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-Nspire CX II and assumes students have no programming experience. For a more direct introduction to Python programming, see the other course on TI-Nspire CX II Python programming available at education.ti.com 'TI Codes'.

Python on the TI-Nspire CX II is implemented using MicroPython, which is a subset of Python. Two important details:

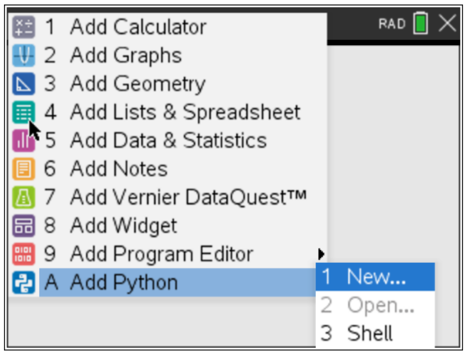
1. MicroPython does not include all standard Python functions, even in the 'built-in' or 'standard' features. Example: shuffle() is not part of the list() functions.
2. Not all MicroPython modules or functions are on the TI-Nspire CX II menus. The most commonly used functions are on the menus.

In order to use the TI-Innovator Hub, a Python program must import the ti_hub module: **from ti_hub import ***. For Rover (Units 4, 5, 6): **import ti_rover as rv**. Thus, all Rover statements begin with 'rv'. For the TI-RGB Array (Unit 7): from ti_hub import* and then create an instance of the RGB Array class: *var*=**rgb_array()**

The first three units of this course focus on the built-in devices of the hub: **LIGHT**, **COLOR**, **SOUND**, and the **BRIGHTNESS** sensor. The next three units (4, 5, 6) deal with Rover functionality and Unit 7 covers the TI-RGB Array. Texas Instruments has created all the 'ti_' Python modules to support the unique features of the TI-Nspire CX II, TI-Innovator Hub, and TI-Innovator™ Rover. The TI-Innovator Hub and Rover modules use an Object-Oriented Approach to coding, so the commands and the syntax are very different from TI-Basic and even the other Python course.

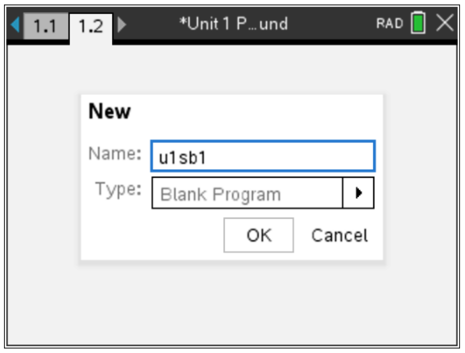
Welcome to the world of TI-Innovator Hub programming using Python on your TI-Nspire CX II! Your first program will operate the red LED on the TI-Innovator Hub circuit board. It is hard to see on the board, but when you turn it on you will know it!

1. Open a new TI-Nspire™ document. The available applications are listed. Select **Add Python** and then select **New…**.



2. Type a name for the Python program (we use **u1sb1**) and press **enter**.

   We will look at the 'Type:' menu later.



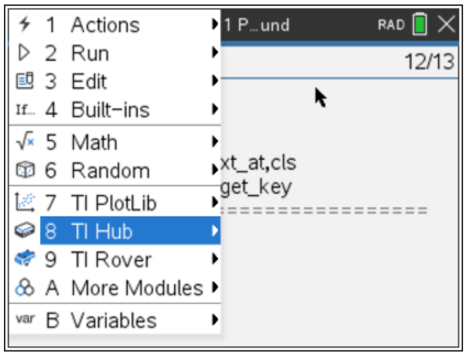
Teacher Tip: Python programs (files) are part of the TI-Nspire document. If you select **Open…** from the menu you will see all the python files in the document (if any).
The next step imports functions necessary to control the TI-Innovator Hub. Python is 'lightweight' by design: the 'built-ins' are 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-Nspire CX II in Python rich and fun.

3. You are now in the Python Editor. Press the **menu** key. Each of these menu items contains related Python programming tools. Our main interest for now is the **TI Hub** menu. Select the top item from that menu:

   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. The next statement you will use is:

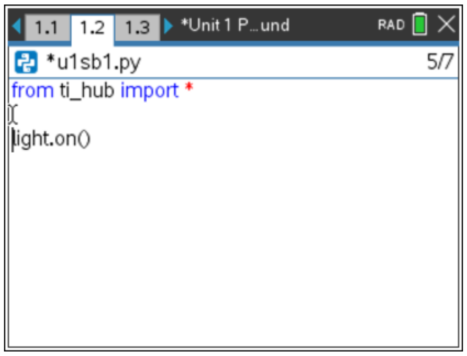
   light.on()

   Can you guess what it does?

   This statement is found on

   menu > TI Hub > Hub Built-in Devices > Light Output > on().

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



```
from ti_hub import *

light.on()
```

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.

