Exposure CS 2021 for CS1

Chapter 18 Section 1-5 Slides

More Graphics:
Mouse Events & Key Events

PowerPoint Presentation
created by:
Mr. John L. M. Schram
and Mr. Leon Schram
Authors of Exposure
Computer Science

Section 18.1

Introduction

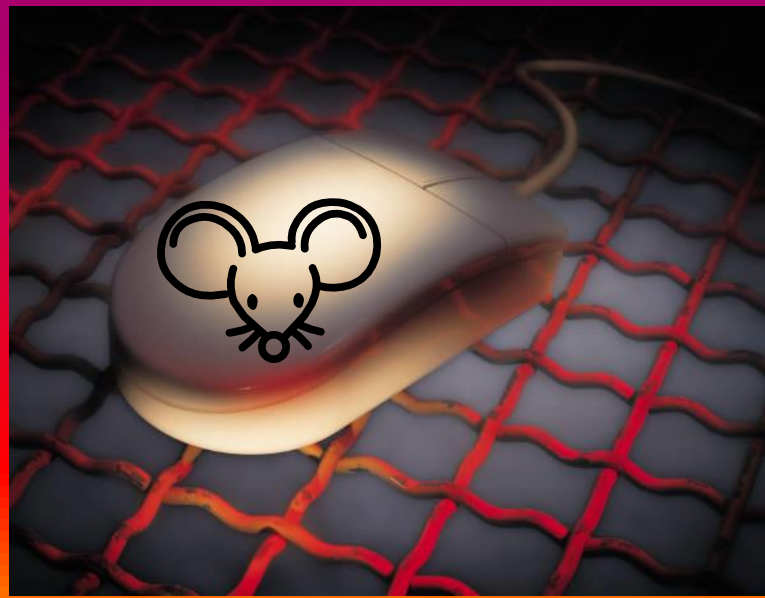
The Last Chapter is Special!

In this chapter you will learn more advanced graphics features. These features will not be as advanced as those taught in AP® Computer Science-A or Advanced Graphics, but they still will allow you to create more impressive graphics programs.

There will be no homework exercises for this chapter. There will also be no quizzes and no test. The information covered in this chapter will not even be part of the Final Exam. However, there are 2 lab assignments – 1 minor lab and 1 major lab – that are based on the information in this chapter. At John Paul II High School the major lab counts as the 4th Quarter Project.

