



#### Unit 2: for loops with the TI-Innovator Hub

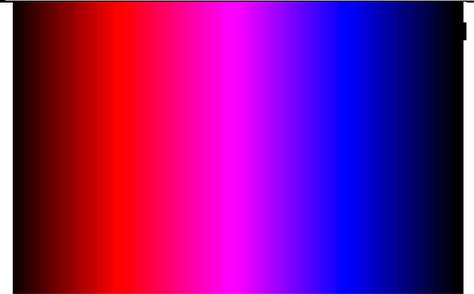
#### Skill Builder 2: Loop Through Color

In this lesson, you will learn about color mixing to make a variety of colors on the color LED using just several **if** statements in a loop.

#### Objectives:

- Use **if** statements to change the color of the LED
- Use copy line/paste line and edit for simpler coding of longer programs.

Many colors are possible (over 16 million!) using the color LED on the TI-Innovator Hub. This lesson develops a program that gradually changes the color LED from red to blue by mixing changing amounts of each color.



1. Start a new Python Hub Project (ours is called COLORS) and add **import color** from `[math] ti_hub....`

*Tip: you can also easily type this statement (and any others) manually using the <a A #> Character Map.*

This program will gradually increase and decrease two of the LED's three color channels to mix some colors.

Start by assigning values to two variables: **step** is the space (increment) between color values and **delay** is the time delay for each color step. Begin with a step of 10 and a delay of 0.1 at first. Type the words **step** and **delay** or use simpler variables like **s** and **d**.

Assign 0 to two variables, **r** and **b**, representing the starting values for the red and blue channels.

```

EDITOR: COLORSV2
PROGRAM LINE 0001
Hub Project
from ti_system import *
from time import *
import color

step=10
delay=0.1
r=b=0

```

2. Add the loop

### while not escape( ):



from `[math] ti_system...`

so that when you run the program you can press the **[clear]** key to exit at any time.

```

EDITOR: COLORSV2
PROGRAM LINE 0011
# Hub Project
from ti_system import *
from time import *
import color

step=10
delay=0.1
r=b=0

while not escape():
-

```



# 10 Minutes of Code – Python

## TI-84 PLUS CE PYTHON WITH THE TI-INNOVATOR™ HUB

### UNIT 2: SKILL BUILDER 2

### STUDENT ACTIVITY

- Begin the loop block by turning the LED on using the current values of **r** and **b** for the red and blue channels. We will not use the green channel at all. These variables will change value further down in the loop.

#### ◆ ◆ `color.rgb(r, 0, b)`

Then use the `delay` variable in the `sleep()` function found on [math] time...

#### ◆ ◆ `sleep(delay)`

- We will write four `if` structures to change the values of **r** and **b**. Each will gradually change one color channel.

The first `if` structure increases the value of **r** when **b** is 0.

`r += step` means the same as `r = r + step`

- Next check to make sure **r** does not exceed 255. If it does, then limit **r** to 255 and start to increase **b**. This gradually adds blue to the bright red LED.

- Do the same with the value of **b**: when **b** exceeds 255 we limit it to 255 and then decrease the value of **r**. This removes red gradually from the mixed colors (bright magenta).

```
EDITOR: COLORSV2
PROGRAM LINE 0014
r=b=0

while not escape():
    color.rgb(r,0,b)
    sleep(delay)
```

```
EDITOR: COLORSV2
PROGRAM LINE 0016

while not escape():
    color.rgb(r,0,b)
    sleep(delay)
    if b==0:
        r+=step
```

```
EDITOR: COLORSV2
PROGRAM LINE 0018

while not escape():
    color.rgb(r,0,b)
    sleep(delay)
    if b==0:
        r+=step
    if r>=255:
        r=255
        b+=step
```

```
EDITOR: COLORSV2
PROGRAM LINE 0012

sleep(delay)

if b==0:
    r+=step
if r>=255:
    r=255
    b+=step
if b>=255:
    b=255
    r-=step
```



- Finally, when **r** reaches (or passes, depending on **step**) 0, limit it to 0 and begin decreasing the value of **b**. This gradually darkens the blue LED

We now have the four if structures shown to the right. Again, think carefully about the effect of each.

**<Run>** the program and watch the color LED brighten to red, then change to magenta (purple) and get pretty bright, then gradually change to blue and then...

```

EDITOR: COLORSV2
PROGRAM LINE 0014
if b==0:
r+=step
if r>=255:
r=255
b+=step
if b>=255:
b=255
r-=step
if r<=0:
r=0
b-=step

```

- When the blue value decreases, it eventually takes on a value less than 0 and then the program tries to use that value in the **color.rgb( )** function which creates the error shown here.

When decreasing **b** from 255 by 10's (our **step** value) we encounter -5 which causes the error. There are several ways to correct this runtime error. Try it yourself before seeing two suggestions in the next step.

```

PYTHON SHELL
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "COLORSV2.py", line 25, in <module>
  File "color.py", line 86, in rgb
ValueError: Value out of range 0 to 255.
>>> |

```

- One fix is:
  - only allow a non-negative value for **b** when executing the **color.rgb()** function:

```

if b >= 0:
  ♦ ♦ color.rgb(r, 0, b)

```

*Tip: to indent a statement use the <Tools> menu.*

- and modify the first **if** statement to include negative values for **b** in the condition:

```

♦ ♦ if b <= 0:

```

- and assign 0 to **b** inside this block:

```

♦ ♦ ♦ ♦ b = 0

```

Now all four **if** structures look pretty much the same!

```

EDITOR: COLORSV2
PROGRAM LINE 0012
step=10
delay=0.01
r=b=0

while not escape():
if b>=0:
color.rgb(r,0,b)
sleep(delay)
if b<=0:
r+=step
b=0

```

*Another possible fix is to just add **if b<0: b=0** right before the **color.rgb( )** function (not shown)*



## 10 Minutes of Code – Python

TI-84 PLUS CE PYTHON WITH THE TI-INNOVATOR™ HUB

UNIT 2: SKILL BUILDER 2

STUDENT ACTIVITY

10. Try various values for step and delay. A large step and a small delay make the program run faster.

**Challenge:** can you incorporate the green channel, too? You can see all the colors of the rainbow if you mix the colors properly.

