Computer Science 1	Lab 18A 1-Day Minor Python Assignment
Simple Paint Program	90, 100 and 110 Point Versions

Assignment Purpose:

The purpose of this program is to demonstrate understanding of *Mouse Interactivity* and *Key Interactivity* in a graphics program.

For your penultimate project you will be combining *Turtle Graphics*, *Traditional Graphics*, *Mouse Interactivity* and *Key Interactivity* to create a very simplistic Paint Program. The main structure of the program, including the entire **drawMenu** procedure, has already been written for you.

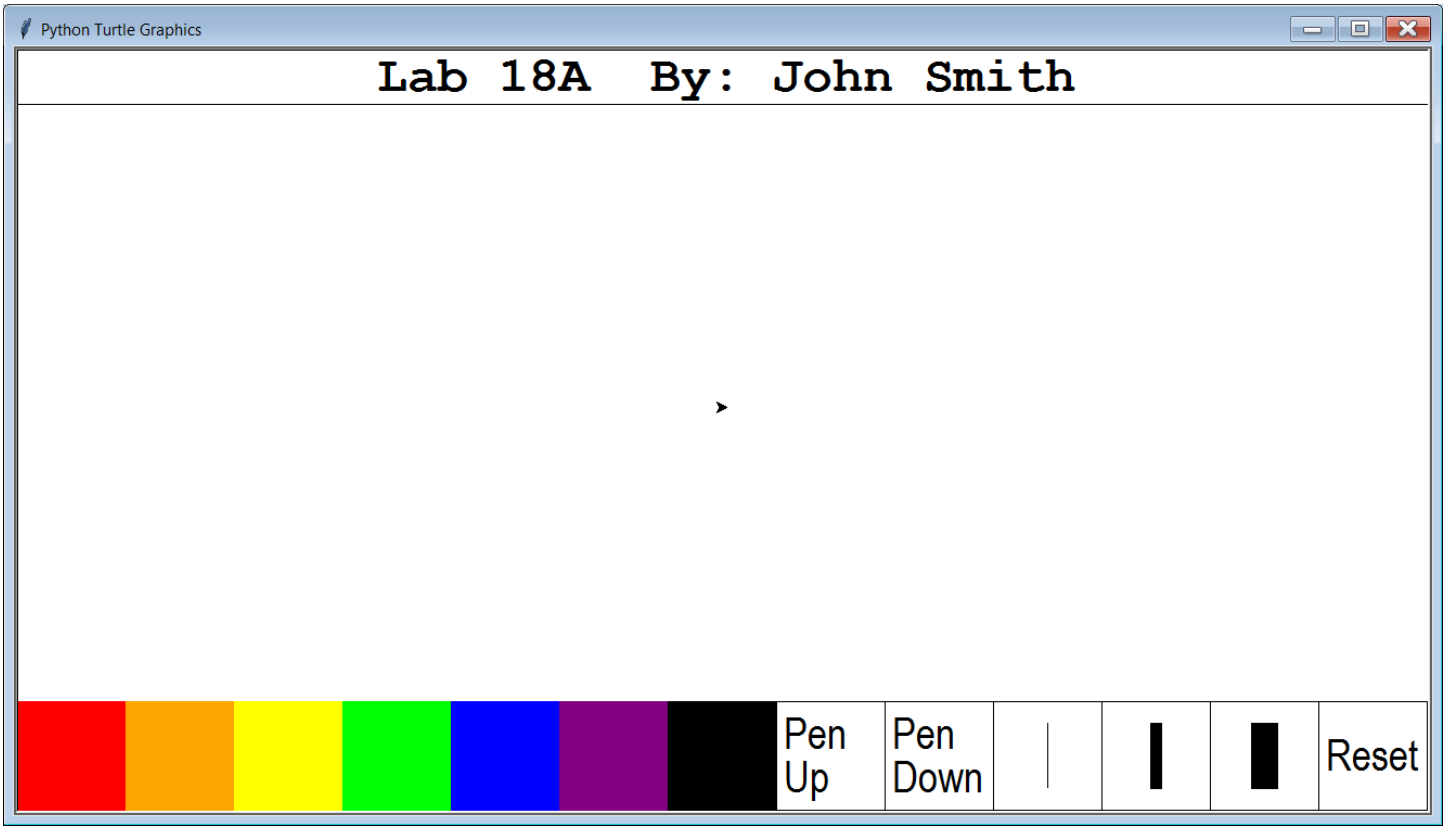
Lab 18A Student Version	Do not copy this file, which is provided.

```
 1 # Lab18Ast.py
 2 # "Simple Paint Program"
 3 # This is the student, starting version of Lab 18A.
 4
 5
 6 from Graphics import *
 7
 8
 9 def drawMenu(): # This procedure is complete.  Do not alter it!
10     reset()
11     drawHeading("John Smith","18A")
12     setColor("red")
13     fillRectangle(0,600,100,700)
14     setColor("orange")
15     fillRectangle(100,600,200,700)
16     setColor("yellow")
17     fillRectangle(200,600,300,700)
18     setColor("green")
19     fillRectangle(300,600,400,700)
20     setColor("blue")
21     fillRectangle(400,600,500,700)
22     setColor("purple")
23     fillRectangle(500,600,600,700)
