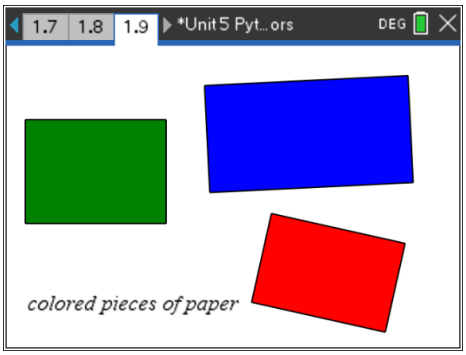
Unit 5: Rover's Sensors — Skill Builder 3: Spot the Color

In this lesson, you will learn to use Rover's color sensor to change direction when a color is detected. Colored paper or large colored shapes are needed for this lesson.

Objectives:

- Use the color sensor to detect and react to a color

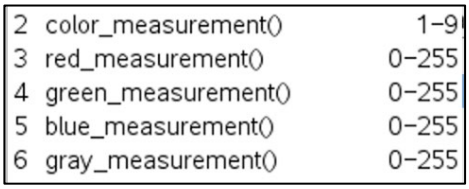


Rover has a color sensor on the bottom. You can see a light shining on the floor under the color sensor. The light helps Rover to see the color beneath it. First write a 'test' program to see what kind of values the color sensor produces and then you will be able to write a program to react to different colors. You will need some colored paper like construction paper or just print out some colored shapes like the rectangles at the right. They should be large enough for Rover to 'see'.

On the **menu > TI-Rover > Inputs**, there are five different color measurements available. The function **color_measurement()** returns a value from 1 to 9 where:

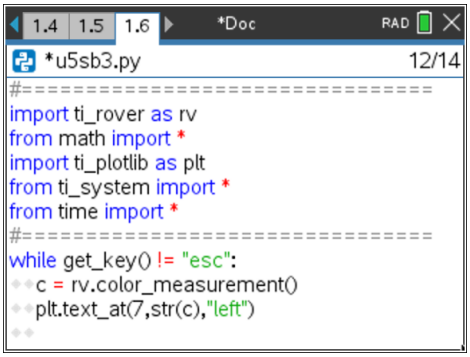
1=red, 2=green, 3=blue, 4=cyan, 5=magenta, 6=yellow, 7=black, 8=white, 9 = gray

The other four measurements return the amount of the indicated color in the range 0…255, as shown on the menu.



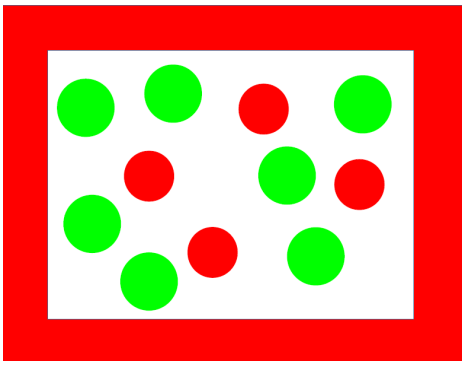
1. Here is a short 'test' program using the Rover Coding template to determine the values that the color measurement functions produce.

```
while get_key() != "esc":
    c = rv.color_measurement()
    plt.text_at(7, str(c), "left")
```



   Try all five color measurements (**color_**, **red_**, **green_**, **blue_**, and **gray_**) on various colored surfaces and observe the values displayed.

2. Depending on your floor color, make some colored pieces of paper (like 'sticky notes' or colorful construction paper) to place in front of Rover so that Rover can 'see' the change in color from the floor. Test these patches first to see what Rover 'sees'.



Write a program to get Rover to 'react' to the difference in colors. In the sample page to the right, the red border should keep Rover on the page:
When Rover 'sees' red, turn around.
When Rover 'sees' green, turn right or left.

You can also add statements to the program to control the light or color LED.

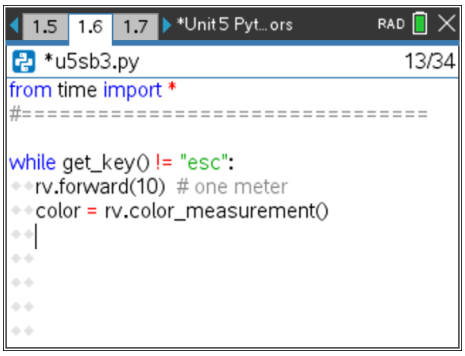
3. Edit the color 'test' program you started. Add a statement at the beginning of the loop to start Rover moving forward:

rv.forward(10)

Next, monitor the color below Rover. Changing the variable **c** to **color** makes the program more readable:

color = rv.color_measurement()

When **color** is red, make Rover turn 180 degrees and, when the color is green, make Rover turn 90 degrees. This requires two **if** statements.

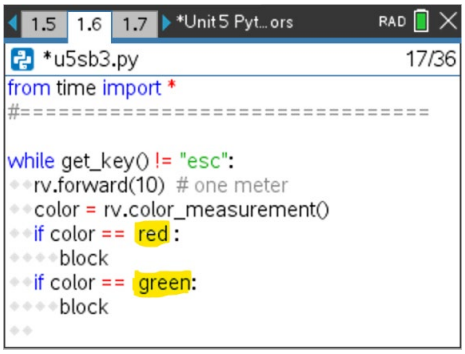


Delete **plt.text_at(7, str(c), "left")**.

4. Add two **if** statements and use the values you got from testing your **color** samples in place of the **?**s:

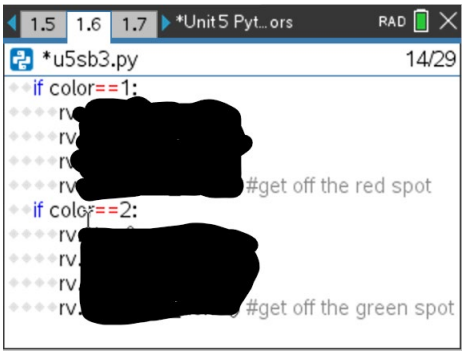
if color == ?:
block
if color == ?:
block

(You will not see the placeholder 'block' appear for the **if** statement and the words 'red' and 'green' in the image are just placeholders.)



5. The two **if** blocks are similar. Just remember red-right and green-left.
Each block will:
   a) Stop Rover
   b) Turn 180 degrees
   c) Go forward a little bit to move away from the colored spot
   d) Tell the TI-Nspire CX II to *wait until* Rover has *finished* these three tasks

Try it yourself before proceeding…



6. Do your **if** blocks resemble this?

```
if color == 1:
  rv.stop()
  rv.right(180)
  rv.forward(1)
  rv.wait_until_done()
```

(*Red; use another number if your color is not red.)*

Tip: If you assign the variable **red = 1** *at the top of your program, then you can write* **if color == red:** *in your* **if** *statement. This makes the intent clear.*