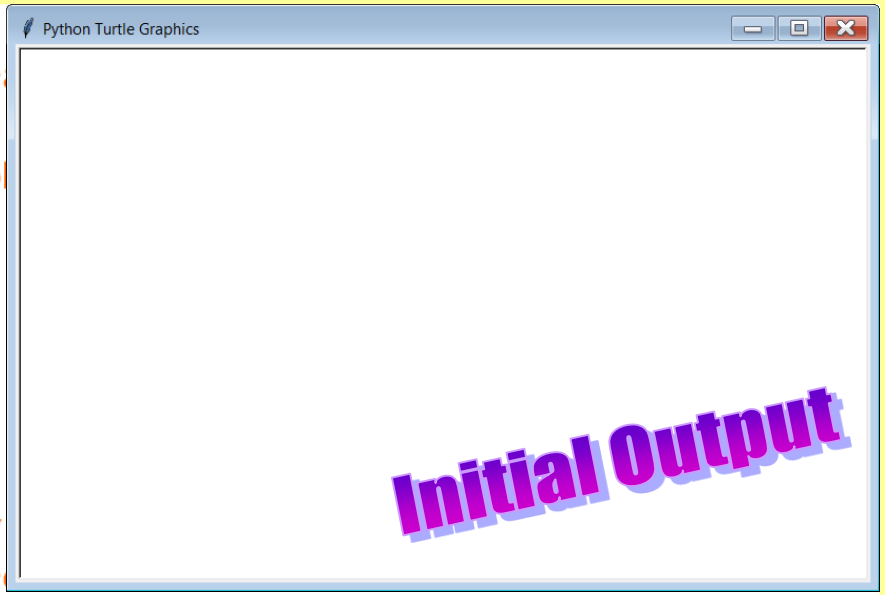
Section 18.2

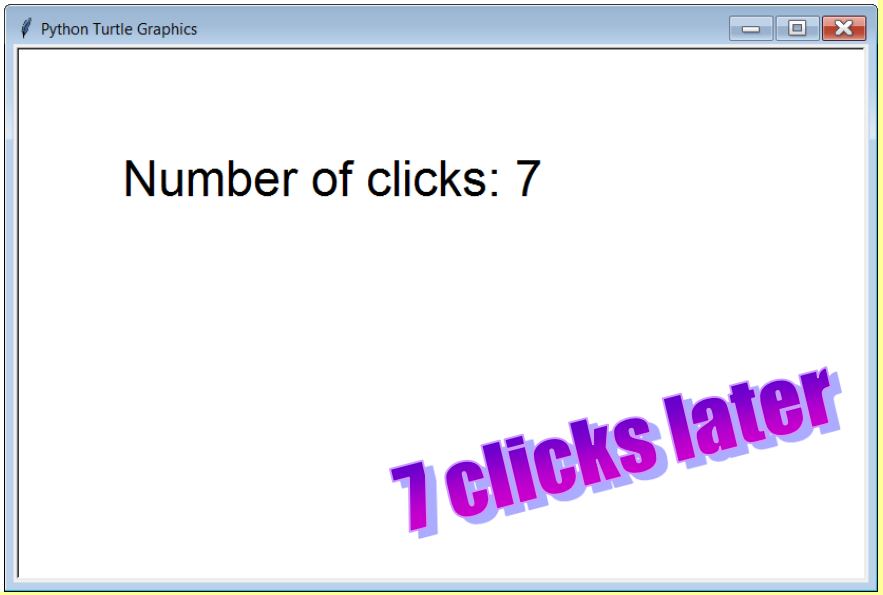
Mouse Events



```
1  # MouseEvents01.py
2  # This program introduces "Mouse Interactivity"
3  # with the <onscreenclick> event which is triggered
4  # any time the user clicks on the graphics screen
5  # with the left mouse button.
6
7
8  from Graphics import *
9
10
11 def display(x,y):     # Parameters (x,y) are required
12     global numClicks  # even if they are not used.
13     clear()
14     numClicks += 1
15     drawString("Number of clicks: "+str(numClicks),100,150,"Arial",28,"normal")
16     update()
17
18
19
20 ##########
21 #  MAIN  #
22 ##########
23
24 numClicks = 0
25 beginGrfx(800,500)
26 onscreenclick(display)
27 endGrfx()
```

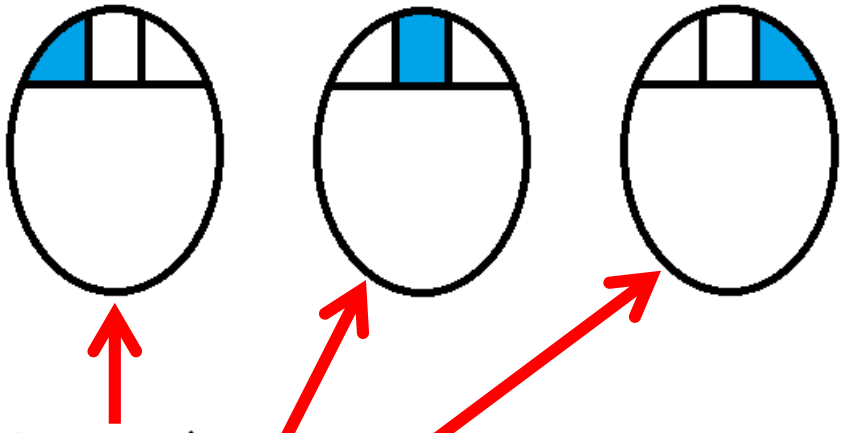
```
1  # MouseEvents01.py
2  # This program introduces "Mouse Inter
3  # with the <onscreenclick> event which
4  # any time the user clicks on the grap
5  # with the left mouse button.
6
7
8  from Graphics import *
9
10
11 def display(x,y):     # Parameters (x,y
12     global numClicks  # even if they ar
13     clear()
14     numClicks += 1
15     drawString("Number of clicks: "+str(numClicks),100,150,"Arial",28,"normal")
16     update()
17
18
19
20 ##########
21 #  MAIN  #
22 ##########
23
24 numClicks = 0
25 beginGrfx(800,500)
26 onscreenclick(display)
27 endGrfx()
```