24     setColor("black")
25     fillRectangle(600,600,700,700)
26     drawRectangle(700,600,800,700)    # pen up
27     drawString("Pen",708,653,"Arial Narrow",24,"normal")
28     drawString("Up",708,693,"Arial Narrow",24,"normal")
29     drawRectangle(800,600,900,700)    #pen down
30     drawString("Pen",808,653,"Arial Narrow",24,"normal")
31     drawString("Down",808,693,"Arial Narrow",24,"normal")
32     drawRectangle(900,600,1000,700)   # thin
33     drawLine(950,620,950,680)
```

```
    drawRectangle(1000,600,1100,700) # medium
    fillRectangle(1045,620,1055,680)
    drawRectangle(1100,600,1200,700) # thick
    fillRectangle(1138,620,1162,680)
    drawRectangle(1200,600,1300,700) # reset
    drawString("Reset",1208,673,"Arial Narrow",24,"normal")
    update()
    penup()
    goto(0,25)
    showturtle()
    speed(1)
    pendown()


def settings(x,y):
    if inside(x,y,0,600,100,700):
        setColor("red")
    #
    # Several elif commands need to be inserted here.
    #
    elif inside(x,y,1200,600,1300,700): # Reset
        drawMenu()

    update()


def moveUp():
    setheading(90)
    forward(100)
    update()



##########
#  MAIN  #
##########

beginGrfx(1300,700)
drawMenu()
listen()
onscreenclick(settings,btn=1)
onkey(moveUp,"Up")



update()
done()

