10 Minutes of Code - Python 84 Plus CE Python

Unit 1: Getting Started with Python and the TI-Innovator™ Hub — Skill Builder 1: Light It Up

In this lesson, you will learn the basics of writing and running a Python program and using the 'light' (the red LED) on the TI-Innovator Hub.

Objectives:

- Create and run a Python program
- Control the light on the TI-Innovator Hub

Teacher Tip: This course introduces students to the TI-Innovator Hub using Python programming on the TI 84 CE Plus Python and assumes students have no programming experience. For a more direct introduction to Python programming, see the other course on TI 84 CE Plus Python programming available at education.ti.com 'TI Codes'.

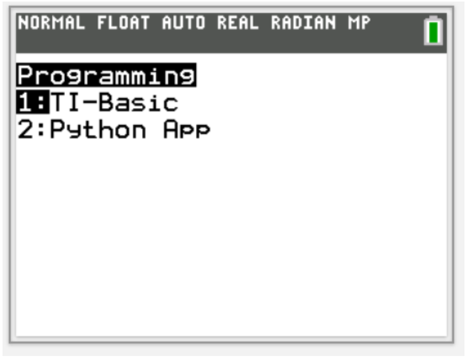
Python on the TI 84 CE Plus Python is implemented using MicroPython, which is a subset of Python. Two important details:

1. MicroPython does not include all standard Python functions, even in the 'built-in' or 'standard' features. Example: shuffle() is not part of the list() functions.
2. Not all MicroPython modules or functions are on the TI-Nspire CX II menus. The most commonly used functions are on the menus.

In order to use the TI-Innovator Hub, a Python program must import the ti_hub module:
from ti_hub import *.

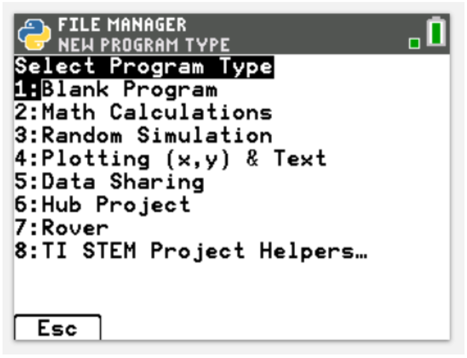
Welcome to the world of TI-Innovator Hub programming using Python on your TI 84 CE Plus Python. Your first program will operate the red LED on the TI-Innovator Hub circuit board. It is hard to see on the board, but when you turn it on you will know it!

1. Open a new program, press number **2 Python App**.



```
Programming
1:TI-Basic
2:Python App
```

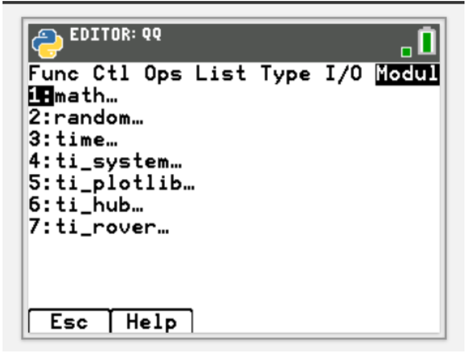
2. File manager is on the top of the screen. Press **Zoom** (New). Type in the name of your program, LIGHTS, then press **Zoom** (Types). Select a program type, for this program we will use **1. Blank Program.** Then type **Graph** for OK.



```
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
```

Teacher Tip: .Python is 'lightweight' by design: the 'built-ins' are all that are necessary to write many robust programs. Other modules are available to give Python more power. TI developers have created custom modules that make programming the TI 84 Plus CE Python in Python rich and fun.

3. You are now in the Python Editor. Press the **y=, function** key. Each of these items contains related Python programming tools. Our main interest for now is the **TI Hub** menu. Tab over to the Module tab. Select **6: ti hub** to import the hub menu.



```
EDITOR: QQ
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
```

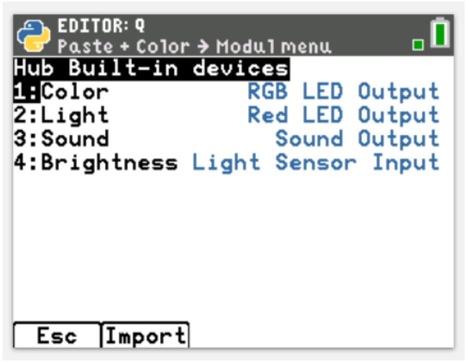
from ti_hub import *

This Python command gives you the tools (commands) needed to operate the devices on (or connected to) the TI-Innovator Hub.

Also*, this statement will check to see if a TI-Innovator Hub is connected. If not, then the program will not run.*

Teacher Tip: The ti_hub module *contains a function* that checks to see if the TI-Innovator Hub is present.

4. Now you want to Import the hub built in devices, select **1:for Hub Built-in devices.** Press enter. Now we want to use **2**: Light for the Red LED Output



```
EDITOR: Q
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
```

5. The next statement you will use is:

   light.on()

   Can you guess what it does?

   This statement is found on:

   y=: fns > Module: > 8:Light > 1: > on().

   All Hub-related features are on the **TI Hub** menu.

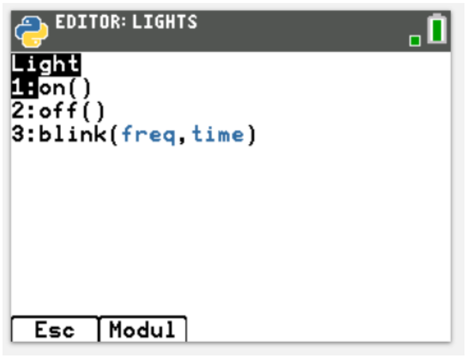
Teacher Tip: Notice that there is very little typing needed! Much of the coding is accessible right on the menus.

