by the set_window () method.
    - ti_draw methods using set_window() coordinates



```
from ti_draw import *
clear()
set_color(25,125,225)
set_pen("medium","dashed")
set_window(-10,10,-10,10)
draw_line(0,0,10,10)
show_draw()_
```

    - ti_draw methods using pixel screen coordinates



```
from ti_draw import *
clear()
set_color(25,125,225)
set_pen("medium","dashed")
draw_line(0,0,319,209)
show_draw()
```

	Shape Menu	Description
1:	draw_line(x1,y1,x2,y2)	Draws a line segment between the specified points (x1,y1) and (x2,y2).
2:	draw_rect(x,y,w,h)	Draws a rectangle with upper left corner at (x,y) with a width of w pixels and height of h.

	Shape Menu	Description
3:	fill_rect(x,y,w,h)	Fills the interior of a rectangle with upper left corner at (x,y) with a width of w pixels and height of h pixels.
4:	draw_circle(x,y,r)	Draws a circle with center located at (x,y) and a radius of r pixels.
5:	fill_circle(x,y,r)	Draws a circle with center located at (x,y) and a radius of r pixels and filled with the specified color (using set_color or black if not defined).
6:	draw_text(x,y,"string")	Draws the string as text on the display with upper left corner of the text starting at (x,y).
7:	draw_poly(x-list,y-list)	Draws a set of lines that may represent a polygon. The lines are drawn using the current pen size and pen color.
8:	fill_poly(x-list,y-list)	The x-list and y-list must be of equal-length of list arguments into a list of (x,y) vertices. The polygon is drawn by connecting each pair of vertices and filling the region with the current pen color.
9:	poly_xy(x,y,sh) sh=1-13	Using the x and y arguments as a center-point location, the requested shape (sh) value below will draw. Shapes are drawn using the current pen color.

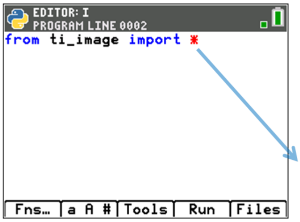
Shape	Description
1	Filled circle of radius 2
2	Open circle of radius 2
3	3x3 filled square
4	3x3 open square
5	x mark is drawn
6	+ mark is drawn
7	Single pixel
8	Filled circle with radius 4 pixels
9	Open circle with radius 4 pixels
10	Filled circle with radius 6 pixels
11	Open circle with radius 6 pixels
12	Filled circle with radius 8 pixels
13	Open circle with radius 8 pixels

	Control Menu	Description
1:	clear()	Clears the drawing area in the Shell. This method must be executed prior to drawing to ensure the Shell drawing area is cleared to view expected results.
2:	clear_rect(x,y,w,h)	Fills the interior of a rectangle with upper left corner at (x,y) and width w height h. White is the default fill color. After pasting the method to the Editor, the method can accept a fifth optional argument to specify a different color via the use of a tuple specifying (r,g,b) value. A valid (r,g,b) tuple contains integer values in range 0 to 255.
3:	set_color(r,g,b) 0-255	Sets the drawing pen color using (r,g,b) tuple.
4:	set_pen("size","style")	Sets the drawing pen to the “size” and “style” for all subsequent drawings until a change is specified. When importing ti_draw, size is "thin", "medium", or "thick" and style is "solid", "dotted", or "dashed". If not specified, default arguments are "thin" and "solid." The Argument Helper > will help fill the correct argument strings. **Note:** When importing ti_plotlib module, pen() method style argument is "solid", "dot", or "dash".
5:	set_window (xmn,xmx,ymn,ymx)	Sets the draw area with coordinates ranges [xmin,xmax] and [ymin,ymax] with (0,0) at midpoint of the ranges. Please note: If argument values are outside of the draw area specified, no error is given. If set_window(xmin,xmax,ymin,ymax) is not executed in a program, the pixel window window size is the default with (xmin,xmax,ymin,ymax) = (0,319,0,209) with (0,0) at upper left hand corner pixel coordinate of the area.
6:	show_draw() [clear]	Must be included to display the draw. Press [clear] to clear the draw and

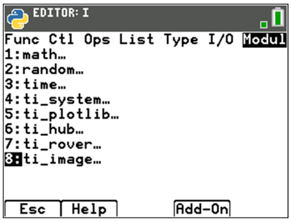
	Control Menu	Description
		return to the Shell prompt. To view Shell history, press [2nd] ▲ and [2nd] ▼.

[Fns...] ti_image Add-On Module

The ti_image module is included in the latest CE Bundle. Use the [Fns…] > Modul [Add-On] feature to paste the import statement to your program. The ti_image menu will then display in the [Fns…] > Modul menu as shown here.

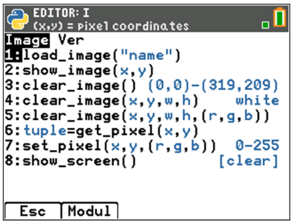


```
EDITOR: I
PROGRAM LINE 0002
from ti_image import *
```

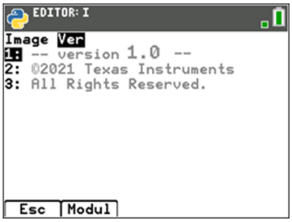


```
EDITOR: I
Func Ctl Ops List Type I/O Modul
1:math…
2:random…
3:time…
4:ti_system…
5:ti_plotlib…
6:ti_hub…
7:ti_rover…
8:ti_image…
```

ti_image menus



```
EDITOR: I
(x,y) = pixel coordinates
Image Ver
1:load_image("name")
2:show_image(x,y)
3:clear_image() (0,0)-(319,209)
4:clear_image(x,y,w,h)     white
5:clear_image(x,y,w,h,(r,g,b))
6:tuple=get_pixel(x,y)
7:set_pixel(x,y,(r,g,b))   0-255
8:show_screen()          [clear]
```



```
EDITOR: I
Image Ver
1: -- version 1.0 --
2: ©2021 Texas Instruments
3: All Rights Reserved.
```

Program information when using ti_image:

- The ti_image module can be used to
  - display your named Python image AppVar stored in Archive memory on your CE.
  - display specified filled color rectangles in a pixel coordinate location.
  - set or get a pixel color.
  - clear the interior of a rectangular screen area.
  - clear the full screen drawing area in Shell from pixel coordinates (0,0) to (319,209).
- After the import statement, use the clear_image() method to clear the Shell drawing area if needed.
- Python image AppVar is a special Python image file (*.8xv).
  - Currently, a library of curriculum images are posted at https://resources.t3europe.eu/t3europe-home?country=15&q=images&cHash=d50a2b65ab1b875dfa3ec11bca12154c
  - When using a Python image AppVar, it is best practice to
    - store the Python image AppVar in Archive memeory. [2nd][mem]
    - know the pixel dimensions of your image for use during coding.
    - know the exact name of your Python image AppVar. You must enter the name with the correct spelling and correct upper and lower case letters. No errors will be given for mistyped Python image AppVar names.

- Keep updated with the latest TI Connect™ CE and TI-SmartView™ CE at education.ti.com/84cetupdate

- (x,y) coordinate arguments are pixel coordinates ONLY in ti_image methods and range from (0,0) to (319,209). Please read more information on each method in the table below. Some methods are offered to paste to the Editor in several formats when optional arguments are offered.
- Press [clear] to clear the draw and return to the Shell prompt. Shell history can be viewed using [2nd] ▲ and [2nd] ▼.

	Control Menu	**Description**
1:	load_image("name")	Loads a Python image AppVar "name" for use in the program. The Python image "name" must be in the exact case and spelling of the Python image AppVar. Please note: There is no error message generated if the AppVar name is NOT specified exactly as named. Python image "name" will be the image used for display in show_image(x,y). **Best Practices:** • Know the pixel dimensions of your Python image. • Memory Tip: Python image AppVars should be stored in Archive Memory.
2:	show_image(x,y)	Displays the image specified in load_ image(“name”). Displays the image with upper left pixel corner (x,y) of the drawing area in the Shell. (x,y) screen pixel coordinates range from upper left as (0,0) to lower right (319,209). If no image name has been specified using load_image(), an error is reported when the program Runs. If “name” is incorrectly entered, no error will display. Use show_screen() method to retain the image in display until [clear] to return to the Shell. To view Shell history, press [2nd] ▲ and [2nd] ▼.

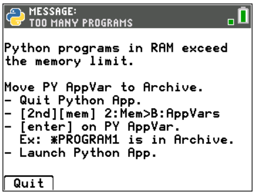
	Control Menu	Description
3:	clear_image() (0,0)-(319,209)	The clear_image() method with no arguments is used to clear the drawing area of the Shell. The drawing area will display as the white screen. The pixels coordinates range from upper left as (0,0) to lower right (319,209). After the full drawing area is "cleared" with this method, use load_image ("name") method and show_image(x,y) to display the “name” image as needed. When also using ti_draw module methods, note that set_pen() color will be set to black when ti_image method, clear_image(), is executed.
4:	clear_image(x,y,w,h) white	Given an (x,y) pixel coordinate for the upper left corner of a rectangle w pixels wide and h pixels high, this method will "clear" the interior rectangular area to white.
5:	clear_image(x,y,w,h,(r,g,b))	Given an (x,y) pixel coordinate for the upper left corner of a rectangle w pixels wide and h pixels high, this method will "clear" the interior rectangular area to the specified RGB color in given in the tuple (r,g,b).
6:	tuple=get_pixel(x,y)	Returns RGB values of the pixel at pixel coordinate (x,y) as a tuple (r,g,b).
7:	set_pixel(x,y,(r,g,b))	Sets the color of the pixel at pixel coordinate (x,y) to the RGB color specified in (r,g,b).
8:	show_screen() [clear]	This method must be used to retain the display of the drawing on the screen when using the ti_image module. When [clear] is pressed after each instance of show_screen(), the program will continue to run until finally clearing the screen to the Shell prompt. To view Shell history, use [2nd] ▲ and [2nd] ▼. See Shell > [Tools] for more Shell options.

Python App Messages

There are several messages that may display while you are in a Python session. Some selected messages are given in the table. Please follow the instructions on the screen and navigate using [Quit], [Esc] or [Ok] as needed.

Memory Management

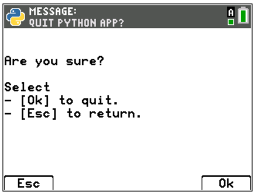
The available memory for the Python experience will be a maximum of 100 Python programs (PY AppVars) or 50K of memory. The modules that are bundled with the app in this Python release will share the same space with all files.



```
MESSAGE:
TOO MANY PROGRAMS

Python programs in RAM exceed
the memory limit.

Move PY AppVar to Archive.
- Quit Python App.
- [2nd][mem] 2:Mem>B:AppVars
- [enter] on PY AppVar.
  Ex: *PROGRAM1 is in Archive.
- Launch Python App.

Quit
```

Use [2nd] [quit] to quit the App

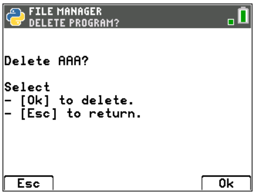
You will be prompted to make sure you want to quit the App. Quitting the App will stop your Python session. When you run the Python App again, your Python AppVar programs and modules will synchronize. The Shell will reinitialize.



```
MESSAGE:
QUIT PYTHON APP?

Are you sure?

Select
- [Ok] to quit.
- [Esc] to return.

Esc                          Ok
```

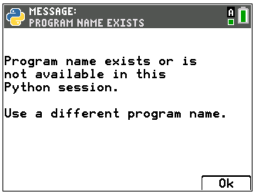
In File Manager, you press [del] on a selected Python program or you select from **File Manager>Manage 2:Delete Program...**.

You will see a dialog to delete or escape back to the File Manager.



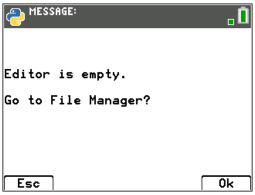
```
FILE MANAGER
DELETE PROGRAM?

Delete AAA?

Select
- [Ok] to delete.
- [Esc] to return.

Esc                          Ok
```

You attempt to create a new or duplicate a Python program that already exists on your CE either in RAM or Archive or disabled for exam mode. Please enter a different name.



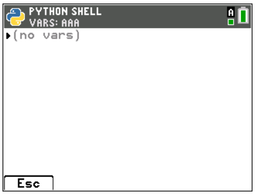
```
MESSAGE:
PROGRAM NAME EXISTS

Program name exists or is
not available in this
Python session.

Use a different program name.

                             Ok
```

You attempt to navigate from the Shell to the Editor but the Editor is empty. Please select an appropriate option for your work.



```
MESSAGE:

Editor is empty.

Go to File Manager?

Esc                          Ok
```

When you execute a Python program, defined variables from the last program executed are listed in the **Shell>Tools> 4:Vars…** menu to use and are avaliable for use in the Shell. If no variables display, you may need to run your program again.



```
▸(no vars)
```

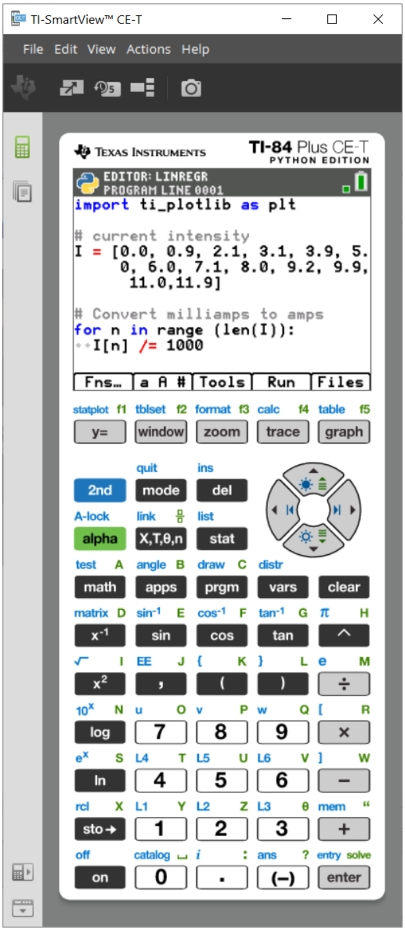
Using TI-SmartView™ CE-T and the Python Experience

This guidebook assumes the latest update of TI-SmartView™ CE-T. Update to the latest TI-SmartView™ CE-T at education.ti.com/84cetupdate.