```

Current Output of Lab18Ast.java

At the beginning, the output demonstrates the “Edit Window” of the Paint Program along with a menu of “buttons” at the bottom of the screen.



90-Point Version Requirements

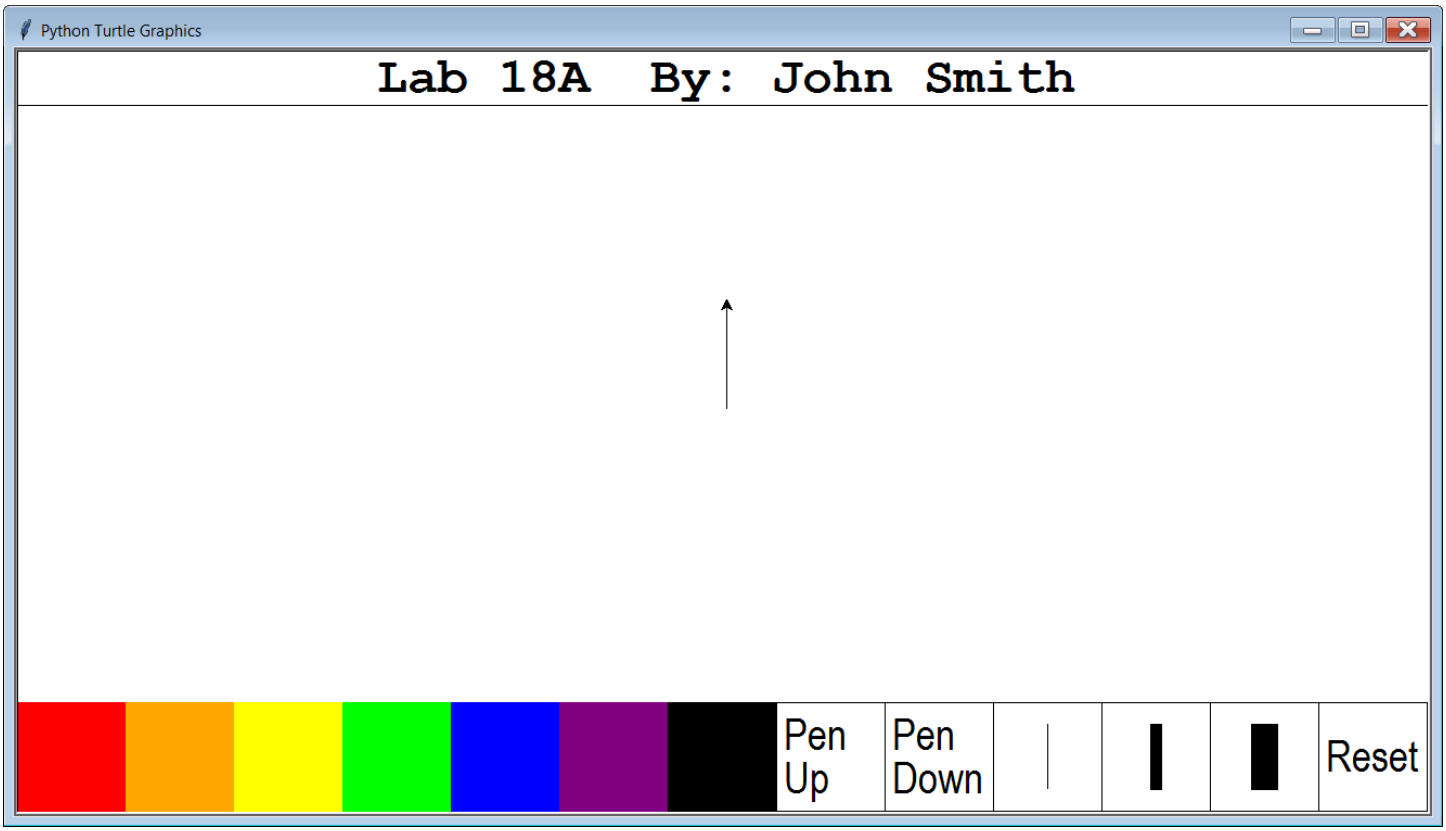
There are 2 main requirements for the 90-point version: Arrow Functionality and Button Functionality.

