Unit 7: The TI-RGB Array | **Application: Smart Lights**

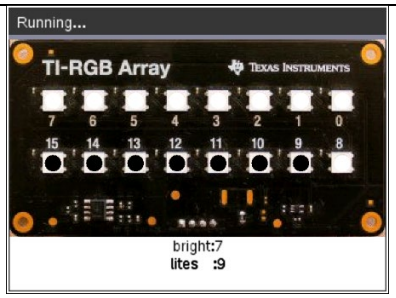
In this application, you will control the number of LEDs lit on the TI-RGB Array using the TI-Innovator Hub's brightness sensor.

Objectives:

- Use the brightness sensor to control the TI-RGB Array
- Adjust the brightness range to suit the TI-RGB Array
- Make sure that all 16 LEDs are impacted by the brightness

Smart Lights

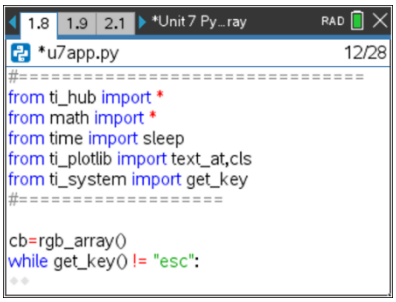
As the room darkens, the lights in the room get brighter. Imagine a 'smart home' with no light switches! Write a program that monitors the brightness and turns on more or less LEDs, as necessary.



Teacher Tip: One of the challenges in this project is getting ALL the LEDs to react to some brightness value, including the end conditions: all on and all off. Your students may have to adjust the **brightness.range()** values to suit your lighting conditions. A smartphone flashlight will be helpful to control brightness.

1. As usual, begin this Python Hub Project using the **rgb_array()** constructor and the **while** loop to terminate the program with **esc**.

```
cb = rgb_array()
while get_key() != "esc":
```



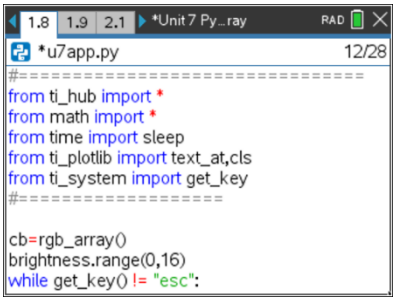
2. Before the **while** loop, set the **brightness.range()** to match the number of LEDs on the TI-RGB Array board that could be lit: 0 to 16.

   Press **menu > TI Hub > Hub Built-in Devices > Brightness Input > range**(min,max) for the statement:

```
brightness.range(0,16)
```

   Use **0,16** because this is the range of the number of LEDs to light up on the board.

   The maximum value the sensor will produce is 16. Is the minimum 0?



Teacher Tip: A value of 0 is hard to achieve on the sensor. The next step converts the brightness value to an integer.

3. In the **while** block, start by reading the **brightness.measurement()** and store the value in a variable (**bright**).

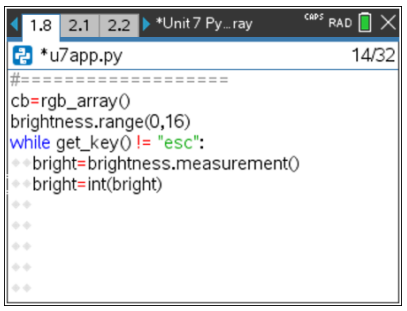
   bright = brightness.measurement()

   The function produces a floating-point number (float, decimal). Convert it to an integer value using:

   bright = int(bright)

   Or, combine the two statements into one operation:

   bright = int(brightness.measurement())



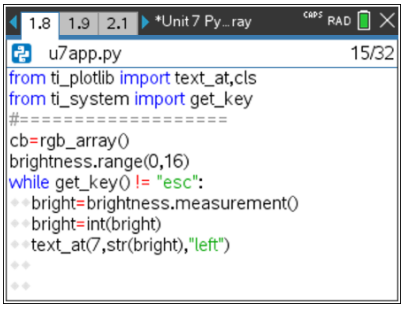
Teacher Tip: Clarity or efficiency? A matter of preference and experience!

