Unit 7: The TI-RGB Array — Skill Builder 1: Light Them Up

In this lesson, you will learn to control the 16 LEDs on the TI-RGB Array both collectively (all at once) and individually (one at a time).

Note: The simulation that is used in this unit to replicate the lights on the TI-RGB Array is used for demonstration purposes only. This tool is not a TI product and is not available for purchase or distribution by TI.
Caution: Rapidly flashing lights may be disturbing for some students, so it is wise to use **sleep()** *statements to slow things down a bit.*

Objectives:

- Light up ALL LEDs and make them blink in unison using a loop
- Use another loop to light up and turn off the LEDs one at a time

TI-RGB Array

back

breadboard ports

The TI-RGB Array is a circuit board with 16 color LEDs and a controller chip and comes with a short 4-wire cable. It connects to the TI-Innovator Hub using the breadboard (BB) ports on the TI-Innovator Hub. Follow the wiring instructions on the back of the circuit board to connect it to the TI-Innovator Hub; connect the TI-Innovator Hub to your TI-84.

Teacher Tip: The TI-RGB Array device forces students do go a little bit deeper into the world of Object-Oriented Programming (OOP). The TI-Innovator Hub implementation in TI-84 Plus CE Python is designed using Classes (definitions of objects). As already discussed in Units 1, 2 and 3 of this course, the onboard accessories (light, color, sound, brightness) have *objects* already defined in the **ti_hub** module. Example: **light.on()** - light is the name of the object and **on()** is a method or function within that class definition.
All external accessories attached to the TI-Innovator Hub require that the programmer create an instance variable (the object) before the methods of that class can be used. The class name is selected from the TI-Innovator Hub menus, but the methods are *not* on the menus at all. They will 'appear' in the editor as you type (as you will see).

©2021 Texas Instruments Incorporated

1. Start a new Python program using the Hub Project template.

Press **[math] ti_hub… Output devices…** and select the **TI-RGB Array**. This pastes **from rgb_arr import *** into your code.

Again, press **[math]** and select the new **TI-RGB Array…** menu item at the bottom of the list. Select the first option **var = rgb_array()**.

Then, in front of the **=** sign, type any variable name. We use **r**.

*This is Object-Oriented Programming at work: The variable (**r**) is an instance (object) of the **rgb_array** class. You will use methods of the class in the rest of these lessons.*

> **Teacher Tip:** A popular variable name is 'rgb' but that may lead to confusion for some students.

2. To get the lights to light up all at once, on the next line of your program type your variable name followed by a period or decimal point.

r

.

Press **[math] TI-RGB Array…** and select **var.set_all(r, g, b)** so that the method is appended to the variable as shown.

3. Enter the three arguments, each one a value from 0 to 255, to control the color of the LEDs on the RGB Array.

4. <**Run**> your program to see all the LEDs light up in your selected color. Notice that the LEDs remain lit even after the program ends.

5. To turn all the LEDs off at once use the statement

r.all_off( )

found on the **[math] TI-RGB Array** menu.

Add a sleep( ) statement to keep the lights on for a few seconds otherwise you won't see anything happen.

When you **<Run>** the program now you should see the LED come on and then turn off.

6. Put your LED control statements into a **for** loop to make them blink on and off several times. Add another **sleep()** after they are turned off. You may want to adjust the sleep times to speed things up a bit. Be sure to indent all statements in the loop block using **<Tools> Indent ►** on each line.

Note: *Do not* include the *constructor* statement **r = rgb_array()** in the loop block. It only needs to be defined once!

<Run> your program to test it before you continue.

7. If your program blinks all LEDs at once, then you are successful. Now let's control the LEDs one at a time with an inner loop.

(demo1.1.gif)

8. Below the **for i…** loop, add an *inner* loop: **for j in range(16):**
   Be sure to change the loop variable from **i** to **j**.
   Indent all four loop block statements so that they now apply to the *inner* loop.
   Change the statement **r.set_all(r, g, b)** to:

   r.set(j, r, g, b)

   You can *either* erase the .**set_all()** statement and select the **.set()** statement from the menu *or* just edit the statement. **r.set(, , ,)** takes *four* arguments: the LED position and the three color values. Use the inner loop variable **j** as the led_position and enter your three color values. Change the **sleep** values and (optionally) the outer loop **range( )** argument to speed things up a bit.

9. **<Run>** your program. Now your 16 LEDs light up one at a time several times.

(demo1.2.gif)