Arrow Functionality

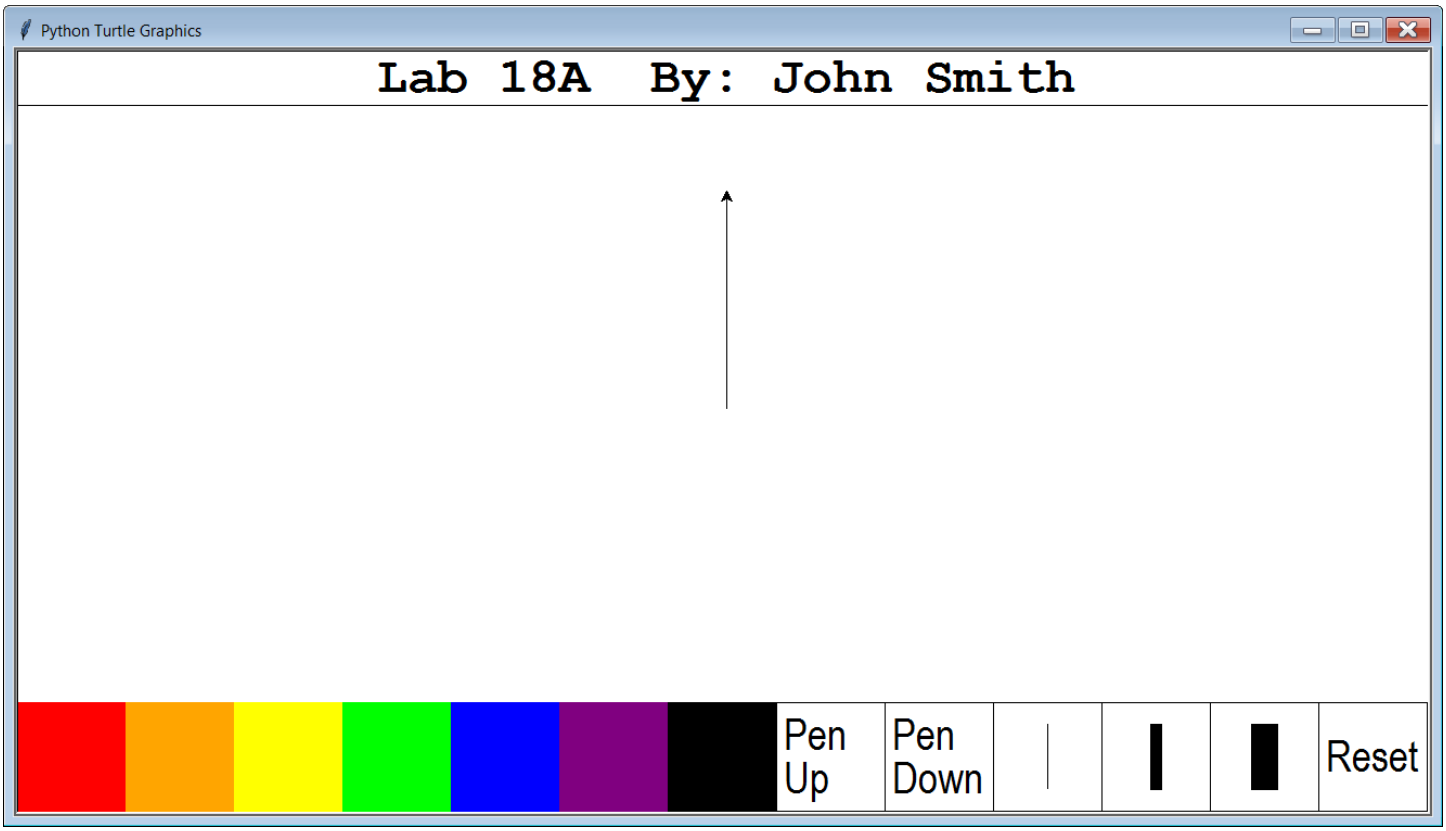
When the user presses any of the 4 arrow keys, the *turtle* needs to move **forward** 100 pixels in that arrow’s direction. This means pressing the <Up Arrow> makes the turtle move *up* 100 pixels; pressing the <Down Arrow> makes the turtle move *down* 100 pixels; pressing the <Left Arrow> makes the turtle move *left* 100 pixels; and pressing the <Right Arrow> makes the turtle move *right* 100 pixels.

To help you, the code necessary to make the <Up Arrow> key functional is provided in the starting file. You need to make the other 3 arrows keys functional on your own. You will notice that the provided **moveUp** procedure uses the **setheading** command, which before now we have not yet used. Part of this assignment is figuring out how to make **setHeading** work for the other 3 directions.