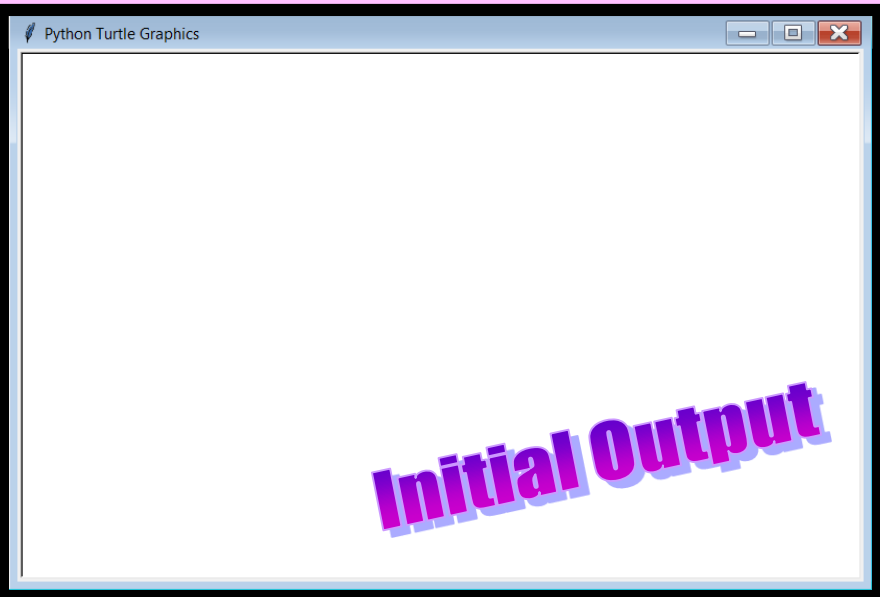


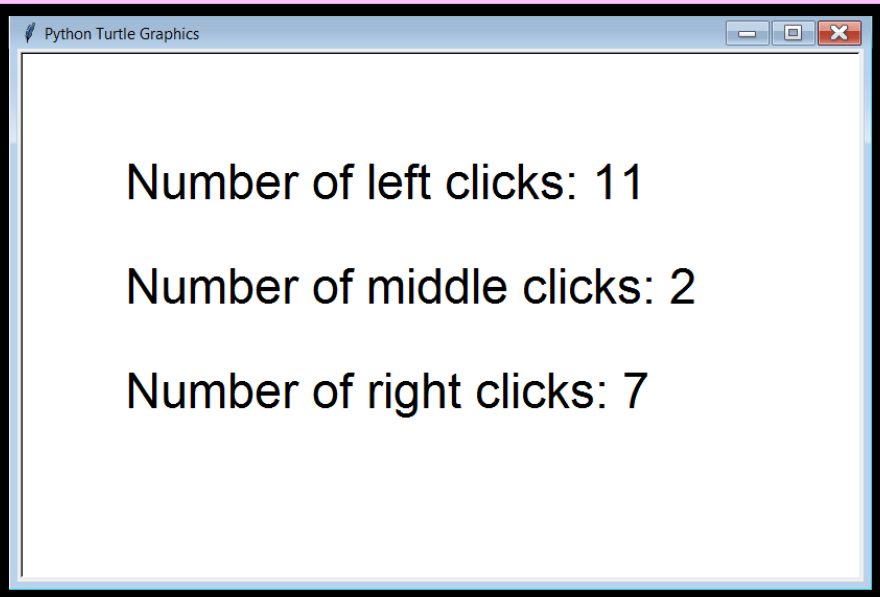
```
# MouseEvents02.py
# This program demonstrates how to distinguish between
# the left, middle & right mouse buttons on a Windows PC.
# NOTE: Some mice have a "middle" button; most do not.
# ALSO: On some mice, the "wheel" is clickable and is
#       actually interpreted as the middle button.


from Graphics import *


def leftDisplay(x,y):
    global numLeftClicks
    clear()
    numLeftClicks += 1
    drawString("Number of left clicks: "+str(numLeftClicks),100,150,"Arial",28,"normal")
    drawString("Number of middle clicks: "+str(numMiddleClicks),100,250,"Arial",28,"normal")
    drawString("Number of right clicks: "+str(numRightClicks),100,350,"Arial",28,"normal")


def middleDisplay(x,y):
    global numMiddleClicks
    clear()
    numMiddleClicks += 1
    drawString("Number of left clicks: "+str(numLeftClicks),100,150,"Arial",28,"normal")
    drawString("Number of middle clicks: "+str(numMiddleClicks),100,250,"Arial",28,"normal")
    drawString("Number of right clicks: "+str(numRightClicks),100,350,"Arial",28,"normal")


def rightDisplay(x,y):
    global numRightClicks
    clear()
    numRightClicks += 1
    drawString("Number of left clicks: "+str(numLeftClicks),100,150,"Arial",28,"normal")
    drawString("Number of middle clicks: "+str(numMiddleClicks),100,250,"Arial",28,"normal")
    drawString("Number of right clicks: "+str(numRightClicks),100,350,"Arial",28,"normal")
```

```
39
40 ##########
41 #  MAIN  #
42 ##########
43
44 numLeftClicks = 0
45 numMiddleClicks = 0
46 numRightClicks = 0
47 beginGrfx(800,500)
48 onscreenclick(leftDisplay,btn=1)
49 onscreenclick(middleDisplay,btn=2)
50 onscreenclick(rightDisplay,btn=3)
51 endGrfx()
```