NOT FOR BEGINNERS: To allow a TI-Innovator Hub program to run *without* a TI-Innovator Hub attached (to check for other coding issues) use a try…except structure:

```
try:
    from ti_hub import *
    hub = True
except:
    hub = False
```

and, throughout the program, whenever a hub instruction is called, check the hub state first:

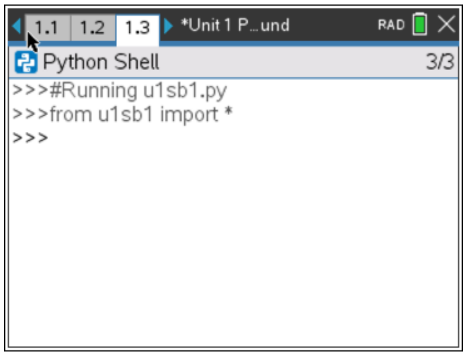
```
if hub: light.on()
```

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

   menu > Run > Run (Ctrl+R)

   or simply press **ctrl+R** on your keypad. Your TI-Nspire CX II screen will look like this one. Pressing **ctrl+R** added a page to your document and placed a Python Shell app on it. You are now using the Python Shell. The Shell is similar to the Calculator application on the TI-Nspire. This is the place where Python programs are executed (actually, the only place).

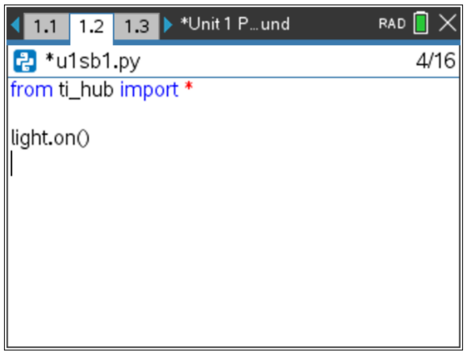
   The **>>>** symbol is the Python command prompt. It is waiting for the next command. But if you look at the TI-Innovator Hub you will see a red LED lit up. That's the result of the **light.on()** statement in your program.



```
>>>#Running u1sb1.py
>>>from u1sb1 import *
>>>
```

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.

6. Your Python program is now on the page before this Shell page. To go back to editing your program, press **ctrl+left arrow**. Notice on the TI-Innovator Hub that the red LED is still on.

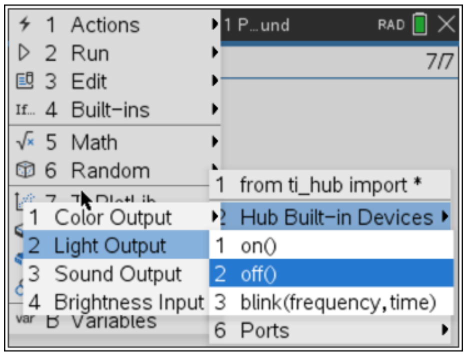


```
from ti_hub import *

light.on()
```

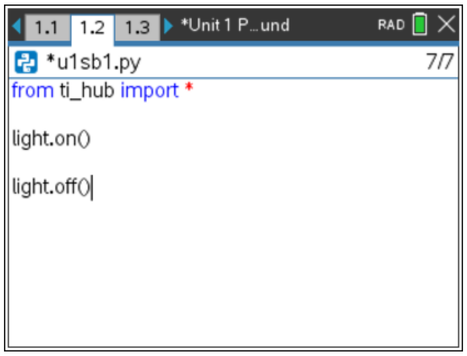
7. Can you guess what command turns the light off? You can find it on:

   menu > TI Hub > Hub Built-in Devices > Light Output > …



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.



```
from ti_hub import *

light.on()

light.off()
```

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 **from ti_hub import** *.

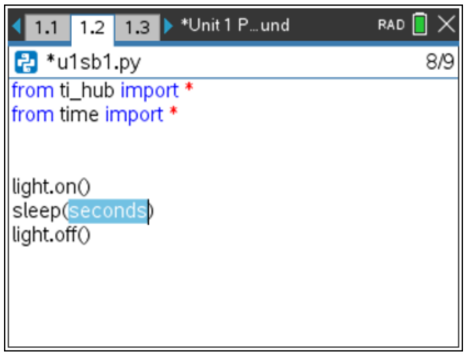
   Press **menu > More Modules > Time** and select:

   from time import *

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

   sleep(*seconds*)

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



```
from ti_hub import *
from time import *


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

The word '*seconds*' is a placeholder.

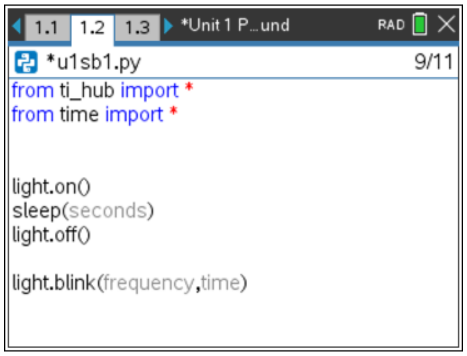
Replace it with a number, like 2 or 3.

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 **ctrl+R**), 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 Output menu 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!



```
from ti_hub import *
from time import *


light.on()
sleep(seconds)
light.off()

light.blink(frequency,time)
```

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.