4. To test your program: add **text_at()** statement found on **menu > TI Hub > Commands**:

   text_at(7, str(bright), "left")

   Recall that you need **str(bright)** because the **text_at()** function requires a string to display. You can either type **str()** or get it from **menu > Built-ins > Type**.

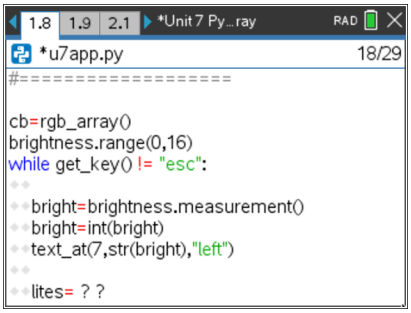
   Run the program to ensure that all seventeen values (0…16) do appear. If not, then adjust the **range()** so that they do. Try using an artificial light source such as a flashlight or the 'flashlight' feature on a smartphone.



Teacher Tip: Varying lighting conditions can make it difficult to achieve perfection but, with practice, students will figure things out.

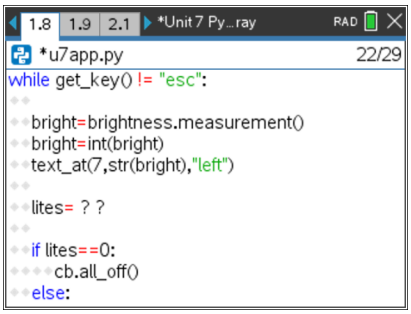
5. Since 0 is the darkest value and 16 is the brightest, we want the number of LEDs lit to be the *opposite:* when bright = 0, there should be 16 LEDs lit and when bright is 16, there should be 0 LEDs lit.
   Write an expression for **lites** in terms of **bright**.

   lites = ? ? ?



6. It's possible that all LEDs should be off:

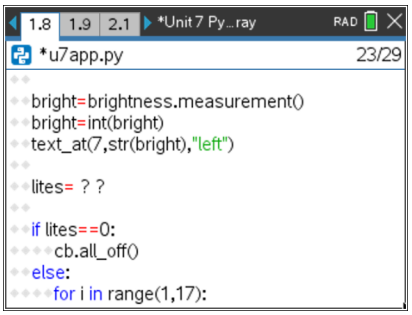
```
if lites == 0:
    cb.all_off()
else:
```



7. We want *all* the LEDs to be affected by the brightness so we will use a **for** loop to control the state of every LED every time. The **lites** variable is a deciding factor when turning a LED on or off:

   for i in range(1,17):

   (Remember that the value 17 is not processed by the loop so **i** *takes on the values from 1 to 16 representing the 16 LEDs.)*



8. Complete the program by adding an **if…else…** statement to tell the TI-Innovator Hub which LEDs are on and which ones are off.

   Hint: If **lites** is 1, then you want to turn on LED *0*. When **lites** is 16, you want to turn on all LEDs (#0 to #15). Use the color (255,255,255) to get a bright white light.

(demoAPP.gif)

Remember to turn **all** the LEDs **off** at the end of the program.

Teacher Tip: The complete program:

```
cb = rgb_arry()
brightness.range (0,16)
while get_key() !=”esc”:
   bright = brightness.measurement()
   bright = int(bright)
   text_at (7,str(bright),”left”)
   lites = 16 – bright
   if lites == 0:
     cb.all_off()
   else
     for i in range (1.17) :
       if i <= lites:
          cb.set(i-1,255,255,255)
       else:
          cb.set(i-1,0,0,0)
 cb.all_off()
```

Tip: Connect a power supply to the TI-Innovator Hub and use cb==rgb_array (“as lamp”) for much brighter LEDs.

For a greater challenge: Instead of each LED being lit or not, have the program *gradually* brighten or darken the next LED as needed. Use a float for **bright** and **lites** and use the decimal part to light up an LED partially.