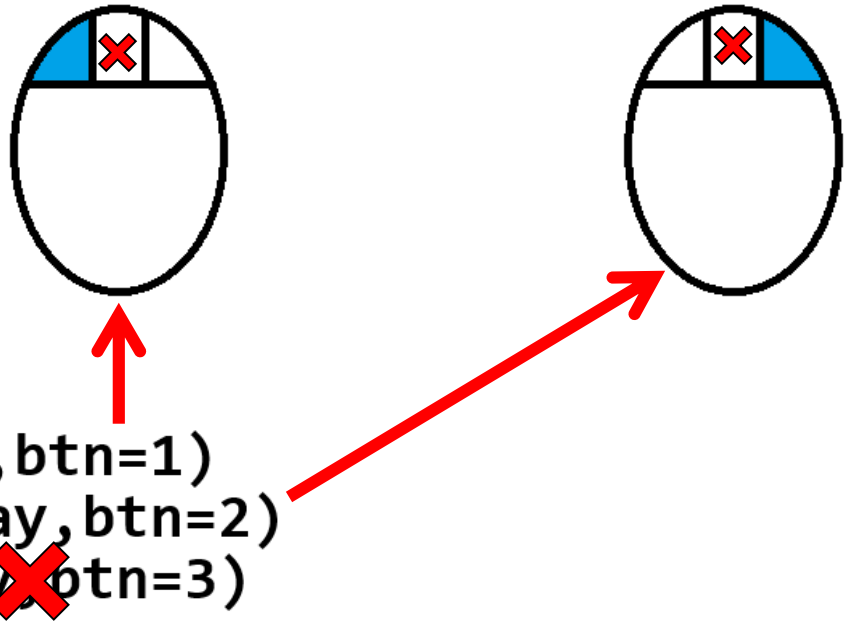


NOTE: On some mice the *wheel* is *clickable* and acts as the *middle button*.