The update includes the latest TI-84 Plus CE-T *Python Edition* emulator OS running the latest Python App. The updated modules of time, ti_system, ti_plotlib, ti_rover* and ti_hub* are included.

Run the Python App on the TI-84 Plus CE-T *Python Edition* emulator.

- The Python App offers
  - File Manager
  - Editor
  - Execution of your Python program in the Shell*



```
EDITOR: LINREGR
PROGRAM LINE 0001
import ti_plotlib as plt

# current intensity
I = [0.0, 0.9, 2.1, 3.1, 3.9, 5.
   0, 6.0, 7.1, 8.0, 9.2, 9.9,
    11.0,11.9]

# Convert milliamps to amps
for n in range (len(I)):
••I[n] /= 1000
Fns… a A # Tools Run Files
```

Hub/Rover Programs

- Create ti_hub/ti_rover Python programs in the CE emulator running the Python App.

  *** Note:** There is no connectivity between TI-SmartView™ CE-T and TI-Innovator™ Hub or TI-Innovator™ Rover. Programs can be created and then run on the CE calculator.

- Quit the Python App to prepare to transfer the Python AppVar(s) from the emulator. The emulator should not “be busy” running an App or program for the next step.

- Change to the Emulator Explorer workspace and send the program(s) to the computer.
- Use TI Connect™ CE to send the Python AppVars from the computer to the CE calculator for the TI-Innovator™ Hub/TI-Innovator™ Rover experience.

Note: To break a running Python program in the Shell, such as if a program is in a continuous loop, press **[on]**. Press **[Tools] [zoom] > 6:New Shell** as an alternate method to halt a running program.

Reminder: For any computer/TI-Python experience: After creating a Python program in a Python development environment on the computer, please validate your program runs on the calculator/emulator in the TI-Python experience. Modify the program as needed.

SmartPad CE App Remote Keypad

- When running the SmartPad CE App on your connected CE will behave as a remote keypad including the special *keypad* mapping offered when the Python App is running.

Emulator Explorer Workspace

- Please quit the Python App so the emulator is not busy when you access the full features of the Emulator Explorer workspace.
- program.py < > PY AppVar conversions are allowed. This is similar to the TI Connect™ CE experience when sending programs to the connected CE calculator.
- When sending a program.py file created in another Python environment, your PY AppVar will need to be edited to run as expected in TI-Python. Use the Python App Editor to modify as needed for the unique modules such as ti_plotlib, ti_system, ti_hub and ti_rover.

Data Import Wizard

- *.csv files of data, formatted as stated in the wizard dialog, will convert data into CE list variables. Methods in ti_system can then be used to share lists between the emulator CE OS and the Python App. This feature is similar to the Data Import Wizard in TI Connect™ CE.
- If decimal numbers are represented with the use of a comma in the *.csv file, the file will not convert using the Data Import Wizard. Please check your computer operating system number formatting and convert the *.csv to use the decimal point representation. The CE calculator list and matrix editor use the number format as, for example, 12.34 and not 12,34.

Using TI Connect™ CE to Convert Python Programs

Please update to TI Connect™ CE for the latest features including converting *.py programs to a PY AppVar as the CE calculator file format.

See TI-84 Plus CE-T e-Guide for more details on the CE calculator, TI-SmartView™ CE-T and TI Connect™ CE.

What is the Python programming experience?

TI-Python is based on CircuitPython, a variant of Python designed to fit in small microcontrollers. The original CircuitPython implementation has been adapted for use by TI.

The internal storage of numbers for computation in this variant of Circuit Python is in limited-precision binary floats and thus cannot exactly represent all possible decimal values. The differences from actual decimal representations that arise when storing these values can lead to unexpected results in subsequent calculations.

- **For Floats** - Displays up to 16 significant digits of precision. Internally, values are stored using 53 bits of precision, which is roughly equivalent to 15-16 decimal digits.
- **For Integers** - The size of integers is limited only by the memory available at the time calculations are performed.

Modules Included in the TI-84 Plus CE-T Python Edition

- Built-ins
- math module
- random module
- time
- ti_system
- ti_plotlib
- ti_hub
- ti_rover

Note: If you have existing Python programs created in other Python development environments, please edit your program(s) to the TI-Python solution. Modules may use different methods, arguments, and ordering of methods in a program as compared to the ti_system, ti_plotlib, ti_hub, and ti_rover modules. In general, be aware of compatibility when using any version of Python and Python modules.

When transferring Python programs from a non-TI platform to a TI platform OR from one TI product to another:

- Python programs that use core language features and standard libs (math, random etc.) can be ported without changes

  Note: List length limit is 100 elements.

- Programs that use platform-specific libraries - matplotlib (for PC), ti_plotlib, ti_system, ti_hub, etc. for TI platforms, will require edits before they will run on a different platform.
- This may be true even between TI platforms.

As with any version of Python, you will need to include imports such as, from math import *, to use any functions, methods, or constants contained in the math module. For an example, to execute the cos() function, use import to import the math module for use.

See CATALOG Listing.

Example:

```
>>>from math import *
>>>cos(0)
1.0
```

Alternate Example:

```
>>>import math
>>>math.cos(0)
1.0
```

Modules available can be displayed in the Shell using the following command

```
>>> help("modules")
__main__ sys gc
random time array
math builtins collections
```

Content of modules can be viewed in the Shell as shown using “import module” and “dir(module).”

Not all module contents appear in the quick paste menus such as [Fns…] or **2nd** [**catalog**].

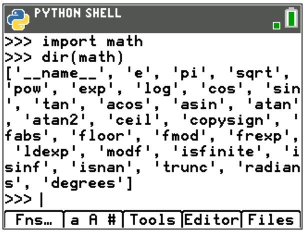
Contents of selected modules and keywords

For list of the modules included in this release, please see:

Appendix: Selected TI-Python Built-in, Keywords, and Module Content

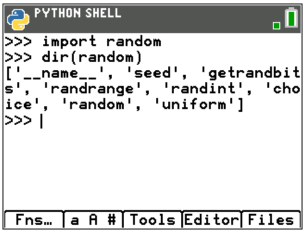
Reminder: For any computer/TI-Python experience: After creating a Python program on the computer, please validate that your program runs on the calculator in the TI-Python experience. Modify the program as needed.

These screens display the module contents for math and random.



```
>>> import math
>>> dir(math)
['__name__', 'e', 'pi', 'sqrt',
'pow', 'exp', 'log', 'cos', 'sin
', 'tan', 'acos', 'asin', 'atan'
, 'atan2', 'ceil', 'copysign', '
fabs', 'floor', 'fmod', 'frexp',
 'ldexp', 'modf', 'isfinite', 'i
sinf', 'isnan', 'trunc', 'radian
s', 'degrees']
>>> |
```

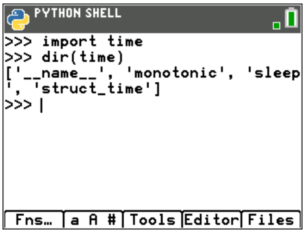
math module



```
>>> import random
>>> dir(random)
['__name__', 'seed', 'getrandbit
s', 'randrange', 'randint', 'cho
ice', 'random', 'uniform']
>>> |
```

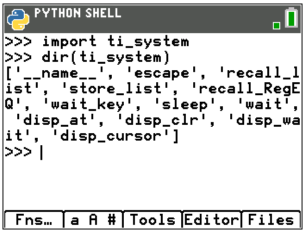
random module

These screens display the module contents for time and ti_system.



```
>>> import time
>>> dir(time)
['__name__', 'monotonic', 'sleep
', 'struct_time']
>>> |
```

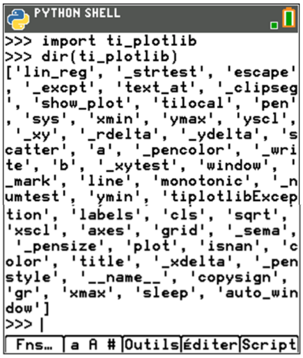
time



```
>>> import ti_system
>>> dir(ti_system)
['__name__', 'escape', 'recall_l
ist', 'store_list', 'recall_RegE
Q', 'wait_key', 'sleep', 'wait',
 'disp_at', 'disp_clr', 'disp_wa
it', 'disp_cursor']
>>> |
```

ti_system

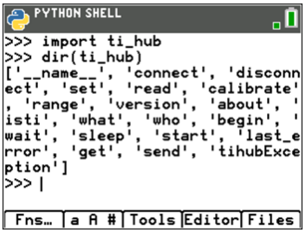
These screens display the module contents for ti_plotlib.



```
>>> import ti_plotlib
>>> dir(ti_plotlib)
['lin_reg', '_strtest', 'escape'
, '_excpt', 'text_at', '_clipseg
', 'show_plot', 'tilocal', 'pen'
, 'sys', 'xmin', 'ymax', 'yscl',
 '_xy', '_rdelta', '_ydelta', 's
catter', 'a', '_pencolor', '_wri
te', 'b', '_xytest', 'window', '
_mark', 'line', 'monotonic', '_n
umtest', 'ymin', 'tiplotlibExcep
tion', 'labels', 'cls', 'sqrt',
'xscl', 'axes', 'grid', '_sema',
 '_pensize', 'plot', 'isnan', 'c
olor', 'title', '_xdelta', '_pen
style', '__name__', 'copysign',
'gr', 'xmax', 'sleep', 'auto_win
dow']
>>> |
```

ti_plotlib

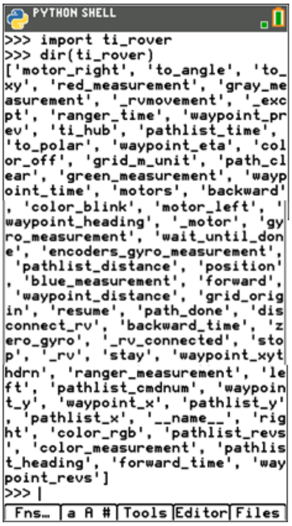
This screen displays the module contents for ti_hub.



```
>>> import ti_hub
>>> dir(ti_hub)
['__name__', 'connect', 'disconn
ect', 'set', 'read', 'calibrate'
, 'range', 'version', 'about', '
isti', 'what', 'who', 'begin', '
wait', 'sleep', 'start', 'last_e
rror', 'get', 'send', 'tihubExce
ption']
>>> |
```

ti_hub

These screens display the module contents for ti_rover.



```
>>> import ti_rover
>>> dir(ti_rover)
['motor_right', 'to_angle', 'to_
xy', 'red_measurement', 'gray_me
asurement', '_rvmovement', '_exc
pt', 'ranger_time', 'waypoint_pr
ev', 'ti_hub', 'pathlist_time',
'to_polar', 'waypoint_eta', 'col
or_off', 'grid_m_unit', 'path_cl
ear', 'green_measurement', 'wayp
oint_time', 'motors', 'backward'
, 'color_blink', 'motor_left', '
waypoint_heading', '_motor', 'gy
ro_measurement', 'wait_until_don
e', 'encoders_gyro_measurement',
 'pathlist_distance', 'position'
, 'blue_measurement', 'forward',
 'waypoint_distance', 'grid_orig
in', 'resume', 'path_done', 'dis
connect_rv', 'backward_time', 'z
ero_gyro', '_rv_connected', 'sto
p', '_rv', 'stay', 'waypoint_xyt
hdrn', 'ranger_measurement', 'le
ft', 'pathlist_cmdnum', 'waypoin
t_y', 'waypoint_x', 'pathlist_y'
, 'pathlist_x', '__name__', 'rig
ht', 'color_rgb', 'pathlist_revs
', 'color_measurement', 'pathlis
t_heading', 'forward_time', 'way
point_revs']
>>> |
```

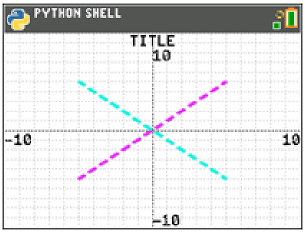
ti_rover

Sample Programs

Use the following Sample Programs to become familiar with methods from the Reference section. These samples also contain several TI-Innovator™ Hub and TI-Innovator Rover™ programs to help you get started with TI-Python.

