Bright Lights

This Unit contains a total of 5 activities that all use the TI-Innovator Hub and its built-in devices. Each activity builds on the coding skills learned in the previous activity.

Objectives:

Programming Objectives:

- Use built-in functions from the TI-Hub Library.
- Use RGB colors to change the LED on the TI-Hub.
- Use the brightness sensor to measure brightness.
- Use the random library to generate integers.
- Use for statements to repeat code.
- Use a while statement to repeat code.
- Use lists to store values.

Key AP Computer Science Principles Standards:

- Represent a value with a variable (AAP-1.A)
- Represent a list or string using a variable (AAP-1.C)
- Write conditional statements (AAP-2.H)
- Write nested conditionals (AAP-2.I)
- Select appropriate libraries or existing code segments to use in creating new programs (AAP-3 D)
- Write iteration statements (AAP-2.K)
- Write expressions that use list indexing and list procedures. (AAP-2.N)
- Write iteration statements to traverse a list. (AAP-2.O)
- Write statements to call procedures (AAP-3.A)
- Develop procedural abstractions to manage complexity in a program by writing procedures. (AAP-3 C)
- For generating random values, write expressions to generate possible values. (AAP-3 E)

This document contains Activity 5 of the Bright lights activities.

Activity 5: Importing Colors

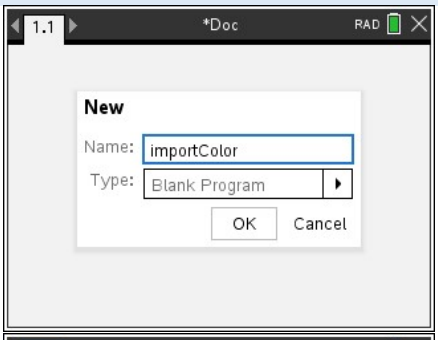
Students use the TI-System library to access values for a spreadsheet.
Students use these values and a loop to set the colors on the LED.

Acitivity 5: Importing Color

Use the TI-System library to access values for a spreadsheet.

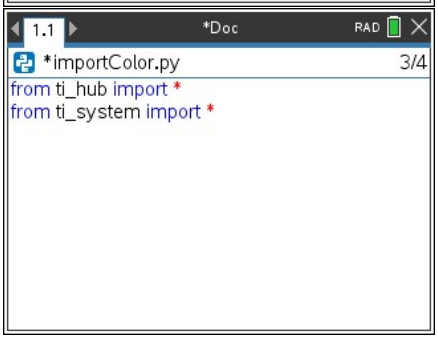
Use these values and a loop to set the colors on the LED.

1. Create a new program named importColor.
   Set the Type to “Blank Program”.



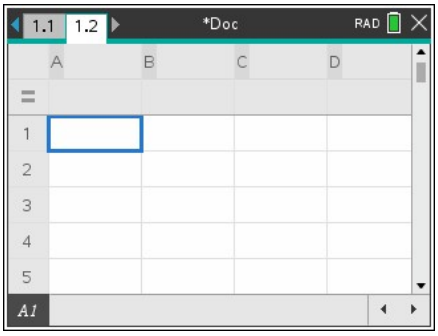
   Import the ti_hub library.

   Import the ti_system library.
   Menu > More Modules > ti_system > from ti_system import *



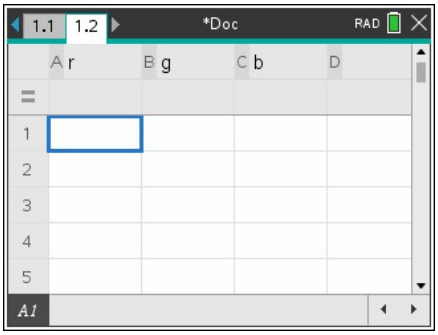
2. Insert a Spreadsheet Page

   There are several ways to add this page.
   Below are three different was to add a page.
   ctrl > i
   ctrl > doc
   doc > insert > Lists and Spreadsheet Page



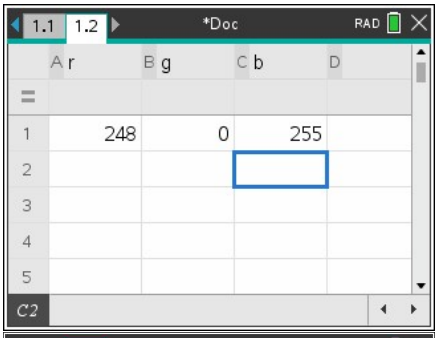
3. You will use the spreadsheet to create three lists: red, green, blue.
   These lists will allow you to easily add pre-set colors to your program.

   Name the three columns. For ease of typing, call them r, g, b.
   Names go in the first row.

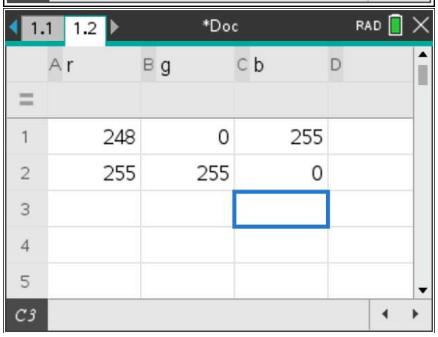


4. Each row will contain the elements for one color.

   Let's put a magenta color in row 1.
   r = 248, g = 0, b = 255
   creates one form of magenta.



5. Add yellow to row 2.
   r = 255, g = 255, b = 0



6. The list r now contains 2 values.
   r[1] = 248 and r[2] = 255

   What value is in b[1]? _____

   What value is in g[2]? _____

   What value is in r[3]? _____

7. Wait, r[3]? Currently there isn't a value in r[3].

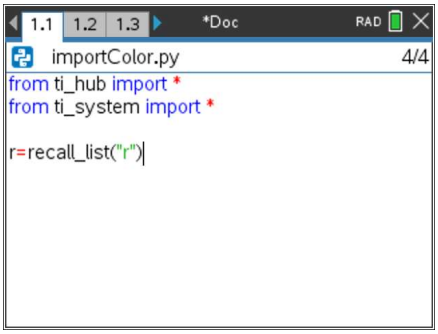
   If you try to use r[3] in code you'll produce an error.
   We'll investigate this potential error later in your code.

8. Add at least 5 more colors to your lists.

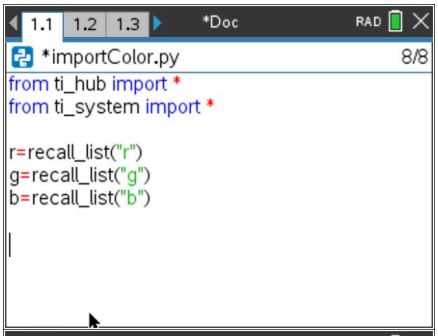
9. Go back to your Python code. You created lists on the spreadsheet; however, it is not part of your code. You need to import these lists into your project.

   Go to:
   Menu > More Modules > Ti_System > recall_list

When you recall a list, you need to store it in a new variable. You can choose to name this variable anything you want. For ease, the code on the right stores it back in variable r.



10. Import both the green and blue values.

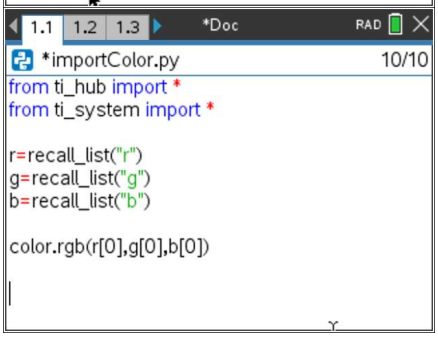


11. Let's verify the colors were imported correctly.
    Add the line:

    color.rgb(r[0], g[0], b[0])

    What color should be displayed when you run your code?_____

    Execute your code. Verify it displays the color you anticipated.
    Hint: It might not be the color you expected! We'll discuss this in the next step.



12. List values on the spreadsheet page start with an index of 1.
    HOWEVER, in python, the index values start at 0!!!

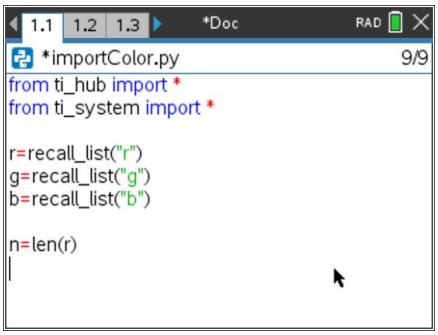
    Therefore, in your python code r[0], g[0], b[0] produces the first item in our list.

    Some languages start with an index of 0, others 1. When learning new languages, it is important to check to see where the index starts in that language.

13. To loop through all the items in the list, you need to know how many items you have. Add the line of code:
    n = len(r)

    This will find and store the length of the list.

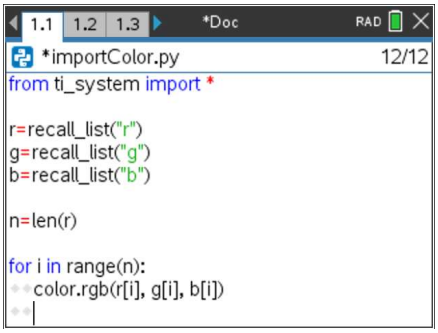
    Menu > built-ins > lists > len



14. The variable n stores the exact number of times you need to change colors. Which type of loop is best suited for executing a set number of times?

15. To cycle through the colors, the code will be:

```
for i in range(n):
    color.rgb(r[i], g[i], b[i])
```



16. Execute your code. It looks like it only displayed the last color.

Why? What do you need to add to your loop?

Add that one line of code.

Execute your code and verify it loops through all the colors.

Teacher Notes:
Students need to add a sleep(1). This will display each color for each second

17. Now that you have a list established on the spreadsheet page, you can easily add or remove colors to your program. Add three more colors to your lists on the spreadsheet pages.

Execute your program. Verify it loops through all the colors.
Did your new colors appear at the end?

18. Now to modify your code. How can you randomly select and display a new color each for 1 second? Modify your code so it will:

Repeat the while loop until the user presses “esc”:
generate a random integer within the length of the list
change the color to that randomly generated number
pause for a second

Hint: Don’t forget to import the random library
Hint: If you have 10 items in the list, the indexes will be from 0 to 9. Your code will need to use the variable n to generate your index value. Don’t present your random values based on the current number of items in your list. You want your code to run properly if the list size changes. If you get a get the error “list index out of range”, you generated a value too large.

Teacher Notes:
Possible Solution:

```
from ti_hub import *
from ti_system import *
from random import*

r=recall_list("r")
g=recall_list("g")
b=recall_list("b")

n=len(r)

while get_key() != "esc":
  i=int(random()*n)            #student could also use i=randint(0,n-1)
  color.rgb(r[i], g[i], b[i])
  sleep(1)
```

Teacher Notes:
Extensions for Activity 5:
1.) Add a fourth column to the spreadsheet. Name this column t for tone. For each rgb color in the spreadsheet, add a tone frequency between 300 and 800. Mofidy your Python code from Activity 5 to play the corresponding tone with the color each time through the loop.
2.) Modify your code from Activity 5. Add code to read the brightness level of the room. Based on the brightness level reading, use if..elif..else statements to determine which color to display from the list.