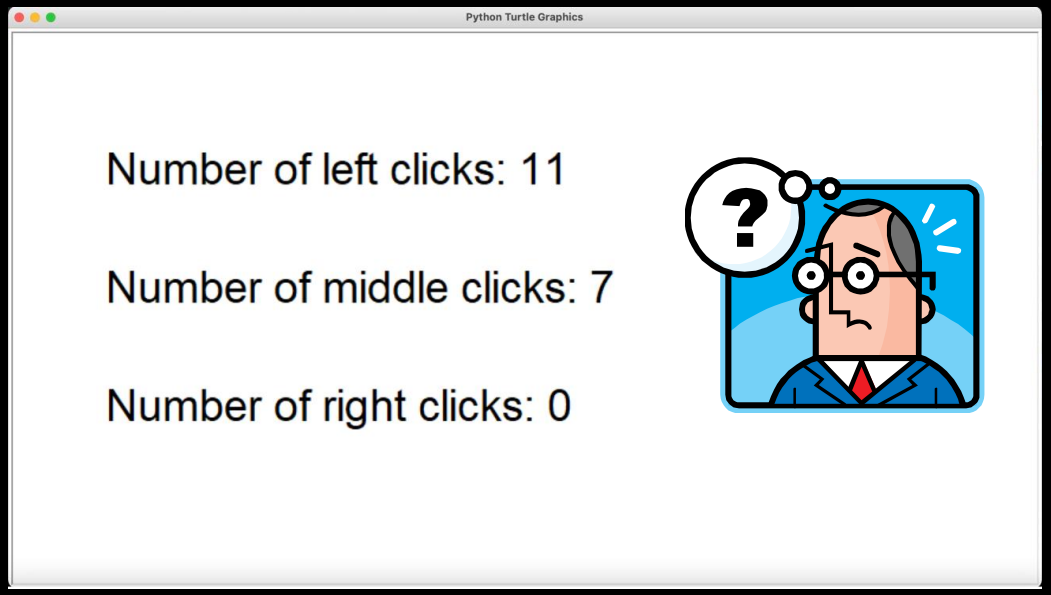




```
39
40 ##########
41 #  MAIN  #
42 ##########
43
44 numLeftClicks = 0
45 numMiddleClicks = 0
46 numRightClicks = 0
47 beginGrfx(800,500)
48 onscreenclick(leftDisplay,btn=1)
49 onscreenclick(middleDisplay,btn=2)
50 onscreenclick(rightDisplay,btn=3)
51 endGrfx()
```

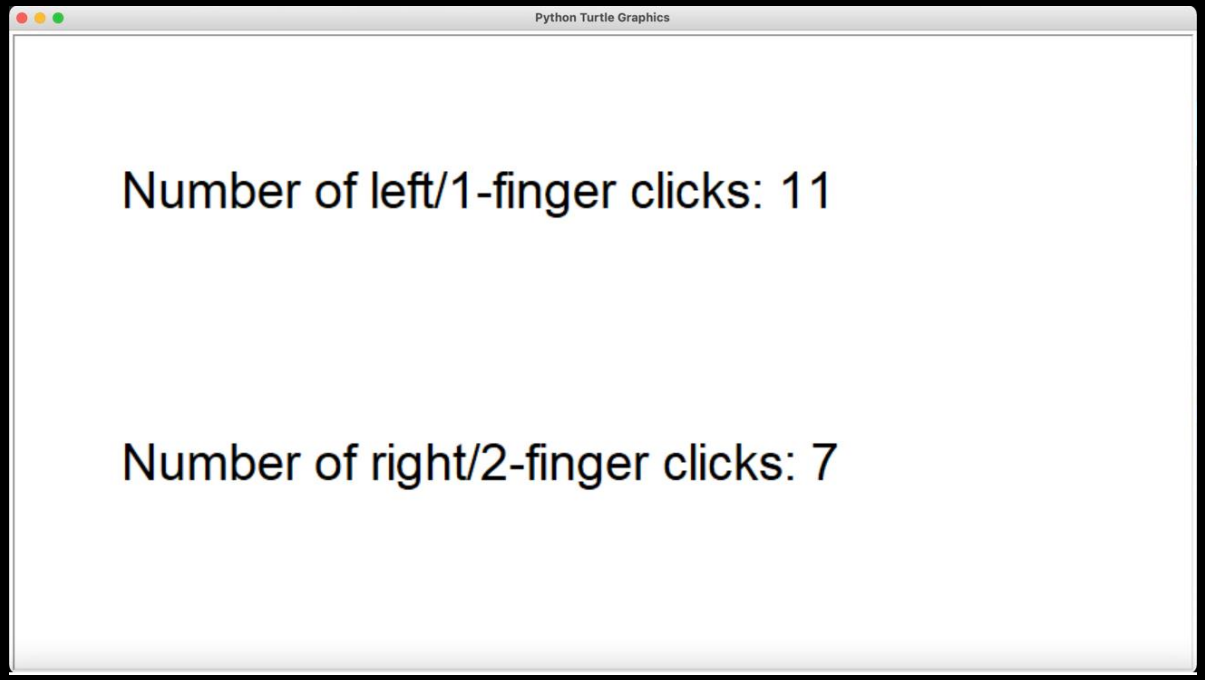


ALSO: If you run this program on a Mac, and right/2-finger click 7 times, you get this output. Since Macs don't have any type of "middle-click" they use a button value of 2 for right/2-finger click.