COLORLIN

```
import ti_plotlib as plt
plt.cls()
plt.window(-10,10,-10,10)
plt.axes("on")
plt.grid(1,1,"dot")
plt.title("TITLE")
plt.pen("medium","solid")
plt.color(28,242,221)
plt.pen("medium","dash")
plt.line(-5,5,5,-5,"")
plt.color(224,54,243)
plt.line(-5,-5,5,5,"")
plt.show_plot()
```



Press clear to display the Shell prompt

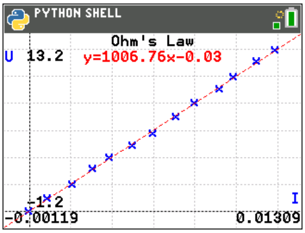
REGEQ1

Setup a regression equation prior to running the Python program in the Python App. An example would be to first, enter two lists in the CE OS. Then, for example, calculate [stat] CALC 4:LinReg(ax+b) for your lists. This stores the regression equation to RegEQ in the OS. Here is a program to recall RegEQ to the Python experience.

```
# Example of recall_RegEQ()
from ti_system import *

reg=recall_RegEQ()
print(reg)
x=float(input("Input x = "))
print("RegEQ(x) = ",eval(reg))
```

LINREGR (Provided in CE Bundle)

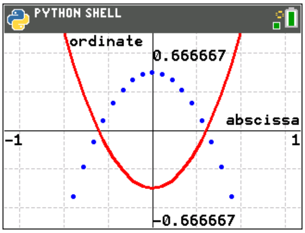
```
import ti_plotlib as plt

# current intensity
I = [0.0, 0.9, 2.1, 3.1, 3.9, 5.0, 6.0, 7.1, 8.0, 9.2, 9.9, 11.0,11.9]

# voltage
for n in range (len(I)):
I[n] /= 1000

# la tension
U = [0, 1, 2, 3.2, 4, 4.9, 5.8, 7, 8.1, 9.1, 10, 11.2, 12]

plt.cls()
plt.auto_window(I,U)
plt.pen("thin","solid")
plt.axes("on")
plt.grid(.002,2,"dot")
plt.title("Ohm's Law")
plt.color (0,0,255)
plt.labels("I","U",11,2)
plt.scatter(I,U,"x")
plt.color (255,0,0)
plt.pen("thin","dash")
plt.lin_reg(I,U,"center",2)
plt.show_plot()
plt.cls()
a=plt.a
b=plt.b
print ("a =",round(plt.a,2))
print ("b =",round(plt.b,2))
```



Press clear to display the Shell prompt

GRAPH (Provided in CE Bundle)

```
import ti_plotlib as plt
#After running the program, press [clear] to clear plot and return to
Shell.

def f(x):
••return 3*x**2-.4

def g(x):
••return -f(x)

def plot(res,xmin,xmax):
••#setup plotting area
••plt.window(xmin,xmax,xmin/1.5,xmax/1.5)
••plt.cls()
••gscale=5
••plt.grid((plt.xmax-plt.xmin)/gscale*(3/4),(plt.ymax-
plt.ymin)/gscale,"dash")
••plt.pen("thin","solid")
••plt.color(0,0,0)
••plt.axes("on")
••plt.labels("abscisse"," ordonnee",6,1)
••plt.pen("medium","solid")

# plot f(x) and g(x)
dX=(plt.xmax -plt.xmin)/res
x=plt.xmin
x0=x
••for i in range(res):
••••plt.color(255,0,0)
••••plt.line(x0,f(x0),x,f(x),"")
••••plt.color(0,0,255)
••••plt.plot(x,g(x),"o")
••••x0=x
••••x+=dX
••plt.show_plot()

#plot(resolution,xmin,xmax)
plot(30,-1,1)
# Create a graph with parameters(resolution,xmin,xmax)
# After clearing the first graph, press the [var] key. The plot()
function allows you to change the display settings
(resolution,xmin,xmax).
```

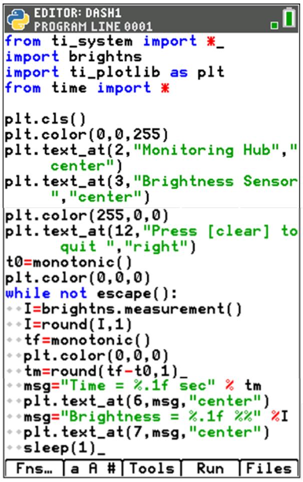


Press clear to display the Shell prompt

DASH1 – Sample TI-Innovator™ Hub Program

See: [Fns…]>Modul: ti_hub module

```
from ti_system import *
import brightns
import ti_plotlib as plt
from time import *

plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","center")
plt.text_at(3,"Brightness Sensor","center")
plt.color(255,0,0)
plt.text_at(12,"Press [clear] to quit ","right")
t0=monotonic()
plt.color(0,0,0)
while not escape():
••I=brightns.measurement()
••I=round(I,1)
••tf=monotonic()
••plt.color(0,0,0)
••tm=round(tf-t0,1)
••msg="Time = %.1f sec" % tm
••plt.text_at(6,msg,"center")
••msg="Brightness = %.1f %%" %I
••plt.text_at(7,msg,"center")
••sleep(1)
```

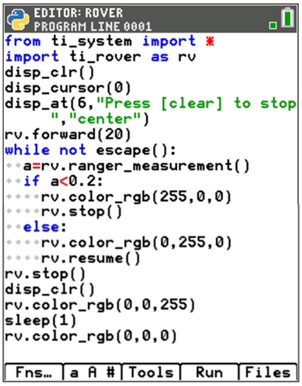


```
from ti_system import *_
import brightns
import ti_plotlib as plt
from time import *

plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","
     center")
plt.text_at(3,"Brightness Sensor
     ","center")
plt.color(255,0,0)
plt.text_at(12,"Press [clear] to
      quit ","right")
t0=monotonic()
plt.color(0,0,0)
while not escape():
••I=brightns.measurement()
••I=round(I,1)
••tf=monotonic()
••plt.color(0,0,0)
••tm=round(tf-t0,1)_
••msg="Time = %.1f sec" % tm
••plt.text_at(6,msg,"center")
••msg="Brightness = %.1f %%" %I
••plt.text_at(7,msg,"center")
••sleep(1)_
```

ROVER – Sample TI-Innovator™ Rover program

See: [Fns…]>Modul ti_rover module

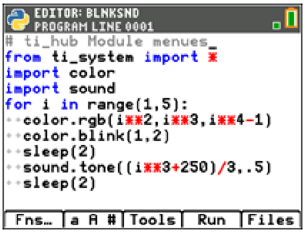
```
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"Press [clear] to stop","center")
rv.forward(20)
while not escape():
••a=rv.ranger_measurement()
••if a<0.2:
••••rv.color_rgb(255,0,0)
••••rv.stop()
••else:
••••rv.color_rgb(0,255,0)
••••rv.resume()
rv.stop()
disp_clr()
rv.color_rgb(0,0,255)
sleep(1)
rv.color_rgb(0,0,0)
```



```
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"Press [clear] to stop
     ","center")
rv.forward(20)
while not escape():
••a=rv.ranger_measurement()
••if a<0.2:
••••rv.color_rgb(255,0,0)
••••rv.stop()
••else:
••••rv.color_rgb(0,255,0)
••••rv.resume()
rv.stop()
disp_clr()
rv.color_rgb(0,0,255)
sleep(1)
rv.color_rgb(0,0,0)
```

BLNKSND - Sample TI-Innovator™ Hub Program

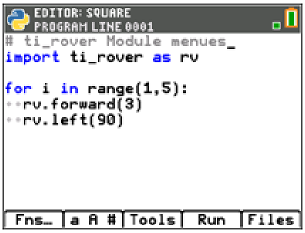
See: [Fns…]>Modul: ti_hub module



```
# ti_hub Module menues
from ti_system import *
import color
import sound
for i in range(1,5):
  color.rgb(i**2,i**3,i**4-1)
  color.blink(1,2)
  sleep(2)
  sound.tone((i**3+250)/3,.5)
  sleep(2)
```

SQUARE - Sample TI-Innovator™ Rover Program

See: [Fns…]>Modul ti_rover module

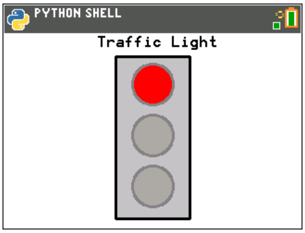


```
# ti_rover Module menues_
import ti_rover as rv

for i in range(1,5):
  rv.forward(3)
  rv.left(90)
```

STOP_GO - Sample ti_draw, ti_image, time Program

See: [Fns…]>Modul [Add-On]

```
from ti_draw import *
from ti_image import *
from time import *
clear()
# Pixel screen upper left (0,0) to (319,209)
draw_text(100,20,"Traffic Light")
set_pen("medium","solid")

draw_rect(120,25,80,175)
set_color(192,192,192)
fill_rect(120,25,80,175)
set_color(128,128,128)
draw_circle(160,55,22)
draw_circle(160,110,22)
draw_circle(160,165,22)

def off(x,y):
••set_color(169,169,169)
••fill_circle(x,y,22)
••set_color(128,128,128)
••draw_circle(x,y,22)

for i in (1,20,1):
# Green
••set_color(51,165,50)
••fill_circle(160,165,22)
••sleep(3)
••off(160,165)
# Yellow
••set_color(247,239,10)
••fill_circle(160,110,22)
••sleep(2)
••off(160,110)
# Red
••set_color(255,0,0)
••fill_circle(160,55,22)
••sleep(3)
••off(160,55)
••show_draw()
```



Traffic Light

Reference Guide for TI-Python Experience

The Python App contains menus of functions, classes, controls, operators and keywords for quick pasting in the Editor or Shell. The following reference table contains the listing of features in [2nd] [catalog] when the App is running. For a complete listing of Python functions, classes, operators, and keywords available in this version, please see "Selected TI-Python Built-in, Keywords, and Module Content."

This table is not intended to be an exhaustive list of Python available in this offering. Other functions supported in this Python offering can be entered using the alpha keys from the keypad.

Most examples given in this table run at the Shell prompt (>>>).

CATALOG Listing

Alphabetical List

- A
- B
- C
- D
- E
- F
- G
- H
- I
- L
- M
- N
- O
- P
- R
- S
- T
- U
- W
- X
- Y
- Symbols

A

#

Delimiter 2nd [catalog]

Syntax: #Your comment about your program.

Description: In Python, a comment begins with the hash tag character, #, and extends to the end of the line. [a A #]

Example:

```
#A short explanation of the code.
```

%

Operator 2nd [catalog]

Syntax: x%y or x % y

Description: Returns remainder of x/y. Preferred use is when x and y are integers. [a A #]

Example:

```
>>>57%2
1
```

See also fmod(x,y).

//

Operator 2nd [catalog]

Syntax: x//y or x // y

Description: Returns the floor division of x/y. [a A #]

Example:

```
>>>26//7
3
>>>65.4//3
21.0
```

[a A #]

Description: Launch [a A #] character palette.

Includes accented characters such as ç à â è é ê ë î ï ô ö ù û

[a A #] shortcut is on screen at [window] in the Editor or Shell

a gradient; slope

Module: ti_plotlib

[2nd] [catalog]

Syntax: plt.a gradient; slope

[Fns...]>Modul or [math] 5:ti_plotlib...> Properties 5:a

Description: After plt.linreg() is last executed in a program, the computed values of slope, a, and intercept , b, are stored in plt.a and plt.b.

Default values: = 0.0

Example:

See sample program: LINREGR.

import commands can be found in [2nd] [catalog] or in the ti_plotlib Setup menu.

abs()

Module: Built-in

[2nd] [catalog]

Syntax: abs(x)

Description: Returns the absolute value of a number. In this release, the argument may be an integer or floating point number.

Note: fabs() is a function in the math module.

Example:

```
>>>abs(-35.4)
35.4
```

acos()

Module: math

sin 7:acos()

Syntax: acos(x)

Description: Returns arc cosine of x in radians.

2nd [catalog]

Example:

```
>>>from math import *
>>>acos(1)
0.0
```

[Fns…] Modul 1:math… > Trig 7:acos()

Alternate Example: [Tools] > 6:New Shell

```
>>>import math
>>>math.acos(1)
0.0
```

import commands can be found in 2nd [catalog]

and

Keyword

2nd [test] Ops 8:and

Syntax: x and y

Description: May return True or False. Returns “x” if “x” is False and “y” otherwise. Pastes with space before and after and. Edit as needed.

[Fns…] > Ops 8:and

Example:

```
>>>2<5 and 5<10
True
>>>2<5 and 15<10
False
>>>{1} and 3
3
>>>0 and 5 < 10
0
```

2nd [catalog]

[a A #]

.append(x)

Module: Built-in

Syntax: listname.append(item)

Description: The method append() appends an item to a list.

Example:

```
>>>listA = [2,4,6,8]
>>>listA.append(10)
>>>print(listA)
[2,4,6,8,10]
```

2nd [list]
List
6: .append(x)

2nd [catalog]

[Fns…] > List
6:.append(x)

as

Keyword

2nd [catalog]

Description: Use as to create an alias when importing a module. See Python documentation for more details.

asin()

Module: math

Syntax: asin()

Description: Returns arc sine of x in radians.

Example:

```
>>>from math import *
>>>asin(1)
1.570796326794897
```

Alternate Example:

```
>>>import math
>>>math.asin(1)
1.570796326794897
```

sin 6:asin()

2nd [catalog]

[Fns…] >
Modul
1:math... >
Trig
6:asin()

import commands can be found in
2nd [catalog]

assert

Keyword 2nd [catalog]

Description: Use assert to test a condition in your code. Returns None or if not, execution of the program will display an AssertionError.

atan()

Module: math

sin 8:atan()

Syntax: atan(x)

Description: Returns arc tangent of x in radians.

[Fns…]>Modul 1:math... > Trig 8 :atan()

Example:

```
>>>from math import *
>>>atan(1)*4
3.141592653589793
```

2nd [catalog]

Alternate Example:

```
>>>import math
>>>math.atan(1)*4
3.141592653589793
```

import commands can be found in 2nd [catalog]

atan2(y,x)

Module: math

sin 9:atan2()

Syntax: atan2(y,x)

Description: Returns arc tangent of y/x in radians. Result is in [-pi, pi].