This is the output after pressing the <Up Arrow> key once:

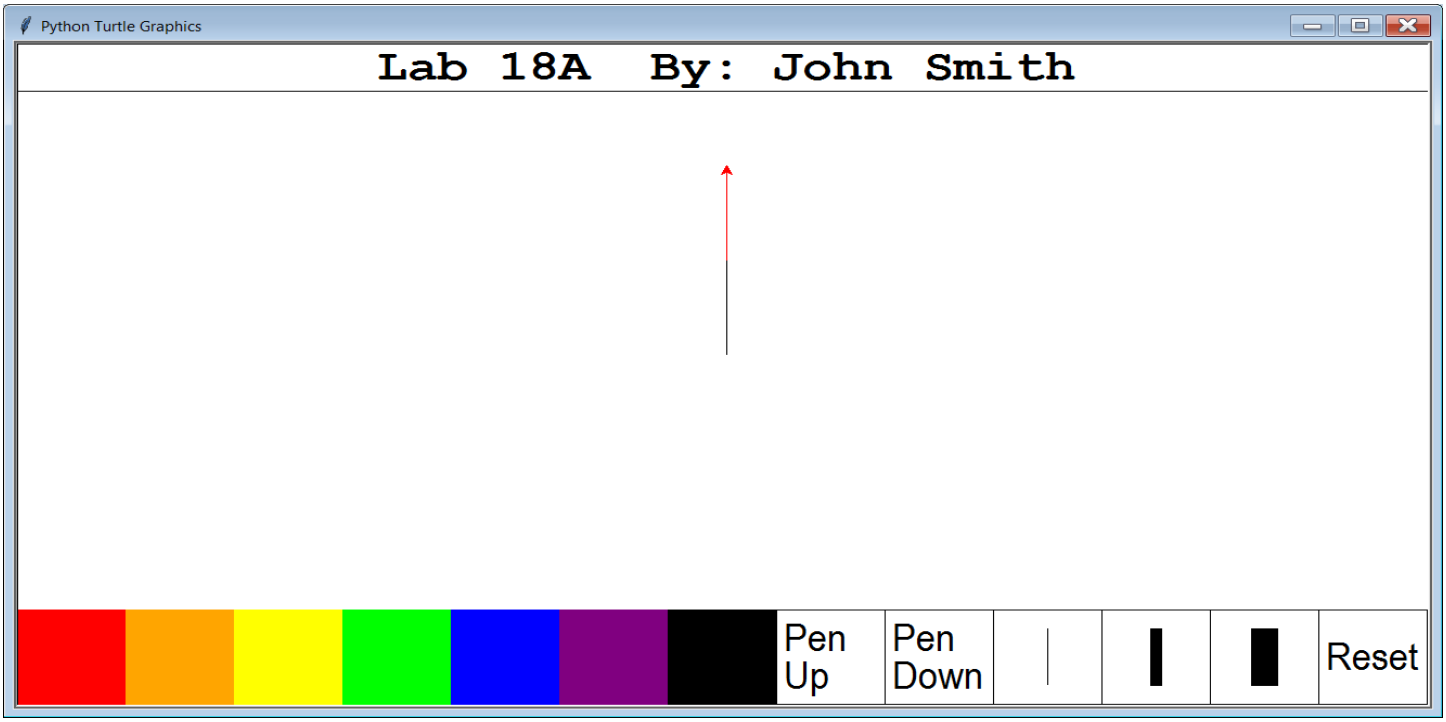


This is the output after pressing the <Up Arrow> key a second time:

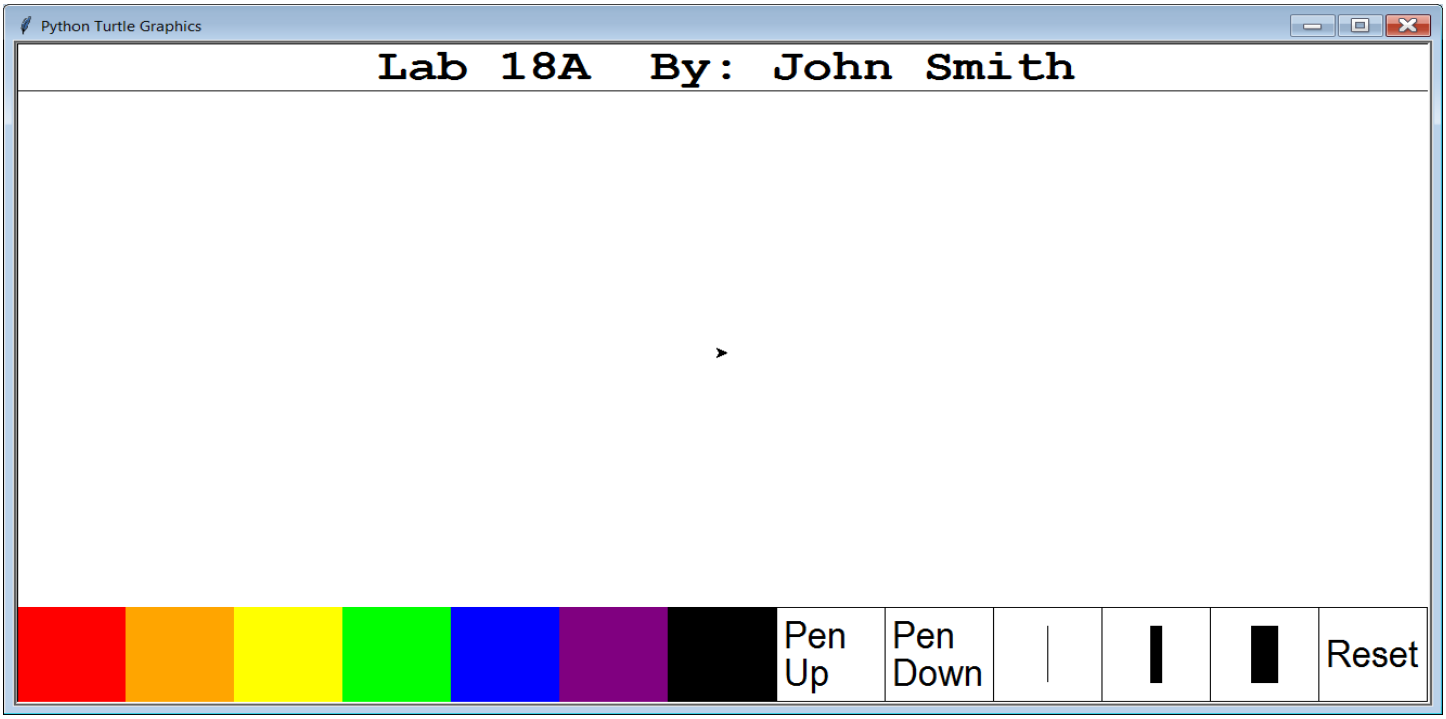


Button Functionality

When the user clicks on any of the "buttons" at the bottom of the window, the appropriate action needs to take place. To help you, the code necessary to make the left-most **red** "button" and the right-most **Reset** "button" functional is provided in the starting file. Below is the output after pressing the <Up Arrow> once, then clicking the **red** "button" and then pressing the <Up Arrow> a second time:



Note that the bottom half of the line is **black**, which is the default color at the beginning of the program, while the top half of the line is **red**, which became the new color after the **red** "button" was clicked. Below, you see the output after the **Reset** button is clicked.



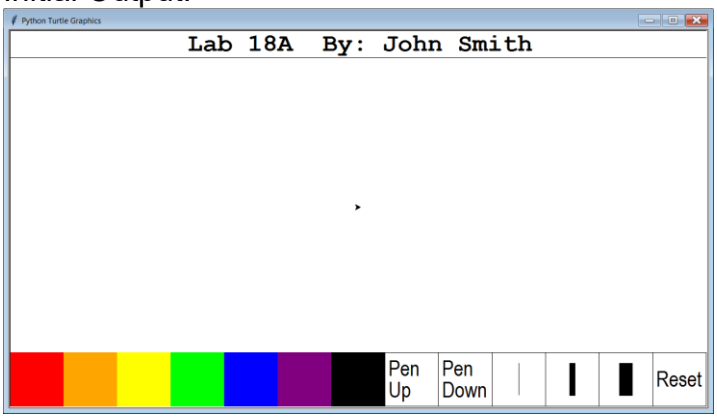
100 and 110-Point Version Requirements

There are 2 addition features that you can include. Each will earn you a 10-point bonus. Do either to earn a 100. Do both to earn a 110.