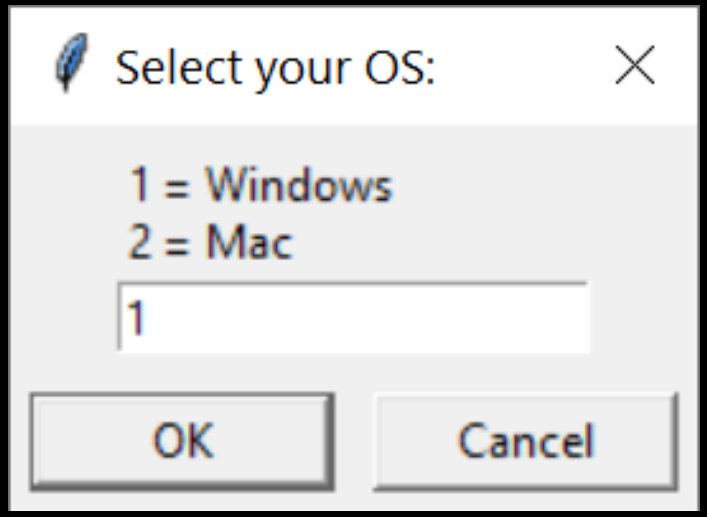
```
1  # MouseEvents03.py
2  # This program demonstrates how to distinguish between a
3  # left/1-finger click & a right/2-finger click on a Mac.
4  # NOTE: On a Mac, button 2 is for right/2-finger click.
5  # ALSO: There is no middle-click on a Mac.
6
7
8  from Graphics import *
9
10
11 def leftDisplay(x,y):
12     global numLeftClicks
13     clear()
14     numLeftClicks += 1
15     drawString("Number of left/1-finger clicks: "+str(numLeftClicks),100,150,"Arial",28,"normal")
16     drawString("Number of right/2-finger clicks: "+str(numRightClicks),100,350,"Arial",28,"normal")
17
18
19 def rightDisplay(x,y):
20     global numRightClicks
21     clear()
22     numRightClicks += 1
23     drawString("Number of left/1-finger clicks: " +str(numLeftClicks),100,150,"Arial",28,"normal")
24     drawString("Number of right/2-finger clicks: " +str(numRightClicks),100,350,"Arial",28,"normal")
25
26
```