[Fns…] > Modul 1:math... > Trig 9:atan2()

Example:

```
>>>from math import *
>>>atan2(pi,2)
1.003884821853887
```

2nd [catalog]

Alternate Example:

```
>>>import math
>>>math.atan2(math.pi,2)
1.003884821853887
```

import commands can be found in 2nd [catalog]

auto_window(xlist,ylist)

Module: ti_plotlib

Syntax: plt.auto_window(xlist,ylist)

Description: Autoscales the plotting window to fit the data ranges within xlist and ylist specified in the program prior to the auto_window().

Note: max(list) - min(list) > 0.00001

Example:

See sample program: LINREGR.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Setup 5:auto_window ()

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

axes("mode")

Module: ti_plotlib

Syntax: plt.axes("mode")

Description: Displays axes on specified window in the plotting area.

Argument:

"mode" argument options:

"off"	no axes
"on"	axes+labels
"axes"	axes only
"window"	window labels only

plt.axes() uses the current pen color setting. To ensure plt.axes() are always drawn as expected, use plt.color() BEFORE plt.axes() to ensure the colors are expected.

Example:

See sample program LINREGR.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Setup 6:axes()

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

b y= intercept

Module: ti_plotlib

2nd [catalog]

Syntax: plt.b y= intercept

[Fns...]>Modul or math 5:ti_plotlib...> Properties 6:b

Description: After plt.linreg() is executed in a program, the computed values of slope, a, and intercept , b, are stored in plt.a and plt.b.

Default values: = 0.0

Example:

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

See sample program LINREGR.

bin(integer)

Module: Built-in

2nd [catalog]

Syntax: bin(integer)

Description: Displays binary format of the integer argument.

See Python documentation for more details.

Example:

```
>>> bin(2)
'0b10'
>>> bin(4)
'0b100'
```

break

Keyword

2nd [catalog]

Description: Use break to break out of a for or while loop.

C

ceil()

Module: math

Syntax: ceil(x)

Description: Returns the smallest integer greater than or equal to x.

Example:

```
>>>from math import *
>>>ceil(34.46)
35
>>>ceil(678)
678
```

math Modul
1:math... Math
8:ceil()

2nd [catalog]

[Fns...] Modul
1:math...Math
8:ceil()

import commands can be found in 2nd [catalog]

choice(sequence)

Module: random

Syntax: choice(sequence)

Description: Returns a random element from a non-empty sequence.

Example:

```
>>>from random import *
>>>listA=[2,4,6,8]
>>>choice(listA)       #Your result may differ.
4
```

math Modul
2:random...
Random
5:choice(sequence)

2nd [catalog]

[Fns...] Modul
2:random...
Random
5:choice(sequence)

import commands can be found in
2nd [catalog]

chr(integer)

Module: Built-in

2nd [catalog]

Syntax: chr(integer)

Description: Returns a string from an integer input representing the unicode character.

See Python documentation for more details.

Example:

```
>>> char(40)
'('
>>> char(35)
'#'
```

class

Keyword

2nd [catalog]

Description: Use class to create a class. See Python documentation for more details.

cls()   clear screen

Module: ti_plotlib

2nd [catalog]

Syntax: plt.cls()   clear screen

[Fns...]>Modul or math 5:ti_plotlib...> Setup 2:cls()

Description: Clears Shell screen for the plotting. Shortcut keys are not in display when plotting.

[Fns...]>Modul or math 5:ti_plotlib...> Draw 2:cls()

Note:plt.cls() has a different behavior than ti_system module disp_clr().

Example:

See sample program: GRAPH.

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

color(r,g,b)   0-255

Module: ti_plotlib

2nd [catalog]

Syntax: plt.color(r,g,b)   0-255

[Fns...]>Modul or math 5:ti_plotlib...> Draw 1:color()

Description: Sets the color for all following graphics/plotting. (r,g,b) values must be specified 0-255. Color specified is used in plot display until color() is again executed with a different color.

Default color is black upon importing ti_plotlib.

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

Example:

See sample program: COLORLIN.

complex(real,imag)

Module: Built-in

2nd [catalog]

Syntax: complex(real,imag)

[Fns...]>Type> 5:complex()

Description: Complex number type.

Example:

```
>>>z = complex(2, -3)
>>>print(z)
(2-3j)
>>>z = complex(1)
>>>print(z)
(1+0j)
>>>z = complex()
>>>print(z)
0j
>>>z = complex("5-9j")
>>>print(z)
(5-9j)
```

Note:"1+2j" is correct syntax. Spaces such as "1 + 2j" will display an Exception.

continue

Keyword 2nd [catalog]

Description: Use continue in a for or while loop to end the current iteration. See Python documentation for more details.

cos()

Module: math

sin Trig
4: cos()

Syntax: cos(x)

Description: Returns cos of x. Angle argument is in radians.

2nd [catalog]

Example:

```
>>>from math import *
>>>cos(0)
1.0
>>>cos(pi/2)
6.123233995736767e-17
```

[Fns...] Modul
1:math... > Trig
4:cos()

Alternate Example:

```
>>>import math
>>>math.cos(0)
1.0
```

Note: Python displays scientific notation using e or E. Some math results in Python will be different than in the CE OS.

.count()

Module: Built-in 2nd [catalog]

Syntax: listname.count(item)

Description: count()is a method that returns the number of occurrences of an item in a list, tuple, bytes, str, bytearray, or array.array object.

Example:

```
>>>listA = [2,4,2,6,2,8,2,10]
>>>listA.count(2)
4
```

def function():

Keyword 2nd [catalog]

Syntax: def function(var, var,...)

Description: Define a function dependent on specified variables. Typically used with the keyword return.

[Fns…]>Func
1:def function():

Example:

[Fns…]>Func
2:return

```
>>>   def f(a,b):
…         return a*b
…
…
…
>>>       f(2,3)
6
```

degrees()

Module: math

sin Trig
2:degrees()

Syntax: degrees(x)

Description: Converts angle x in radians to degrees.

2nd [catalog]

Example:

```
>>>from math import *
>>>degrees(pi)
180.0
>>>degrees(pi/2)
90.0
```

[Fns…]>Modul
1:math…>Trig
2:degrees()

del

Keyword 2nd [catalog]

Description: Use del to delete objects such as variables, lists, etc.
See Python documentation for more details.

disp_at(row,col,"text")

Module: ti_system

Syntax: disp_at(row,col,"text")

Description: Display text starting at a row and column position on the plotting area.

REPL with cursor >>>| will appear after text if at end of program. Use disp_cursor() to control cursor display.

Argument:

row	1 - 11, integer
column	1 - 32, integer
"text"	is a string which will wrap on the screen area

Optional arguments for color and background shown here: disp_at(row,col,"text","align",color 0-15, background color 0-5)

Example:

Sample program:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,6,"hello")
disp_cursor(0)
disp_wait()
```

[2nd] [catalog]

[2nd] [rcl]
ti_system
7:disp_at()

[Fns…]>Modul
or [math]
4:ti_system
7:disp_at()

import commands can be found in [2nd] [catalog] or in the ti_system Modul menu.

disp_at(row,"text","align")

Module: ti_system

Syntax: disp_at(row,"text","align")

Description: Display text aligned as specified on the plotting screen for row 1-11. Row is cleared before display. If used in a loop, content refreshes with each display.

REPL with cursor >>>| will appear after text if at end of program. Use disp_cursor() to control cursor display before the use of disp_at() in your program.

Argument:

row	1 - 11, integer
"text"	is a string which will wrap on the screen area
"align"	"left" (default) "center" "right"

Optional argument shown here: disp_at (row,"text","align","color 0-15, background color 0-15)

Example:

Sample program:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

[Fns…]>Modul
or math
4:ti_system
7:disp_at()

import commands can be found in 2nd [catalog] or in the ti_system Modul menu.

disp_clr() **clear text screen**

Module: ti_system

Syntax: disp_clr() clear text screen

Description: Clear the screen in the Shell environment. Row 0-11, integer may be used as an optional argument to clear a display row of the Shell environment.

Example:

Sample program:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
8:disp_clr()

[Fns…]>Modul
or math
4:ti_system
8:disp_clr()

import commands can be found in 2nd [catalog] or in the ti_system Modul menu.

disp_cursor() 0=off 1=on

Module: ti_system

Syntax: disp_cursor() 0=off 1=on

Description: Control the display of the cursor in the Shell when a program is running.

Argument:

0 = off

not 0 = on

Example:

Sample program:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
0:disp_cursor()

[Fns…]>Modul or
math
4:ti_system
0:disp_cursor()

import commands can be found in 2nd [catalog] or in the ti_system Modul menu.

disp_wait() [clear]

Module: ti_system

Syntax: disp_wait() [clear]

Description: Stop the execution of program at this point and display screen content until [clear] is pressed and the screen is cleared.

Example:

Sample program:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
9:disp_wait()

[Fns…]>Modul
or math
4:ti_system
9:disp_wait()

import commands can be found in 2nd [catalog] or in the ti_system Modul menu.

E

e

Module: math

Syntax: math.e or e if math module was imported

Description: Constant e displays as shown below.

Example:

```
>>>from math import *
>>>e
2.718281828459045
```

Alternate Example:

```
>>>import math
>>>math.e
2.718281828459045
```

[2nd] [e] (above [÷])

[Fns…] > Modul 1:math… > Const 1:e

elif :

Keyword

See if..elif..else.. for details.

[2nd] [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

else:

Keyword

2nd [catalog]

See if..elif..else.. for details.

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

escape()

Module: ti_system

2nd [catalog]

Syntax: escape()

As a program line:

Description: escape() returns True or False.

Initial value is False.

When the [clear] key on CE is pressed, the value is set to True.

When the function is executed the value is reset to False.

2nd [rcl]
ti_system
5:while not escape():
6:if escape ():break

[Fns…]>Modul or math
4:ti_system
5:while not escape():
6:if escape ():break

Example of use:

while not escape():

In a while loop running in a program where the program offers to end the loop but keep the script running.

if escape():break

Can be used to a debug program to inspect the vars using Shell [vars] after running the program and using this break.

import commands can be found in 2nde [catalog] or in the ti_system Modul menu.

eval()

Module: Built-in

2nd [catalog]

Syntax: eval(x)

Description: Returns the evaluation of the expression x.

[Fns…] I/O
3:eval()

Example:

```
>>>a=7
>>>eval(“a+9”)
16
>>>eval(‘a+10’)
17
```

except exception:

Keyword

2nd [catalog]

Description: Use except in a try..except code block.
See Python documentation for more details.

exp()

Module: math

Syntax: exp(x)

Description: Returns e**x.

Example:

```
>>>from math import *
>>>exp(1)
2.718281828459046
```

Alternate Example: [Tools] > 6:New Shell

```
>>>import math
 >>>math.exp(1)
2.718281828459046
```

2nd [e^x] (above ln)

2nd [catalog]

[Fns…] > Modul 1:math… 4:exp()

import commands can be found in 2nd [catalog].

.extend()

Module: Built-in

2nd [catalog]

Syntax: listname.extend(newlist)

Description: The method extend() is a method to extend newlist to the end of a list.

Example:

```
>>>listA = [2,4,6,8]
>>>listA.extend([10,12])
>>>print(listA)
[2,4,6,8,10,12]
```

fabs()

Module: math

Syntax: fabs(x)

Description: Returns the absolute value of x

Example:

```
>>>from math import *
>>>fabs(35-65.8)
30.8
```

2nd [catalog]

[Fns…] > Modul 1:math... 2:fabs()

import commands can be found in 2nd [catalog].

See also Built-in function abs().

False

Keyword

Description: Returns False when statement executed is False. "False" represents the false value of objects of type bool.

Example:

```
>>>64<=32
False
```

2nd [test] (above math)

2nd [catalog]

[Fns…] > Ops B:False

[a A #]

finally:

Keyword

2nd [catalog]

Description: Use finally in a try..except..finally code block. See Python documentation for more details.

float()

Module: Built-in

2nd [catalog]

Syntax: float(x)

Description: Returns x as a float.

[Fns…] > Type
2:float()

Example:

```
>>>float(35)
35.0
>>>float("1234")
1234.0
```

floor()

Module: math

math Modul
1:math
9:floor()

Syntax: floor(x)

Description: Returns the largest integer less than or equal to x.

2nd [catalog]

Example:

```
>>>from math import *
>>>floor(36.87)
36
>>>floor(-36.87)
-37
>>>floor(254)
254
```

[Fns…] > Modul
1:math
9:floor()

import commands can be found in 2nd [catalog]

fmod(x,y)

Module: math

Syntax: fmod(x,y)

Description: See Python documentation for more details. Preferred use is when x and y are floats.

May not return the same result as x%y.

Example:

```
>>>from math import *
>>>fmod(50.0,8.0)
2.0
>>>fmod(-50.0,8.0)
-2.0
>>>-50.0 - (-6.0)*8.0        #validation from description
-2.0
```

See also: x%y.

$\boxed{\text{math}}$ Modul
1:math
7:fmod()

$\boxed{\text{2nd}}$ [catalog]

[Fns…] > Modul
1:math…
7:fmod()

import commands can be found in $\boxed{\text{2nd}}$ [catalog]

for i in list:

Keyword

Syntax: for i in list:

Description: Used to iterate over list elements.

Example:

```
>>> for i in [2,4,6]:
…       print(i)
…
…
…
2
4
6
```

[Fns…] Ctl
7:for i in list:

$\boxed{\text{2nd}}$ [catalog]

for i in range(size):

Keyword

Syntax: for i in range(size)

Description: Used to iterate over a range.

Example:

```
>>> for i in range(3):
…       print(i)
…
…
…
0
1
2
```

[Fns…] Ctl
4:for i in range
(size):

2nd [catalog]

for i in range(start,stop):

Keyword

Syntax: for i in range(start,stop)

Description: Used to iterate over a range.

Example:

```
>>> for i in range(1,4):
…       print(i)
…
…
…
1
2
3
```

[Fns…] Ctl
5:for i in range
(start,stop):

2nd [catalog]

for i in range(start,stop,step):

Keyword

[Fns…] Ctl
6:for i in range
(start,stop,step):

Syntax: for i in range(start,stop,step)

Description: Used to iterate over a range.

Example:

2nd [catalog]

```
>>> for i in range(1,8,2):
…       print(i)
…
…
…
1
3
4
7
```

str.format()   string format

Module: Built-in

2nd [catalog]

Syntax:str.format()

Description: Formats the given string. See Python documentation for more details.

Example:

```
>>> print("{+f}".format(12.34))
+12.340000
```

frexp()

Module: math

Syntax: frexp(x)

Description: Returns a pair (y,n) where x == y * 2**n. y is float where 0.5<abs(y)<1; and n is integer.

Example:

```
>>>from math import *
>>>frexp(2000.0)
(0.9765625, 11)
>>>0.9765625 * 2**11      #validate description
2000.0
```

math Modul
1:math
A:frexp()

2nd [catalog]

[Fns…] > Modul
1:math
A:frexp()

import commands can be found in 2nd [catalog]

from PROGRAM import *

Keyword

Syntax: from PROGRAM import *

Description: Used to import a program. Imports the public attributes of a Python module into the current name space.

Shell [Tools]
A:from PROGRAM import *

2nd [catalog]

from math import *

Keyword

Syntax: from math import *

Description: Used to import all functions and constants from the math module.

math Modul
1:math…
1:from math import *

[Fns..] > Modul
1:math…
1:from math import *

2nd [catalog]

from random import *

Keyword

Syntax: from random import *

Description: Used to import all functions from the random module.

math Modul
2:random…
1:from random import *

[Fns..] > Modul
2:random…
1:from random import *

2nd [catalog]

from time import *

Keyword

Syntax: from time import *

Description: Used to import all methods from the time module.

Example:

See sample program: DASH1.

2nd [catalog]

math Modul
3:time…
1:from time import
*

[Fns…]>Modul
3:time…
1:from time import
*

from ti_system import *

Keyword

Syntax: from ti_system import *

Description: Used to import all methods from the ti_ system module.

Example:

See sample program: REGEQ1.

2nd [catalog]

math Modul
4:ti_system…
1:from system
import *

[Fns…]>Modul
4:ti_system…
1:from system
import *

from ti_hub import *

Keyword 2nd [catalog]

Syntax: from ti_hub import *

Description: Used to import all methods from the ti_ hub module. For individual input and output devices, use the dynamic module functionality by selecting the device from [Fns…]>Modul>ti_hub>Import menu when in the Editor.

See:ti_hub module – Add import to Editor and add ti_ hub sensor module to the Modul menu.

Example:

See sample program: DASH1.

global

Keyword

2nd [catalog]

Description: Use global to create global variables inside a function.

See CircuitPython documentation for more details.

grid(xscl,yscl,"style")

Module: ti_plotlib

2nd [catalog]

Syntax: plt.grid(xscl,yscl,"style")

[Fns...]>Modul or math 5:ti_plotlib...> Setup 3:grid()

Description: Displays a grid using specified scale for x and y axes. Note: All plotting takes place when plt.show_plot() is executed.

Setting grid color is the optional argument of (r,g,b) using values 0-255 with default value of gray (192,192,192).

Default value for xscl or yscl = 1.0.

"style" = "dot" (default), "dash", "solid" or "point"

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

Example:

See sample programs: COLORLIN or GRAPH.

grid(xscl,yscl,"style",(r,g,b))

Module: ti_plotlib

Syntax: plt.grid(xscl,yscl,"style",(r,g,b))

Description: Displays a grid using specified scale for x and y axes. Note: All plotting takes place when plt.show_plot() is executed.

Setting grid color is the optional argument of (r,g,b) using values 0-255 with default value of gray (192,192,192).

Default value for xscl or yscl = 1.0.

"style" = "dot" (default), "dash", "solid" or "point" .

If the xscl or yscl values are less than 1/50th of the difference between xmax-xmin or ymax-ymin, then an exception of 'Invalid grid scale value.'

Example:

See sample program: GRAPH.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Setup 3:grid()

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

H

hex(integer)

Module: Built-in 2nd [catalog]

Syntax: hex(integer)

Description: Displays hexadecimal format of the integer argument. See Python documentation for more details.

Example:

```
>>> hex(16)
'0x10'
>>> hex(16**2)
'0x100'
```

I

"if :"

See if..elif..else.. for details.

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

if..elif..else..

Keyword

2nd [catalog]

Syntax: ••Gray indent identifiers automatically provided in the Python App for ease of use.

if :

••

elif :

••

else:

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

Description: if..elif..else is a conditional statement. The Editor provides automatic indents as gray dots to assist your correct programming indents.

Example: Create and run this program, say S01, from the Editor

```
def f(a):
••if a>0:
••••print(a)
••elif a==0:
••••print("zero")
••else:
••••a=-a
••••print(a)
```

Shell interaction

```
>>> # Shell Reinitialized
>>> # Running S01
>>>from S01 import *      #automatically pastes
>>>f(5)
5
>>>f(0)
zero
>>>f(-5)
5
```

if..else..

Keyword 2nd [catalog]

See if..elif..else.. for details.

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

.imag

Module: Built-in 2nd [catalog]

Syntax:var.imag

Description: Returns the imaginary part of a specified variable of complex number type.

Example:

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

import math

Keyword

Syntax: import math 2nd [catalog]

Description: The math module is accessed using this command. This instruction imports the public attributes of the "math" module within its own namespace.

import random

Keyword

Syntax: import random

[2nd] [catalog]

Description: The random module is accessed using this command. This instruction imports the public attributes of the "random" module within its own namespace.

import ti_hub

Keyword

[2nd] [catalog]

Syntax: import ti_hub

Description: The ti_hub module is accessed using this command. This instruction imports the public attributes of the ti_hub module wihin its own namespace.

For individual input and output devices, use the dynamic module functionality by selecting the device from [Fns…]>Modul>ti_hub>Import menu when in the Editor.

See:[Fns…] > Modul: ti_hub module.

import time

Keyword

[2nd] [catalog]

Syntax: import time

Description: The time module is accessed using this command. This instruction imports the public attributes of the time module within its own name-space.

See:[Fns…] > Modul: time and ti_system modules.

import ti_plotlib as plt

Keyword

Syntax: import ti_plotlib as plt

Description: The ti_plotlib module is accessed using this command. This instruction imports the public attributes of the ti_plotlib module wihin its own namespace. Attributes of the ti_plotlib module must be entered as plt.attribute.

Example:

See sample program: COLORLIN.

[2nd] [catalog]

[math] Modul
5:ti_plotlib...
1:import ti_plotlib as plt

[Fns…]>Modul
5:ti_plotlib...
1:import ti_plotlib as plt

import ti_rover as rv

Keyword

Syntax: import ti_rover as rv

Description: The ti_rover module is accessed using this command. This instruction imports the public attributes of the ti_rover module within its own name-space. Attributes of the ti_rover module must be entered as rv.attribute.

Example:

See sample program: ROVER.

[2nd] [catalog]

[math] Modul
7:ti_rover...
1:import ti_rover as rv

[Fns…]>Modul
7:ti_rover...
1:import ti_rover as rv

import ti_system

Keyword 2nd [catalog]

Syntax: import ti_system

Description: The ti_system module is accessed using this command. This instruction imports the public attributes of the ti_system module within its own name-space.

Example:

See sample program: REGEQ1.

in

Keyword 2nd [catalog]

Description: Use in to check if a value is in a sequence or to iterate a sequence in a for loop.

.index(x)

Module: Built-in 2nd [catalog]

Syntax:var.index(x)

Description: Returns the index or position of an element of a list. See Python documentation for more details.

Example:

```
>>> a=[12,35,45]
>>> print(a.index(12))
0
>>> print(a.index(35))
1
>>> print(a.index(45))
2
```

input()

Module : Built-in 2nd [catalog]

Syntax: input()

input()

Description: Prompt for input

[Fns…] I/O
2:input()

Example:

```
>>>input("Name? ")
Name? Me
'Me'
```

Alternate Example:

```
Create Program A
len=float(input("len: "))
print(len)
```

```
Run Program A
>>> # Shell Reinitialized
>>> # Running A
>>>from A import *
len: 15        (enter 15)
15.0          (output float 15.0)
```

.insert(index,x)

Module : Built-in

Syntax: listname.insert(index,x)

Description: The method insert() inserts an item x after index within a sequence.

Example:

```
>>>listA = [2,4,6,8]
>>>listA.insert(3,15)
>>>print(listA)
[2,4,6,15,8]
```

2nd [list] List
8:.insert(index,x)

2nd [catalog]

[Fns…] > List
8:.insert(index,x)

int()

Module : Built-in

Syntax: int(x)

Description: Returns x as an integer object.

Example:

```
>>>int(34.67)
34
>>>int(1234.56)
1234
```

2nd [catalog]

[Fns…] > Type
1:int()

is

Keyword

Description: Use is to test if two objects are the same object.

2nd [catalog]

labels("xlabel","ylabel",x,y)

Module: ti_plotlib

2nd [catalog]

Syntax: plt.labels("xlabel","ylabel",x,y)

[Fns…]>Modul or math 5:ti_plotlib...> Setup 7:labels()

Description: Displays "xlabel" and "ylabel" labels on the plot axes at row positions x and y. Adjust as needed for your plot display.

"xlabel" is positioned on specified row x (default row 12) and is right justified.

"ylabel" is positioned on specified row y (default row 2) and is left justified.

Note: plt.labels("|","",12,2) will paste with x and y row defaults, 12,2 , which then can be modified for your program.

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

Example:

See sample program: GRAPH.

lambda

Keyword

2nd [catalog]

Syntax: lambda arguments : expression

Description: Use lambda to define an anonymous function. See Python documentation for details.

len()

Module: Built-in

Syntax: len(sequence)

Description: Returns the number of items in the argument. The argument may be a sequence or a collection.

See Python documentation for more details.

Example:

```
>>>mylist=[2,4,6,8,10]
>>>len(mylist)
5
```

2nd [list] (above stat) List
3:len()

2nd [catalog]

[Fns…] > List
3:len()

line(x1,y1,x2,y2,"mode")

Module: ti_plotlib

Syntax: plt.line(x1,y1,x2,y2,"mode")

Description: Displays a line segment from (x1,y1) to (x2,y2)

Size and style are set using pen() and color() before line().

Arguments:

x1,y1, x2,y2 are real floats.

"mode": When default "", no arrowhead draws. When "arrow" a vector arrowhead at (x2,y2) draws.

Example:

See sample program: COLORLIN.

2nd [catalog]

[Fns…]>Modul or math
5:ti_plotlib...> Draw
7:line or vector

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

lin_reg(xlist,ylist,"disp",row)

Module: ti_plotlib

Syntax: plt.lin_reg(xlist,ylist,"disp",row)

Description: Calculates and draws the linear regression model, ax+b, of xlist,ylist. This method must follow the scatter method. Default display of equation is "center" at row 11.

Argument:

"disp"	"left" "center" "right"
row	1 - 12

plt.a (slope) and plt.b (intercept) are stored when lin_reg executes.

Example:

See sample program: LINREGR.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Draw 8:lin_reg()

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

list(sequence)

Module: Built-in

Syntax: list(sequence)

Description: Mutable sequence of items of the save type.

list()" converts its argument into the "list" type. Like many other sequences, the elements of a list do not need to be of the same type.

Example:

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
```

Example:

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
>>> list({1,2,"c", 7})
[7, 1, 2, 'c']
>>> list("foobar")
['f', 'o', 'o', 'b', 'a', 'r']
```

2nd [list] (above stat) List
2:list(sequence)

2nd [catalog]

[Fns…] > List
2:list(sequence)

log(x,base)

Module: math

Syntax: log(x,base)

Description: log(x) with no base returns the natural logarithm x.

Example:

```
 >>>from math import *
>>>log(e)
1.0
>>>log(100,10)
2.0
>>>log(32,2)
5.0
```

[2nd] [log] for log (x,10)

[2nd] [ln] for log (x) (natural log)

[math] Modul
1:math...
6:log(x,base)

[2nd] [catalog]

[Fns...] > Modul
1:math...
6:log(x,base)

import commands can be found in [2nd] [catalog]

math.function

Module: math

[2nd] [catalog]

Syntax: math.function

Description: Use after import math command to use a function in the math module.

Example:

```
>>>import math
>>>math.cos(0)
1.0
```

max()

Module: Built-in

[2nd] [list] (above [stat]) List 4:max()

Syntax: max(sequence)

Description: Returns the maximum value in the sequence. See Python documentation for more information on max().

[2nd] [catalog]

Example:

```
>>>listA=[15,2,30,12,8]
>>>max(listA)
30
```

[Fns…] > List 4:max()

min()

Module: Built-in

[2nd] [list] (above [stat]) List 5:min()

Syntax: min(sequence)

Description: Returns the minimum value in the sequence. See Python documentation for more information on min().

[2nd] [catalog]

Example:

```
>>>listA=[15,2,30,12,8]
>>>min(listA)
2
```

[Fns…] > List 5:min()

monotonic() **elapsed time**

Module: time

2nd [catalog]

Syntax: monotonic() elapsed time

Description: Returns a value of time from the point of execution. Use the return value to compare against other values from monotonic().

[Fns…]>Modul or math 3:time 3:momotonic()

Example:

Sample program:

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

```
Run the program EXAMPLE until execution stops.
>>>15.0
```

import commands can be found in 2nd [catalog] or in the time Modul menu.

N

None

Keyword — 2nd [catalog]

Description: None represents the absence of a value.

Example: — [a A #]

```
>>> def f(x):
…       x
…
…
…
>>> print(f(2))
None
```

nonlocal

Keyword — 2nd [catalog]

Syntax: nonlocal

Description: Use nonlocal to declare a variable is not local. See Python documentation for more details.

not

Keyword — 2nd [test] Ops
0:not

Syntax: not x

Description: Evaluates to True if x is False and False otherwise. Pastes with space before and after the keyword not. Edit as needed.

[Fns…] > Ops
0:not

Example:

```
>>> not 2<5       #edit the space before not
False
>>>3<8 and not 2<5
False
```

2nd [catalog]

[a A #]

O

oct(integer)

Module: Built-in

[2nd] [catalog]

Syntax: oct(integer)

Description: Returns the octal representation of the integer. See Python documentation for more details.

Example:

```
>>> oct(8)
'0o10'
>>> oct(64)
'0o100'
```

or

Keyword

[2nd] [test] Ops 9:or

[Fns…] > Ops 9:or

[2nd] [catalog]

[a A #]

Syntax: x or y

Description: May return True or False. Returns x if x evaluates as True and y otherwise. Pastes with space before and after or. Edit as needed.

Example:

```
>>>2<5 or 5<10
True
>>>2<5 or 15<10
True
>>>12<5 or 15<10
False
>>> 3 or {}
3
>>> [] or {2}
{2}
```

ord("character")

Module: Built-in 2nd [catalog]

Syntax: ord("character")

Description: Returns the unicode value of the character. See Python documentation for more details.

Example:

```
>>> ord("#")
35
>>> ord("/")
47
```

pass

Keyword

2nd [catalog]

Description: Use pass in an empty function or class definition as a placeholder for future code as you build out your program. Empty definitions will not cause an error when program is executed.

pen("size","style")

Module: ti_plotlib

2nd [catalog]

Syntax: plt.pen("size","style")

[Fns…]>Modul or math 5:ti_plotlib...> Draw 9:pen()

Description: Sets the appearance of all following lines until the next pen() is executed.

Argument:

Default pen() is "thin" and "solid."

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

"size"	"thin"
	"medium"
	"thick"
"style"	"solid"
	"dot"
	"dash"

Example:

See sample programs: COLORLIN or GRAPH.

pi

Module: math

$\boxed{\text{2nd}}$ [π] (above $\boxed{\text{sin}}$)

Syntax: math.pi or pi if math module imported.

Description: Constant pi displays as shown below.

[Fns…] > Modul 1:math… > Const 2:pi

Example:

```
>>>from math import *
>>>pi
3.141592653589793
```

Alternate Example:

```
>>>import math
>>>math.pi
3.141592653589793
```

plot(xlist,ylist,"mark")

Module: ti_plotlib

Syntax: plt.plot(xlist,ylist,"mark")

Description: A line plot displays using ordered pairs from specified xlist and ylist. The line style and size are set using plt.pen().

xlist and ylist must be real floats and lists must bee the same dimension.

Argument:

"mark" is the mark character as follows:

o	filled dot (default)
+	cross
x	x
.	pixel

Example:

See sample program: LINREGR.

[2nd] [catalog]

[Fns...]>Modul or [math] 5:ti_plotlib...> Draw 5:Connected Plot with Lists

import commands can be found in [2nd] [catalog] or in the ti_plotlib Setup menu.

plot(x,y,"mark")

Module: ti_plotlib

Syntax: plt.plot(x,y,"mark")

Description: A point plot, (x,y) displays using specified x and y.

xlist and ylist must be real floats and lists must be the same dimension.

Argument:

"mark" is the mark character as follows:

o	filled dot (default)
+	cross
x	x
.	pixel

Example:

See sample program: LINREGR.

$\boxed{\text{2nd}}$ [catalog]

[Fns...]>Modul or $\boxed{\text{math}}$ 5:ti_plotlib...> Draw 6:plot a Point

import commands can be found in $\boxed{\text{2nd}}$ [catalog] or in the ti_plotlib Setup menu.

pow(x,y)

Module: math

Syntax: pow(x,y)

Description: Returns x raised to the power y. Converts both x and y to float. See Python documentation for more information.

Use the built-in pow(x,y) function or ** for computing exact integer powers.

Example:

```
>>>from math import *
>>>pow(2,3)
>>>8.0
```

Example using: Built-in:

[Tools] > 6:New Shell

```
>>>pow(2,3)
8
>>>2**3
8
```

[math] Modul
1:math
5:pow(x,y)

[2nd] [catalog]

[Fns…] > Modul
1:math
5:pow(x,y)

import commands can be found in [2nd] [catalog]

print()

Module: Built-in

Syntax: print(argument)

Description: Displays argument as string.

Example:

```
>>>x=57.4
>>>print("my number is =", x)
my number is= 57.4
```

[2nd] [catalog]

[Fns…] > I/O
1:print()

radians()) degree ▶radians

Module: math

Syntax: radians(x)

Description: Converts angle x in degrees to radians.

Example:

```
>>>from math import *
>>>radians(180.0)
3.141592653589793
>>>radians(90.0)
1.570796326794897
```

sin Trig
1:radians()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig
1:radians()

raise

Keyword

2nd [catalog]

Syntax: raise exception

Description: Use raise to raise a specified exception and stop your program.

randint(min,max)

Module: random

Syntax: randint(min,max)

Description: Returns a random integer between min and max.

Example:

```
>>>from random import *
>>>randint(10,20)
>>>15
```

Alternate Example:

```
>>>import random
>>>random.randint(200,450)
306
```

Results will vary given a random output.

[math] Modul
2:random
4:randint
(min,max)

[Fns…] > Modul
2:random…
4:randint
(min,max)

[2nd] [catalog]

import commands can be found in [2nd] [catalog]

random()

Module: random

Syntax: random()

Description: Returns a floating point number from 0 to 1.0. This function takes no arguments.

Example:

```
>>>from random import *
>>>random()
0.5381466990230621
```

Alternate Example:

```
>>>import random
>>>random.random()
0.2695098437037318
```

Results will vary given a random output.

[math] Modul
2:random…
Random
2:random()

[Fns…] > Modul
2:random…
Random
2:random()

[2nd] [catalog]

import commands can be found in [2nd] [catalog]

random.function

Module: random

Syntax: random.function

Description: Use after import random to access a function in the random module.

Example:

```
>>>import random
>>>random.randint(1,15)
2
```

Results will vary given a random output.

[2nd] [catalog]

randrange(start,stop,step)

Module: random

Syntax: randrange(start,stop,step)

Description: Returns a random number from start to stop by step.

Example:

```
>>>from random import *
>>>randrange(10,50,2)
12
```

Alternate Example:

```
>>>import random
>>>random.randrange(10,50,2)
48
```

Results will vary given a random output.

math Modul
2:random…
Random
6:randrange
(start,stop,step)

math Modul
2:random…
Random
6:randrange
(start,stop,step)

2nd [catalog]

import commands can be found in 2nd [catalog]

range(start,stop,step)

Module: Built in

2nd [catalog]

Syntax: range(start,stop,step)

Description: Use range function to return a sequence of numbers. All arguments are optional. Start default is 0, step default is 1 and sequence ends at stop.

Example:

```
>>> x = range(2,10,3)
>>> for i in x
…       print(i)
…
…
2
5
8
```

.real

Module: Built-in 2nd [catalog]

Syntax:var.real

Description: Returns the real part of a specified variable of complex number type.

Example:

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

var=recall_list("name") 1-6

Module: ti_system

Syntax:var=recall_list("name") 1-6

Description: Recall a predefined OS list. List length must be less than or equal to 100.

Argument: "**name**"

For OS L1-L6

1 - 6

"1" - "6"

'1' - '6'

For OS custom list "name"

----- Max 5 characters, numbers or letters, starting with letters, and letters must be uppercase.

Examples:

"ABCDE"

"R12"

"L1" will be custom L1 and not OS L1

Reminder: Python is double precision. Python supports more digits than in the OS.

Example:

Sample program:

Create a list in the OS.
LIST={1,2,3}

Run Python App.
Create a new program AA.

```
import ti_system as *
xlist=recall_list("LIST")
print xlist
```

Run program AA.
Shell displays output.

```
[1.0, 2.0, 3.0]
```

[2nd] [catalog]

[2nd][rcl]
ti_system
4:var=recall_
list()

[Fns…]>Modul
or [math]
4:ti_system
4:var=recall_
list()

import commands can be found in [2nd] [catalog] or in the ti_system Modul menu.

var=recall_RegEQ()

Module: ti_system

Syntax:var=recall_RegEQ()

Description: Recall the RegEQ variable from the CE OS. The regression equation must be computed in the OS prior to recalling RegEQ in the Python App.

Example:

See sample program: REGEQ1.

[2nd] [catalog]

[2nd][rcl]
ti_system
4:var=recall_
REGEQ()

[Fns…]>Modul
or [math]
4:ti_system
4:var=recall_
REGEQ()

import commands can be found in [2nd] [catalog] or in the ti_system Modul menu.

.remove(x)

Module: Built-in

Syntax: listname.remove(item)

Description: The method remove() removes the first instance of item from a sequence.

Example:

```
>>>listA = [2,4,6,8,6]
>>>listA.remove(6)
>>>print(listA)
[2,4,8,6]
```

[2nd] [list]
List
7:.remove(x)

[2nd] [catalog]

[Fns…] > List
7:.remove(x)

return

Module: Built-in

2nd [catalog]

Syntax: return expression

Description: A return statement defines the value produced by a function. Python functions return None by default. See also: def function():

[Fns…] > Func
1:def function():

Example:

[Fns…] > Func
2:return

```
>>>  def f(a,b):
…        return a*b
…
…
…
>>>      f(2,3)
6
```

.reverse()

Module: Built-in

2nd [catalog]

Syntax: listname.reverse()

Description: Reverses the order of items in a sequence.

Example:

```
>>>list1=[15,-32,4]
>>>list1.reverse()
>>>print(list1)
[4,-32,15]
```

round()

Module: Built in

2nd [catalog]

Syntax: round(number, digits)

Description: Use round function to return a floating point number rounded to the specified digits. Default digit is 0 and returns the nearest integer.

Example:

```
>>>round(23.12456)
23
>>>round(23.12456,3)
23.125
```

scatter(xlist,ylist,"mark")

Module: ti_plotlib

Syntax: plt.scatter(xlist,ylist,"mark")

Description: A sequence of ordered pair from (xlist,ylist) will be plotted with mark style specified. The line style and size are set using plt.pen().

xlist and ylist must be real floats and lists must bee the same dimension.

Argument:

"mark" is the mark character as follows:

o	filled dot (default)
+	cross
x	x
.	pixel

Example:

See sample program: LINREGR.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Draw 4:scatter()

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

seed()

Module: random

Syntax: seed() or seed(x) where x is integer

Description: Initialize random number generator.

Example:

```
>>>from random import *
>>>seed(12)
>>>random()
0.9079708720366826
>>>seed(10)
>>>random()
0.9063990882481896
>>>seed(12)
>>>random()
0.9079708720366826
```

Results will vary given a random output.

math Modul 2:random… Random 7:seed()

[Fns…] > Modul 2:random… Random 7:seed()

2nd [catalog]

import commands can be found in 2nd [catalog]

set(sequence)

Module: Built-in

2nd [catalog]

Syntax: set(sequence)

Description: Returns a sequence as a set. See Python documentation for more details.

Example:

```
>>> print(set("84CE")
{'E', '8', '4', 'C'}
```

show_plot()   display > [clear]

Module: ti_plotlib

Syntax: plt.show_plot()   display > [clear]

Description: Executes the display of the plot as set up in the program.

show_plot() must be placed after all plotting setup objects. The program order of plotting objects are suggested by the Setup menu ordering.

For plotting template help, from File Manager, select [New] ([zoom]) and then [Types] ([zoom]) to select the "Plotting (x,y) & Text" program type.

After running the program, the plotting display is cleared by pressing [clear] to return to the Shell prompt.

Example:

See sample programs: COLORLIN or GRAPH.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Setup 9:show_plot

[Fns…]>Modul or [math] 5:ti_plotlib… > Draw 9:show_plot()

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

sin()

Module: math

Syntax: sin()

Description: Returns sine of x. Argument angle is in radians.

Example:

```
>>>from math import *
>>>sin(pi/2)
1.0
```

sin 3:sin()

2nd [catalog]

[Fns…] > Modul 1:math… > Trig 3:sin()

import commands can be found in 2nd [catalog]

sleep(seconds)

Module: ti_system; time

Syntax: sleep(seconds)

Description: Sleep for a given number of seconds. Seconds argument is a float.

Example:

Sample program:

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

Run the program TIME
```
>>>15.0
```

2nd [catalog]

2nd [rcl] ti_system A:sleep()

[Fns…]>Modul or math 4:ti_system A:sleep()

[Fns…]>Modul or math 3:time 2:sleep()

import commands can be found in 2nd [catalog] or in the ti_system Modul menu.

.sort()

Module: Built-in

Syntax: listname.sort()

Description: The method sorts a list in place. See Python documentation for more details.

Example:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>listA.sort()
>>>print(listA)       #listA updated to a sorted list
[2,3,3,4,4,4,5,6,6,7,8,9]
```

2nd [list]
(above stat
List A:.sort()

2nd [catalog]

[Fns…] >
List
A:sort()

sorted()

Module: Built-in

Syntax: sorted(sequence)

Description: Returns a sorted list from sequence.