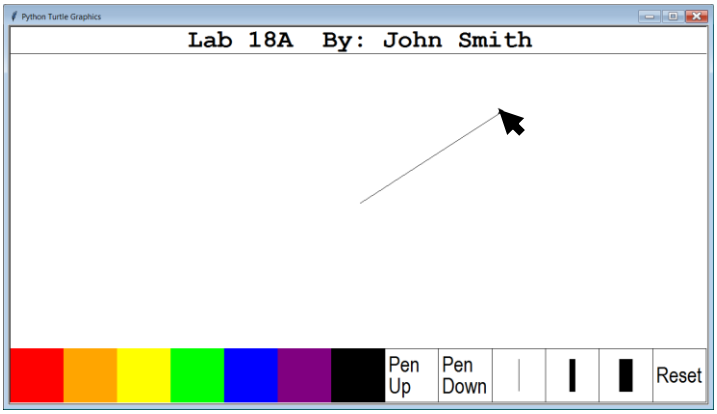
10-Point Bonus #1

When the user clicks anywhere in the edit window, a line should be drawn from the previous location to the current location. See the example below:

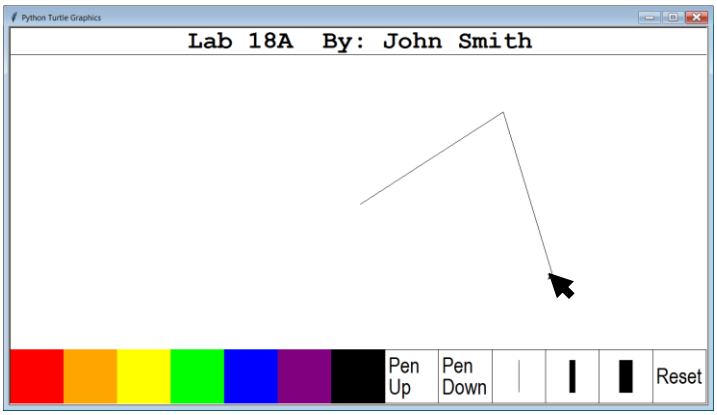
Initial Output:



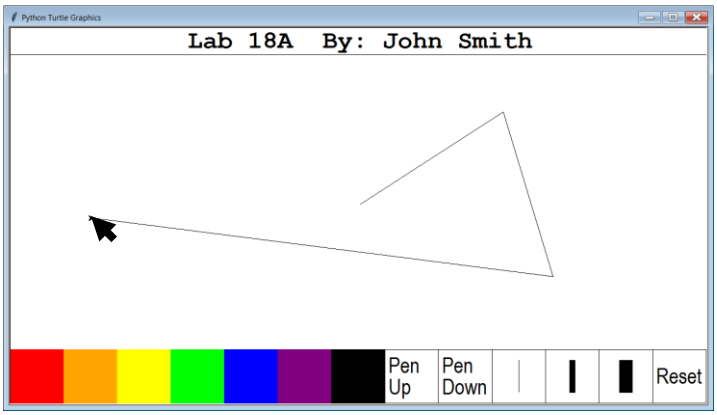
First Click:



Second Click:



Third Click:

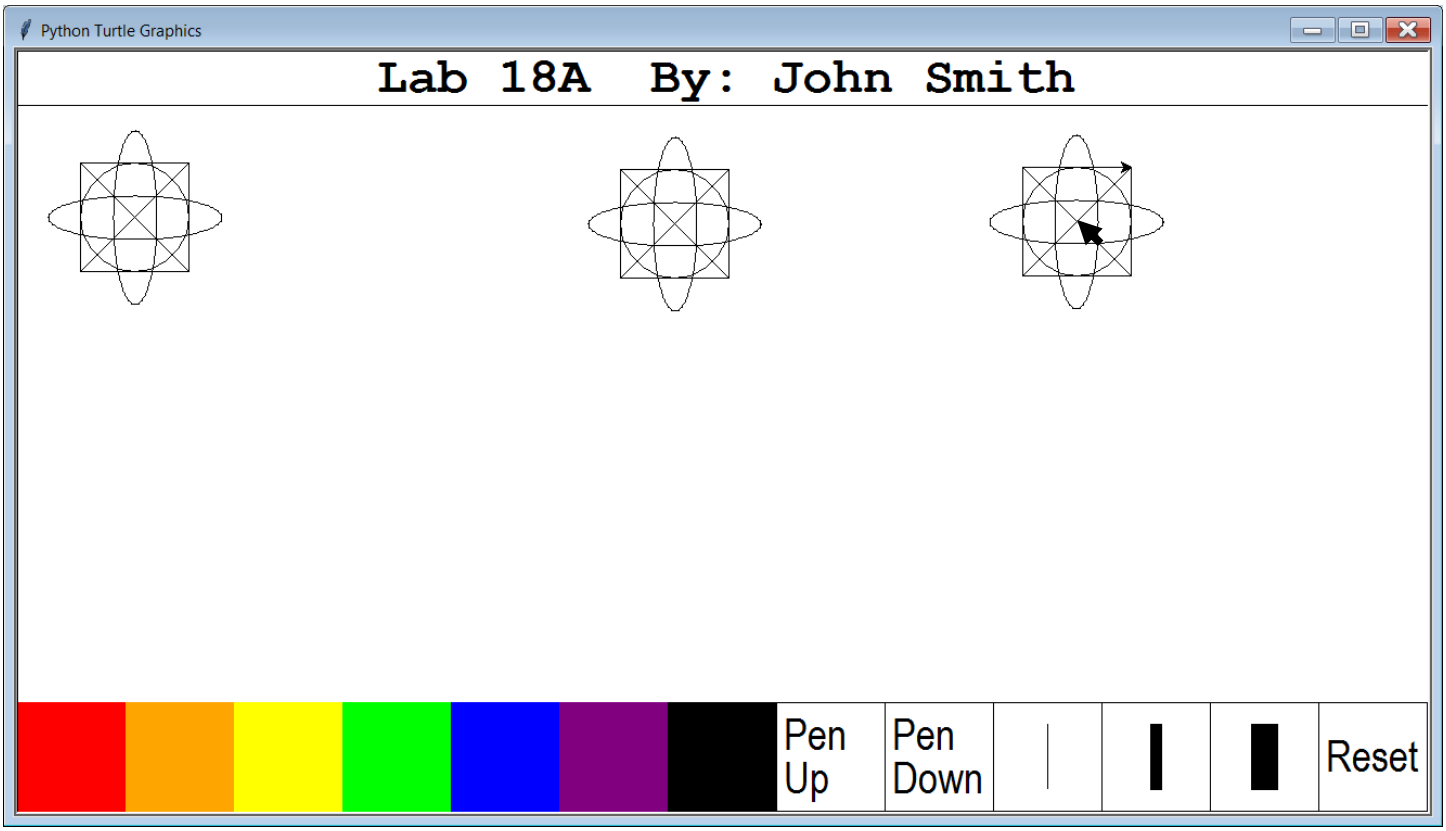


10-Point Bonus #2

When the user RIGHT-clicks anywhere in the edit window, a special design is displayed at the click location. Examples are shown on the next page.

NOTE:
The special design shown on the next page is just an example. You need to create an ORIGINAL design of YOUR OWN CREATION. To get credit, your design cannot match anyone else's.

This is the output after RIGHT-clicking 3 times in the edit window with the default **width**:



This is the output after right-clicking 2 more times with medium **width** & 1 more time with max **width**: