Scribbler Test

For all of these exercises, please save your python code (as separate .py files). So, each time you get to a new question (or sub question with a different program), create a new file in Calico and save it as the question name (for example, "2b.py" would be a program to draw a right triangle with the robot, and "2c.py" would be a program to draw a right triangle in the graphics window.

---

1. Combine the 360 GIF sample program with other programs such as line following or flashlight programming to have the robot wait to start taking pictures until either the IR sensor is blocked or a light is shined into the light sensor. Sample programs that you may want to see:
https://raw.github.com/CSavvy/python/master/360_gif.py,
https://raw.github.com/CSavvy/python/master/light_alive.py,
https://raw.github.com/CSavvy/python/master/light_path.py

---

2. Open the Calico Graphics documentation page (http://calicoproject.org/Calico_Graphics) and search (ctrl+f) for "line." You'll need to be able to draw lines on a graphics window later. Remember to search for other functions using words you think are useful (for example, if you want to draw a circle search for "circle") when they come up. For the scribbler robot, open the Calico Myro documentation page (http://wiki.roboteducation.org/Calico_Myro).

a. Look at the robot_joystick.py program, it shows some graphics functions and has some basics of what you'll need in the following questions. Run it on your robot just to see what it's like. (https://raw.github.com/CSavvy/python/master/extras/robot_joystick.py)

b. Make a program to draw a **right** triangle with the robot (the size of the triangle doesn't matter).

c. When you've made it with the robot, make another program to draw a **right** triangle in a graphics window.

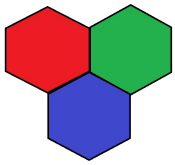
d. Make a program to draw an **isosceles** triangle with the robot (the size of the triangle doesn't matter).

e. When you've made it with the robot, make another program to draw an **isosceles** triangle in a graphics window.

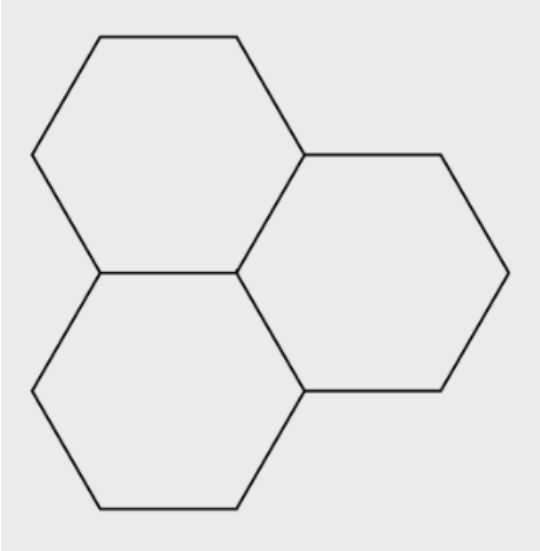
---

3. Now you'll draw a more complex shape. Remember to search through the Calico documentation for a function that would be useful here.

a. Draw 3 adjacent hexagons with the Scribbler (just the outline of the hexagons - no need to fill in the color). Below is a picture of a larger honeycomb so you can get an idea for what you need to do.



b. Use a graphics window to draw the start of a honeycomb pattern (just draw 3 adjacent hexagons, colored if you can, but outline is ok). If you look at the documentation here (http://calicoproject.org/Calico_Graphics#Shape_Methods), you can use code that is very (!) similar to the Scribbler code to draw the honeycomb (using find and replace is very helpful). You'll need to change some things, but the main work is already done. An example is below:



---

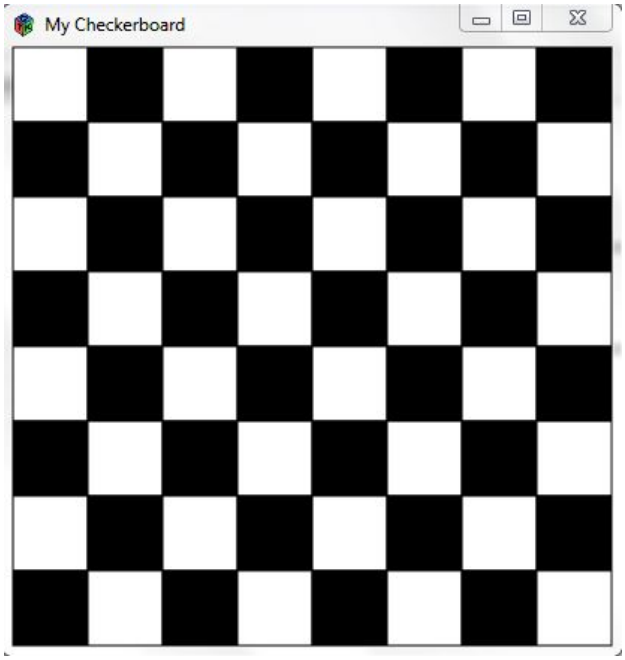
4. See the example of using the % (mod) operator below (run it in Calico and play with it for understanding). Basically, % gives you the remainder from division, so 7%5 is 2, and 5%5 is 0.

```
# Prints multiples of 5
for i in range(50):
  if( i % 5 == 0 ):
     print(i)
```

Another example: This makes a multiplication table in a Window. Run it and see what it's like. It even colors the text like a checkerboard.

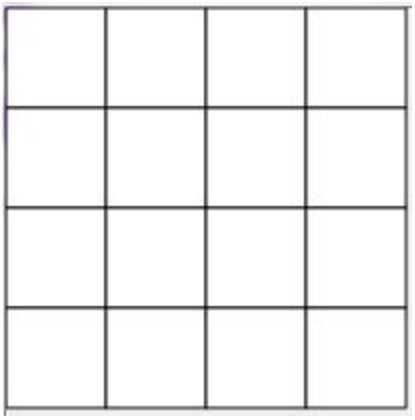
```
from Graphics import *

win = Window("Multiplication Table", 300, 650)

for x in range(1, 5):
   for y in range(1, 12):
      text = Text((20 + x*50, 20 + y*50), str(x*y))
      if ((x + y)%2 == 0):
         text.setFill(Color("red"))
      text.draw(win)
```

a. Draw a checkerboard in graphics using a for loop. It should be 8x8, and look like the picture below:



b. For the robot, draw a series of squares (using a for loop) to outline a 4x4 checkerboard

pattern. It doesn’t need to be perfect — it’s the code that matters.



Done!