Example:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>sorted(listA)
[2,3,3,4,4,4,5,6,6,7,8,9]
>>>print(listA)       #listA did not change
[4,3,6,2,7,4,8,9,3,5,4,6]
```

2nd [list] (above
stat) List
0:sorted()

2nd [catalog]

[Fns…] > List
0:sorted()

.split(x)

Module: Built-in

2nd [catalog]

Syntax:var.split(x)

Description: Method returns a list by specified separator. See Python documentation for more details.

Example:

```
>>> a="red,blue,green"
>>> a.split(",")
['red', 'blue', 'green']
```

sqrt()

Module: math

math Modul
1:math
3:sqrt()

Syntax: sqrt(x)

Description: Returns square root of x.

Example:

2nd [catalog]

```
>>>from math import *
>>>sqrt(25)
5.0
```

[Fns…] > Modul
1:math
3:sqrt()

import commands can be found in 2nd [catalog].

store_list("name",var) 1-6

Module: ti_system

Syntax: store_list("name",var) 1-6

Description: Stores a list from the execution of a Python script to an OS list variable "name" where var is a defined Python list. List length must be less than or equal to 100.

Argument: "**name**"

For OS L1-L6

1 - 6

"1" - "6"

'1' - '6'

For OS custom list "name"

----- Max 5 characters, numbers or letters, starting with letters, and letters must be uppercase.

Examples:

"ABCDE"

"R12"

"L1" will be custom L1 and not OS L1

Reminder: Python is double precision which is more digits than supported in the OS.

Example:

```
>>>a=[1,2,3]
>>>store_list("1",a)
>>>
```

Quit the Pyton App and press [2nd][L1] (above [1]) and [enter] on the Home Screen to see list [L1] as {1 2 3}.

[2nd] [catalog]

[2nd][rcl]
ti_system
3:var=store_list
()

[Fns…]>Modul
or [math]
4:ti_system
3:var=store_list
()

import commands can be found in [2nd] [catalog] or in the ti_system Modul menu.

str()

Module: Built-in

2nd [catalog]

Syntax: str(argument)

Description: Converts "argument" to a string.

[Fns…]
> Type
3 :str()

Example:

```
>>>x=2+3
>>>str(x)
'5'
```

sum()

Module: Built-in

2nd [list] (above stat) List
9:sum()

Syntax: sum(sequence)

Description: Returns the sum of the items in a sequence.

2nd [catalog]

Example:

[Fns…] > List
9:sum()

```
>>>listA=[2,4,6,8,10]
>>>sum(listA)
30
```

T

tan()

Module: math

[sin] 5:tan()

Syntax: tan(x)

Description: Returns tangent of x. Angle argument is in radians.

[Fns…] > Modul 1:math… > Trig 5:tan()

Example:

```
>>>from math import *
>>>tan(pi/4)
1.0
```

[2nd] [catalog]

import commands can be found in [2nd] [catalog]

text_at(row,"text","align")

Module: ti_plotlib

[2nd] [catalog]

Syntax: plt.text_at(row,"text","align")

[Fns…]>Modul or [math] 5:ti_plotlib…> Draw 0:text_at()

Description: Display "text" in plotting area at specified "align".

row	integer 1 through 12
"text"	string is clipped if too long
"align"	"left" (default) "center" "right"
optional	1 clears line prior to text (default) 0 line doe not clear

import commands can be found in [2nd] [catalog] or in the ti_plotlib Setup menu.

Example:

See sample program: DASH1.

time.function

Module: Built-in

[2nd] [catalog]

Syntax: time.function

Description: Use after import time to access a function in the time module.

Example:

See:[Fns…]>Modul: time and ti_system modules.

title("title")

Module: ti_plotlib

[2nd] [catalog]

Syntax: plt.title("title")

[Fns…]>Modul or [math] 5:ti_plotlib...> Setup 8:title()

Description: "title" displays centered on top line of window. "title is clipped if too long.

Example:

See sample program: COLORLIN.

import commands can be found in [2nd] [catalog] or in the ti_plotlib Setup menu.

ti_hub.function

Module: ti_hub 2nd [catalog]

Syntax: ti_hub.function

Description: Use after import ti_hub to access a function in the ti_hub module.

Example:

See:[Fns…]>Modul: ti_hub module.

ti_system.function

Module: ti_system 2nd [catalog]

Syntax: ti_system.function

Description: Use after import ti_system to access a function in the ti_system module.

Example:

```
>>> # Shell Reinitialized
>>>import ti_system
>>>ti_system.disp_at(6,8,"texte")

  texte>>>|

#will appear at row 6, col 8 with Shell
prompt as shown.
```

True

Keyword

Description: Returns True when statement executed is True. "True" represents the true value of objects of type bool.

Example:

```
>>>64>=32
True
```

2nd [test] (above math)

2nd [catalog]

[Fns…] > Ops
A:True

[a A #]

trunc()

Module: math

Syntax: trunc(x)

Description: Returns the real value x truncated to an integer.

Example:

```
>>>from math import *
>>>trunc(435.867)
435
```

math Modul
1:math…
0:trunc()

2nd [catalog]

[Fns…] > Modul
1:math…
0:trunc()

import commands can be found in 2nd [catalog]

try:

Keyword

Description: Use try code block to test the code block for errors. Also used with except and finally. See Python documentation for more details.

2nd [catalog]

tuple(sequence)

Module: Built-in

2nd [catalog]

Syntax: tuple(sequence)

Description: Converts sequence into a tuple. See Python documentation for more details.

Example:

```
>>>a=[10,20,30]
>>>tuple(a)
(10,20,30)
```

type()

Module: Built-in

2nd [catalog]

Syntax: type(object)

[Fns…]>Type>6:type ()

Description: Returns the type of the object.

Example:

```
>>>a=1.25
>>>print(type(a))
<class 'float'>
>>>b=100
>>>print(type(b))
<class 'int'>
>>>a=10+2j
>>>print(type(c))
<class 'complex'>
```

uniform(min,max)

Module: random

Syntax: uniform(min,max)

Description: Returns a random number x (float) such that min <= x <= max.

Example:

```
>>>from random import *
>>>uniform(0,1)
0.476118
>>>uniform(10,20)
16.2787
```

Results will vary given a random output.

math Modul
2:random…
Random
3:uniform
(min,max)

2nd [catalog]

[Fns…] > Modul
2:random…
Random
3:uniform
(min,max)

import commands can be found in 2nd [catalog]

wait_key()

Module: ti_system

2nd [catalog]

Syntax: wait_key()

Description: Returns a combined keycode representing the key pressed, merged with 2nd and/or alpha. The method waits for a key to be pressed before returning to the program.

Example:

See:[Fns…]>Modul: time and ti_system modules.

See: Keypad mapping for wait_key().

while condition:

Keyword

[Fns…] Ctl
8:while condition:

Syntax: while condition:

Description: Executes the statements in the following code block until "condition" evaluates to False.

2nd [catalog]

Example:

```
>>> x=5
>>> while x<8:
…        x=x+1
…        print(x)
…
…
6
7
8
```

window(xmin,xmax,ymin,ymax)

Module: ti_plotlib

2nd catalog

Syntax: plt.window(xmin,xmax,ymin,ymax)

Description: Defines the plotting window by mapping the the specified horizontal interval (xmin, xmax) and vertical interval (ymin, ymax) to the allotted plotting area (pixels).

This method must be executed before any other ti_plotlib module commands are executed.

The ti_plotlib Properties vars, xmin, xmax, ymin, ymax will be updated to the argument values. The default values are (-10, 10, -6.56, 6.56).

Example:

See sample program: GRAPH.

[Fns...]>Modul or math 5:ti_plotlib...> Setup 4:window()

import commands can be found in 2nd catalog or in the ti_plotlib Setup menu.

with

Keyword

2nd catalog

Description: See Python documentation for more details.

xmax default 10.00

Module: ti_plotlib

Syntax: plt.xmax default 10.00

Description: Specified variable for window arguments defined as plt.xmax.

Default values:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Example:

See sample program: GRAPH.

2nd [**catalog**]

[Fns...]>Modul or **math** 5:ti_plotlib...> Properties 2:xmax

import commands can be found in **2nd** [**catalog**] or in the ti_plotlib Setup menu.

xmin default -10.00

Module: ti_plotlib

Syntax: plt.xmin default -10.00

Description: Specified variable for window arguments defined as plt.xmin.

Default values:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Example:

See sample program: GRAPH.

2nd [catalog]

[Fns...]>Modul or math 5:ti_plotlib...> Properties 1:xmin

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

yield

Keyword

2nd [catalog]

Description: Use yield to end a function. Returns a generator. See Python documentation for more details.

ymax default 6.56

Module: ti_plotlib

2nd [catalog]

Syntax: plt.ymax default 6.56

[Fns...]>Modul or math
5:ti_plotlib...>
Properties
4:ymax

Description: Specified variable for window arguments defined as plt.ymax.

Default values:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

Example:

See sample program: GRAPH.

ymin default -6.56

Module: ti_plotlib

Syntax: plt.ymin default -6.56

Description: Specified variable for window arguments defined as plt.ymin.

Default values:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Example:

See sample program: GRAPH.

2nd [catalog]

[Fns…]>Modul or math
5:ti_plotlib...>
Properties
3:ymin

import commands can be found in 2nd [catalog] or in the ti_plotlib Setup menu.

Symbols

@

Operator

[alpha] [θ] (above [3])

Description: Decorator – See general Python documentation for details.

[2nd] [catalog]

<<

Operator

[2nd] [catalog]

Syntax: x<<n

Description: Bitwise left shift by n bits.

>>

Operator

[2nd] [catalog]

Syntax: x>>n

Description: Bitwise right shift by n bits.

|

Operator

[2nd] [catalog]

Syntax: x|y

Description: Bitwise or.

&

Operator

[2nd] [catalog]

Syntax: x&y

Description: Bitwise and.

^

Operator $\boxed{\text{2nd}}$ [catalog]

Syntax: x^y

Description: Bitwise exclusive or.

~

Operator $\boxed{\text{2nd}}$ [catalog]

Syntax: ~x

Description: Bitwise not; the bits of x inverted.

x<=y

Operator

Syntax: x<=y

Description: Comparison; x less than or equal to y.

Example:

```
>>>2<=5
True
>>>3<=0
False
```

math
1:math > Ops
7:x<=y

2nd [catalog]

[Fns…] > Ops
7:x<=y

[a A #]

x<y

Operator

Syntax: x<y

Description: Comparison; x strictly less than y.

Example:

```
>>>6<10
True
>>>12<-15
False
```

math
1:math > Ops
6:x<y

2nd [catalog]

[Fns…] > Ops
6:x<y

[a A #]

x>=y

Operator

Syntax: x>=y

Description: Comparison; x greater than or equal to y.

Example:

```
>>>35>=25
True
>>>14>=65
False
```

math
1:math > Ops
5:x>=y

2nd [catalog]

[Fns…] > Ops
5:x>=y

[a A #]

x>y

Operator

Syntax: x>y

Description: Comparison; x strictly greater than y.

Example:

```
>>>35>25
True
>>>14>65
False
```

math
1:math > Ops
4:x>y

2nd [catalog]

[Fns…] > Ops
4:x>y

[a A #]

x!=y

Operator

Syntax: x!=y

Description: Comparison; x not equal to y.

Example:

```
>>>35!=25
True
>>>14!=10+4
False
```

[math]
1:math > Ops
3:x!=y

[2nd] [catalog]

[Fns…] > Ops
3:x!=y

[a A #]

x==y

Operator

Syntax: x==y

Description: Comparison; x is equal to y.

Example:

```
>>>75==25+50
True
>>>1/3==0.333333
False
>>>1/3==0.3333333        #equal to stored Python value
True
```

[math]
1:math > Ops
2:x==y

[2nd] [catalog]

[Fns…] > Ops
2:x==y

[a A #]

x=y

Operator

[sto→]

Syntax: x=y

Description:y is stored in variable x

Example:

```
>>>A=5.0
>>>print(A)
5.0
>>>B=2**3        #Use [ ^ ] on keypad for **
>>>print(B)
8
```

[math]
1:math > Ops
1:x=y

[2nd] [catalog]

[Fns…] > Ops
1:x=y

[a A #]

\

Delimiter

[2nd] [catalog]

Description: Backslash character.

[a A #]

\t

Delimiter

[2nd][catalog]

Description: Tab space between strings or characters.

\n

Delimiter

[2nd] [catalog]

Description: New line to display string neatly on the screen.

' '

Delimiter

Description: Two single quotes paste.

Example:

```
>>>eval('a+10')
17
```

2nd [mem]
(above +)

2nd [catalog]

[a A #]

" "

Delimiter

Description: Two double quotes paste.

Example:

```
>>>print("Ok")
```

alpha ["]
(above +)

2nd [catalog]

[a A #]

Appendix

Selected TI-Python Built-in, Keywords, and Module Content

Built-ins

Built-ins	Built-ins	Built-ins
__name__	abs -- <function>	BaseException -- <class 'BaseException'>
__build_class__ -- <function>	all -- <function>	ArithmeticError -- <class 'ArithmeticError'>
__import__ -- <function>	any -- <function>	AssertionError -- <class 'AssertionError'>
__repl_print__ -- <function>	bin -- <function>	AttributeError -- <class 'AttributeError'>
bool -- <class 'bool'>	callable -- <function>	EOFError -- <class 'EOFError'>
bytes -- <class 'bytes'>	chr -- <function>	Exception -- <class 'Exception'>
bytearray -- <class 'bytearray'>	dir -- <function>	GeneratorExit -- <class 'GeneratorExit'>
dict -- <class 'dict'>	divmod -- <function>	ImportError -- <class 'ImportError'>
enumerate -- <class 'enumerate'>	eval -- <function>	IndentationError -- <class 'IndentationError'>
filter -- <class 'filter'>	exec -- <function>	IndexError -- <class 'IndexError'>
float -- <class 'float'>	getattr -- <function>	KeyboardInterrupt -- <class 'KeyboardInterrupt'>
int -- <class 'int'>	setattr -- <function>	ReloadException -- <class

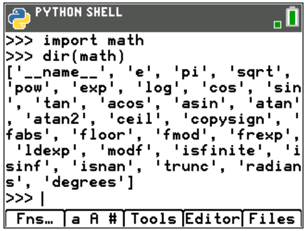
Built-ins	Built-ins	Built-ins
		'ReloadException'>
list -- <class 'list'>	globals -- <function>	KeyError -- <class 'KeyError'>
map -- <class 'map'>	hasattr -- <function>	LookupError -- <class 'LookupError'>
memoryview -- <class 'memoryview'>	hash -- <function>	MemoryError -- <class 'MemoryError'>
object -- <class 'object'>	help -- <function>	NameError -- <class 'NameError'>
property -- <class 'property'>	hex -- <function>	NotImplementedError -- <class 'NotImplementedError'>
range -- <class 'range'>	id -- <function>	OSError -- <class 'OSError'>
set -- <class 'set'>	input -- <function>	OverflowError -- <class 'OverflowError'>
slice -- <class 'slice'>	isinstance -- <function>	RuntimeError -- <class 'RuntimeError'>
str -- <class 'str'>	issubclass -- <function>	StopIteration -- <class 'StopIteration'>
super -- <class 'super'>	iter -- <function>	SyntaxError -- <class 'SyntaxError'>
tuple -- <class 'tuple'>	len -- <function>	SystemExit -- <class 'SystemExit'>
type -- <class 'type'>	locals -- <function>	TypeError -- <class 'TypeError'>
zip -- <class 'zip'>	max -- <function>	UnicodeError -- <class 'UnicodeError'>
classmethod -- <class 'classmethod'>	min -- <function>	ValueError -- <class 'ValueError'>
staticmethod -- <class 'staticmethod'>	next -- <function>	ZeroDivisionError -- <class 'ZeroDivisionError'>

Built-ins	Built-ins	Built-ins
Ellipsis -- Ellipsis	oct -- <function>	
	ord -- <function>	
	pow -- <function>	
	print -- <function>	
	repr -- <function>	
	round -- <function>	
	sorted -- <function>	
	sum -- <function>	

keywords

keywords	keywords	keywords
False	elif	lambda
None	else	nonlocal
True	except	not
and	finally	or
as	for	pass
assert	from	raise
break	global	return
class	if	try
continue	import	while
def	in	with
del	is	yield

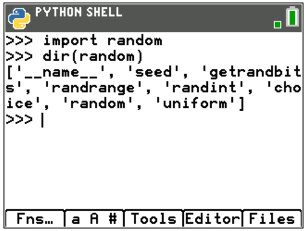
math



```
>>> import math
>>> dir(math)
['__name__', 'e', 'pi', 'sqrt',
'pow', 'exp', 'log', 'cos', 'sin
', 'tan', 'acos', 'asin', 'atan'
, 'atan2', 'ceil', 'copysign', '
fabs', 'floor', 'fmod', 'frexp',
 'ldexp', 'modf', 'isfinite', 'i
sinf', 'isnan', 'trunc', 'radian
s', 'degrees']
>>> |
```

math	math	math
__name__	acos -- <function>	frexp -- <function>
e -- 2.71828	asin -- <function>	ldexp -- <function>
pi -- 3.14159	atan -- <function>	modf -- <function>
sqrt -- <function>	atan2 -- <function>	isfinite -- <function>
pow -- <function>	ceil -- <function>	isinf -- <function>
exp -- <function>	copysign -- <function>	isnan -- <function>
log -- <function>	fabs -- <function>	trunc -- <function>
cos -- <function>	floor -- <function>	radians -- <function>
sin -- <function>	fmod -- <function>	degrees -- <function>
tan -- <function>		

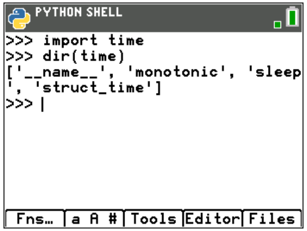
random



```
>>> import random
>>> dir(random)
['__name__', 'seed', 'getrandbit
s', 'randrange', 'randint', 'cho
ice', 'random', 'uniform']
>>> |
```

random	random	random
__name__	randint -- <function>	
seed -- <function>	choice -- <function>	
getrandbits -- <function>	random -- <function>	
randrange -- <function>	uniform -- <function>	

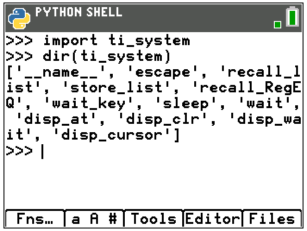
time



```
>>> import time
>>> dir(time)
['__name__', 'monotonic', 'sleep
', 'struct_time']
>>> |
```

time	time	time
__name__		
monotonic		
sleep		
struc_time		

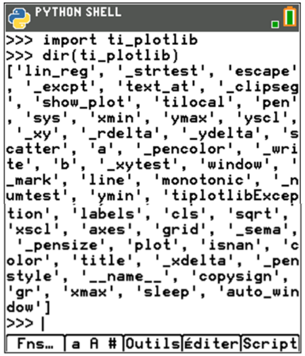
ti_system



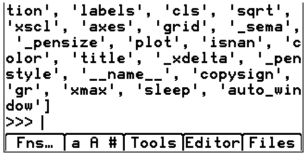
```
>>> import ti_system
>>> dir(ti_system)
['__name__', 'escape', 'recall_l
ist', 'store_list', 'recall_RegE
Q', 'wait_key', 'sleep', 'wait',
 'disp_at', 'disp_clr', 'disp_wa
it', 'disp_cursor']
>>> |
```

ti_system	ti_system	ti_system
__name__	recall_RegEQ	disp_at
escape	wait_key	disp_clr
recall_list	sleep	disp_wait
store_list	wait	disp_cursor

ti_plotlib



```
>>> import ti_plotlib
>>> dir(ti_plotlib)
['lin_reg', '_strtest', 'escape'
, '_excpt', 'text_at', '_clipseg
', 'show_plot', 'tilocal', 'pen'
, 'sys', 'xmin', 'ymax', 'yscl',
 '_xy', '_rdelta', '_ydelta', 's
catter', 'a', '_pencolor', '_wri
te', 'b', '_xytest', 'window', '
_mark', 'line', 'monotonic', '_n
umtest', 'ymin', 'tiplotlibExcep
tion', 'labels', 'cls', 'sqrt',
'xscl', 'axes', 'grid', '_sema',
 '_pensize', 'plot', 'isnan', 'c
olor', 'title', '_xdelta', '_pen
style', '__name__', 'copysign',
'gr', 'xmax', 'sleep', 'auto_win
dow']
>>> |
```

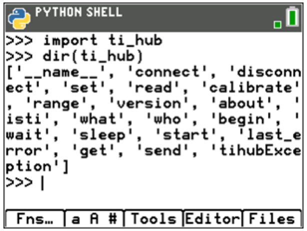


```
tion', 'labels', 'cls', 'sqrt',
'xscl', 'axes', 'grid', '_sema',
 '_pensize', 'plot', 'isnan', 'c
olor', 'title', '_xdelta', '_pen
style', '__name__', 'copysign',
'gr', 'xmax', 'sleep', 'auto_win
dow']
>>> |
```

ti_plotlib	ti_plotlib	ti_plotlib
__name__	a	grid
lin_reg	_pencolor	-pensize
_strtest	_write	_sema
escape	b	-pensize
_except	_xytest	plot
text_alt	window	isnan
_clipseg	_mark	color

ti_plotlib	ti_plotlib	ti_plotlib
show-plot	line	title
tilocal	monotonic	_xdelta
pen	_ntest	_penstyle
sys	ymin	copysign
xmin	tiplotlibException	gr
ymax	lables	xmax
yscl	cls	sleep
_xy	sqrt	auto_window
_rdelta	xscl	
_ydelta	axes	
scatter		

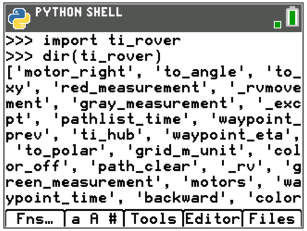
ti_hub



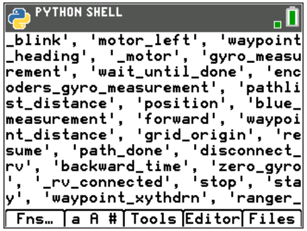
```
>>> import ti_hub
>>> dir(ti_hub)
['__name__', 'connect', 'disconn
ect', 'set', 'read', 'calibrate'
, 'range', 'version', 'about', '
isti', 'what', 'who', 'begin', '
wait', 'sleep', 'start', 'last_e
rror', 'get', 'send', 'tihubExce
ption']
>>> |
```

ti_hub	ti_hub	ti_hub
__name__	version	last_error
connect	begin	sleep
disconnect	start	tihubException
set	about	wait
read	isti	get
calibrate	what	send
range	who	

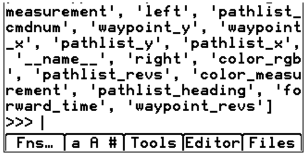
ti_rover



```
>>> import ti_rover
>>> dir(ti_rover)
['motor_right', 'to_angle', 'to_
xy', 'red_measurement', '_rvmove
ment', 'gray_measurement', '_exc
pt', 'pathlist_time', 'waypoint_
prev', 'ti_hub', 'waypoint_eta',
 'to_polar', 'grid_m_unit', 'col
or_off', 'path_clear', '_rv', 'g
reen_measurement', 'motors', 'wa
ypoint_time', 'backward', 'color
```



```
_blink', 'motor_left', 'waypoint
_heading', '_motor', 'gyro_measu
rement', 'wait_until_done', 'enc
oders_gyro_measurement', 'pathli
st_distance', 'position', 'blue_
measurement', 'forward', 'waypoi
nt_distance', 'grid_origin', 're
sume', 'path_done', 'disconnect_
rv', 'backward_time', 'zero_gyro
', '_rv_connected', 'stop', 'sta
y', 'waypoint_xythdrn', 'ranger_
```



```
measurement', 'left', 'pathlist_
cmdnum', 'waypoint_y', 'waypoint
_x', 'pathlist_y', 'pathlist_x',
 '__name__', 'right', 'color_rgb
', 'pathlist_revs', 'color_measu
rement', 'pathlist_heading', 'fo
rward_time', 'waypoint_revs']
>>> |
```

ti_rover	ti_rover	ti_rover
__name__	color_blink	_rv
motor_right	motor_left	stay
to_angle	waypoint_heading	waypoint_xythdrn
to_xy	_motor	ranger_measurement
red_measurment	gyro_measutrment	left
rvmovement	wait_until_done	pathlist_cmdnum
gray_measurment	encoders_gyro_measurement	waypoint_y
_excpt	pathlist_distance	waypoint-x
ti_hub	position	pathlist_y
waypoint_prev	blue_measurement	pathlist_x

ti_rover	ti_rover	ti_rover
pathlist_time	forward	right
waypoint_revs	waypoint_distance	color_rgb
to_polar	grid_origin	pathlist-revs
waypoint_eta	resume	color_measurement
color_off	path_done	tiroverException
grid_m_unit	disconnect_rv	forward_time
path_clear	backward_time	pathlist_heading
green_measurement	zero-gyro	
waypoint_time	_rv_connected	
motors	stop	
backward		

.

Keypad mapping for wait_key()

statplot f1 y=	73 48 73	tblset f2 window	72 75 72	format f3 zoom	46 87 46	calc f4 trace	90 59 90	table f5 graph	68 74 68
2nd	---- ---- ----	quit mode	69 64 69	ins del	10 11 10	|◀ ◀	2 14 2	▶| ▶	1 15 1
A-lock alpha	---- ---- ----	link X,T,θ,n	180 65 180	list stat	49 58 49	▲	3 ---- 3	▼	4 ---- 4
test A math	50 51 154	angle B apps	44 57 155	draw C prgm	45 47 156	distr vars	53 56 53	clear	9 9 9
matrix D x^{-1}	182 55 157	$\sin^{-1}$ E sin	183 184 158	$\cos^{-1}$ F cos	185 186 159	$\tan^{-1}$ G tan	187 188 160	π H ^	132 181 161
√ I x^2	189 190 162	EE J ,	139 152 163	{ K (	133 236 164	} L)	134 237 165	e M ÷	131 239 166
10^x N log	193 194 167	u O 7	149 249 168	v P 8	150 250 169	w Q 9	151 251 170	[R ×	130 135 171
e^x S ln	191 192 172	L4 T 4	146 246 173	L5 U 5	147 247 174	L6 V 6	148 248 175	] W −	129 136 176
rcl X sto→	138 12 177	L1 Y 1	143 243 178	L2 Z 2	144 244 179	L3 θ 3	145 245 204	mem " +	128 54 203
off on	break ---- ----	catalog ␣ 0	142 62 153	*i* : .	141 238 198	ans ? (−)	140 197 202	entry solve enter	5 13 5

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.