Repeat: If you try to run this program *without* a TI-Innovator Hub attached you will get an error message and the program stops. The **import ti_hub** statement checks to make sure the TI-Innovator Hub is available.

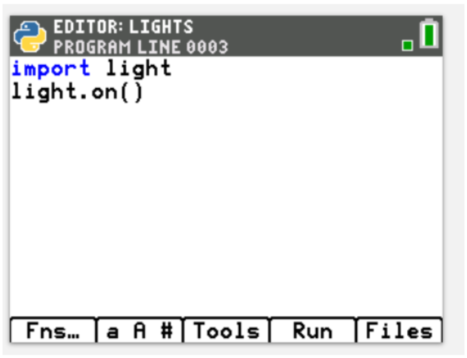
6. You are now ready to run this amazing program. You could use

   Trace **> Run**

   Your TI 84 Plus CE Python screen will look like this one. Pressing **trace, run** added a page to your document and placed a Python Shell app on it. You are now using the Python Shell. This is the place where Python programs are executed (actually, the only place). If you look at the TI-Innovator Hub you will see a red LED lit up. That's the result of the **light.on()** statement in your program.


```
EDITOR: LIGHTS
Light
1:on()
2:off()
3:blink(freq,time)
```

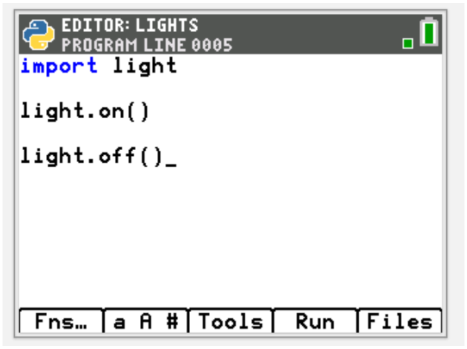
Teacher Tip: light.on() is a much more complex statement than it looks. light is an instance of a class created within the **ti_hub** module and .on() is a method of the class that turns the light on. On the surface, it is an easy command to use, but underneath there is a lot of work going on. This is the power of Python at work: it makes physical programming accessible to even complete beginners.



```
import light
light.on()
```

7. Can you guess what command turns the light off? You can find it on:

   fns > Module >8: Light > 2: light off()



```
import light

light.on()

light.off()_
```

8. Add the **light.off()** command after the **light.on()** command. It is okay to skip lines in the Python Editor. They have no effect on the execution and they make the code easier to read. Run the program again. Do you see the LED blink quickly? Too fast?

   In the next few steps, we add a feature that lets you control just how long the LED stays lit.

Teacher Tip: Another 'import' command is coming. In later lessons we will use a programming project 'template' that provides most of the necessary building blocks for TI-Innovator Hub programming.

9. Place your cursor on the line below **import light**.

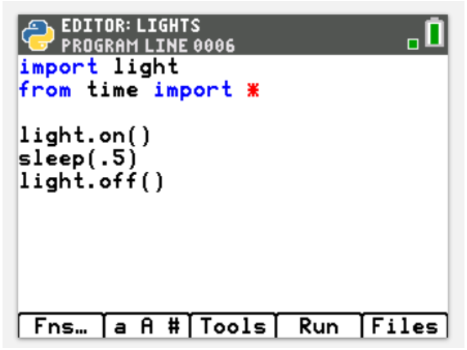
   Press y=, Fns, module, 3: time, and 1: from time import*

   from time import *

   Between the **light.on()** and **light.off()** statements add the statement

   sleep(*seconds*)

   found on the **Time** menu as well.



```
import light
from time import *

light.on()
sleep(.5)
light.off()
```

The word '*seconds*' is a placeholder.

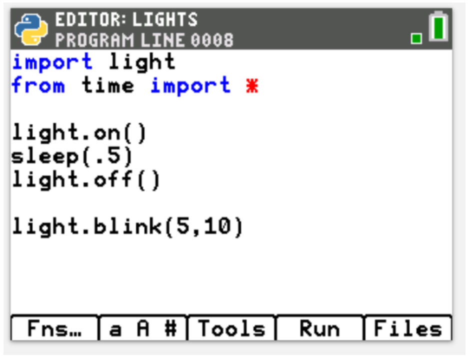
Replace it with a number, like 2 or 3.

The sleep**()** function tells the computer to wait or pause for that many seconds before going to the next statement in the program.

When you run the program now (press **trace to Run**), the LED will stay lit for your chosen number of seconds before turning off.

10. To make the light blink, you could repeat the sequence of statements in the program or….

    on the Light there is also a **blink()** function that causes the LED to blink. **light.blink()** has two parameters: *frequency* and *time*. Replace both with numbers and figure out the pattern. Pay attention to the pop-up tool tip!


EDITOR: LIGHTS
PROGRAM LINE 0008

```
import light
from time import *

light.on()
sleep(.5)
light.off()

light.blink(5,10)
```

Teacher Tip: The **blink()** function has a tool tip when '*frequency*' is selected: 0.1–20 Hz. Hz stands for Hertz and means 'cycles per second' so a frequency of 3 means that the LED will blink 3 times in one second. The '*time'* parameter (0.1–100 s) tells the TI-Innovator Hub for how many seconds to blink. So, this is a simple multiplication example: light.blink(3,5) will make the LED blink 3*5=15 times in 5 seconds.

To repeat: **.on()**, **.off()**, and **.blink()** are all *methods* of a light *class* of which the light variable is an *instance*. This is Object-Oriented Programming, or OOP, in which the TI-Innovator Hub and Rover tools are designed.