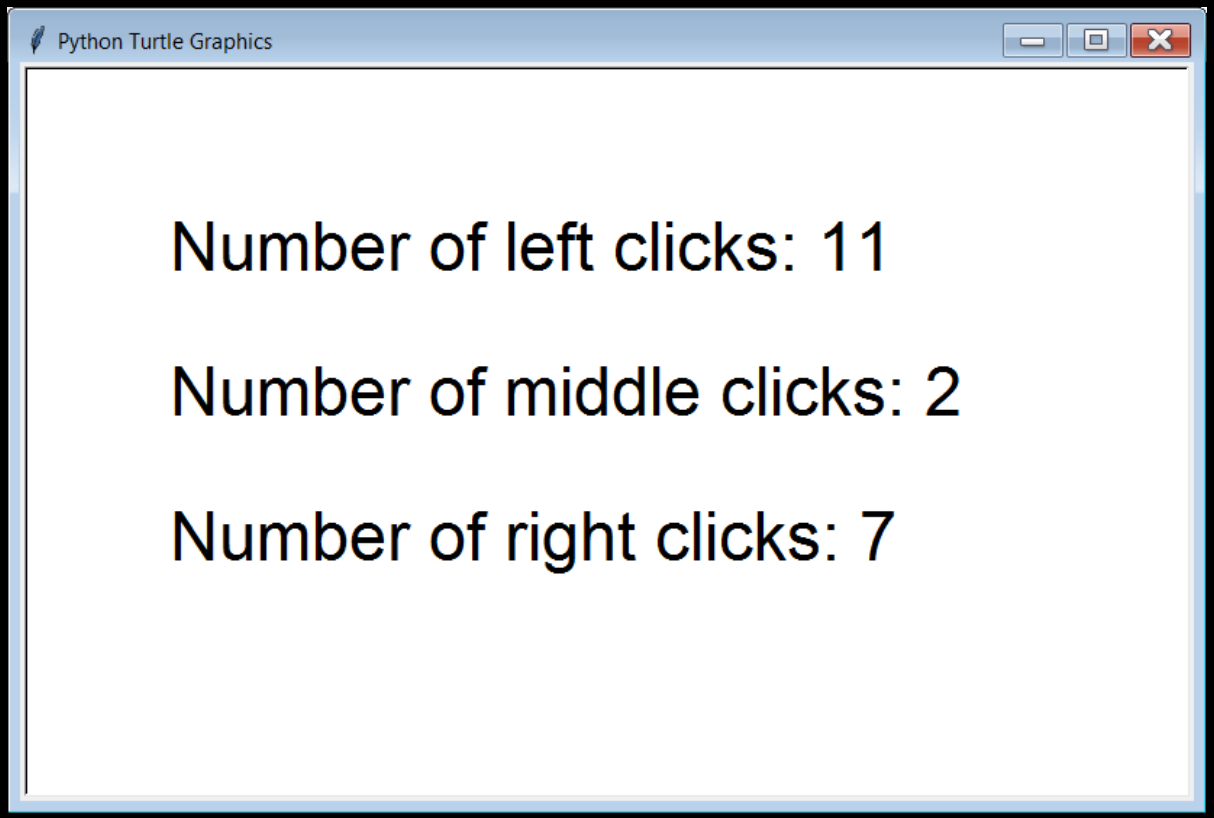
```
27
28 ##########
29 #  MAIN  #
30 ##########
31
32 numLeftClicks = 0
33 numRightClicks = 0
34 beginGrfx(800,500)
35 onscreenclick(leftDisplay,btn=1)
36 onscreenclick(rightDisplay,btn=2)
37 endGrfx()
```

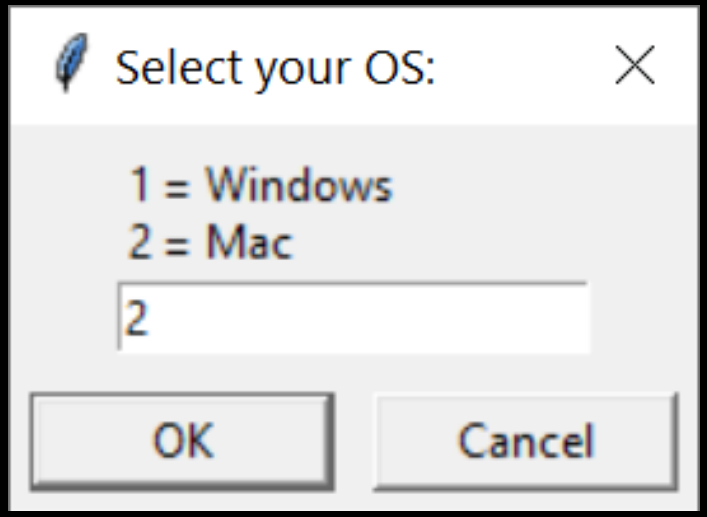


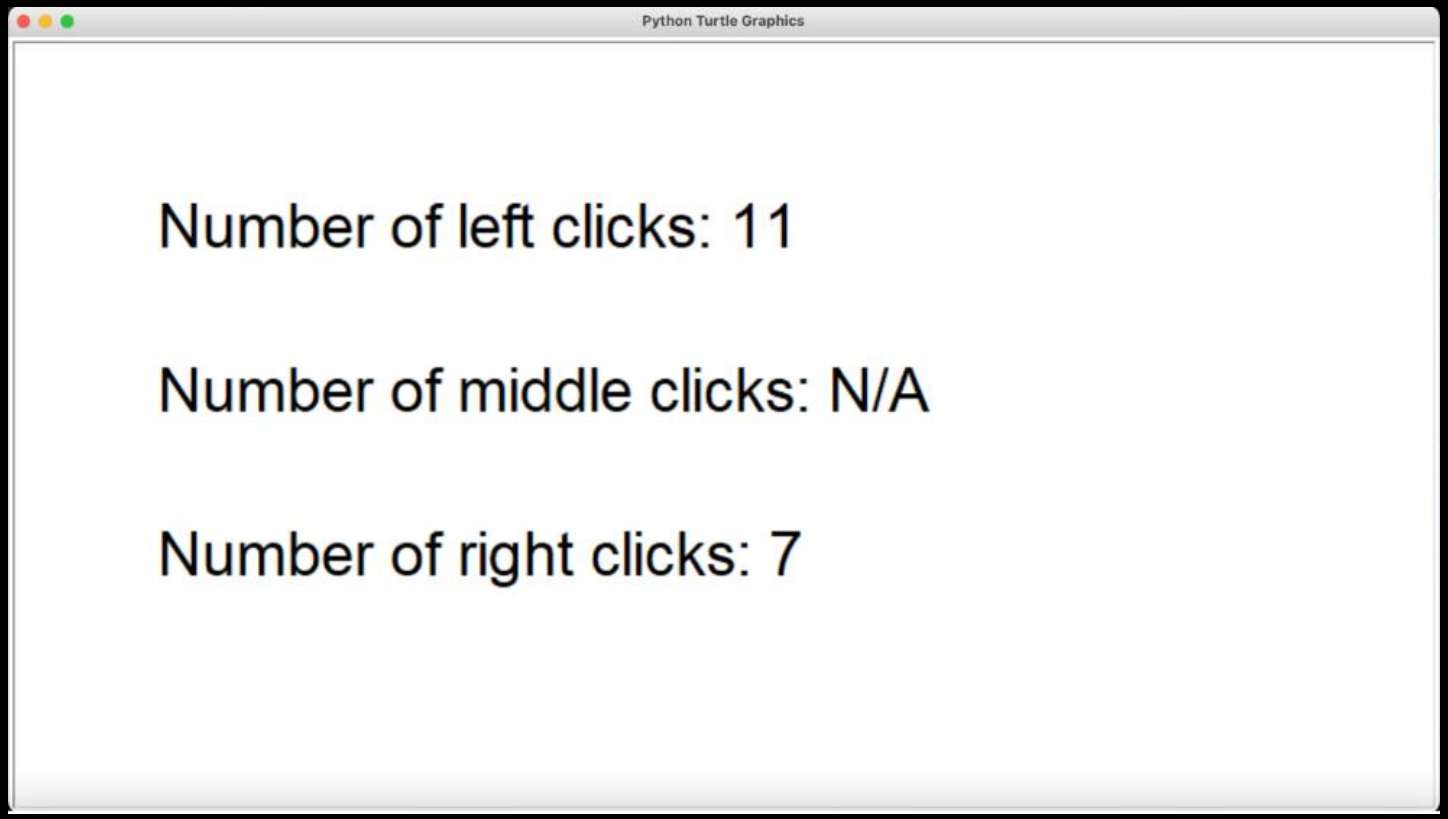
Determining the Operating System

```
 1 # MouseEvents04.py
 2 # This program will allow the user to select the computer's
 3 # operating system and use this information to properly
 4 # determine the button value of the right/2-finger click
 5 # and whether or not a middle-click exists.
 6 # NOTE: There is no guarantee that the user will select
 7 #       the correct operating system.
:
43 numLeftClicks = 0
44 numMiddleClicks = 0
45 numRightClicks = 0
46 right = 4
47 middle = 4
48
49 os = numinput("Select your OS:","1 = Windows\n2 = Mac")
50 if os == 1 : # Windows
51     middle = 2
52     right = 3
53 if os == 2 : # Mac
54     right = 2
55     numMiddleClicks = "N/A"
56
57 beginGrfx(800,500)
58 onscreenclick(leftDisplay,btn=1)
59 onscreenclick(middleDisplay,btn=middle)
60 onscreenclick(rightDisplay,btn=right)
61 endGrfx()
```


Select your OS:
1 = Windows
2 = Mac
1
OK
Cancel

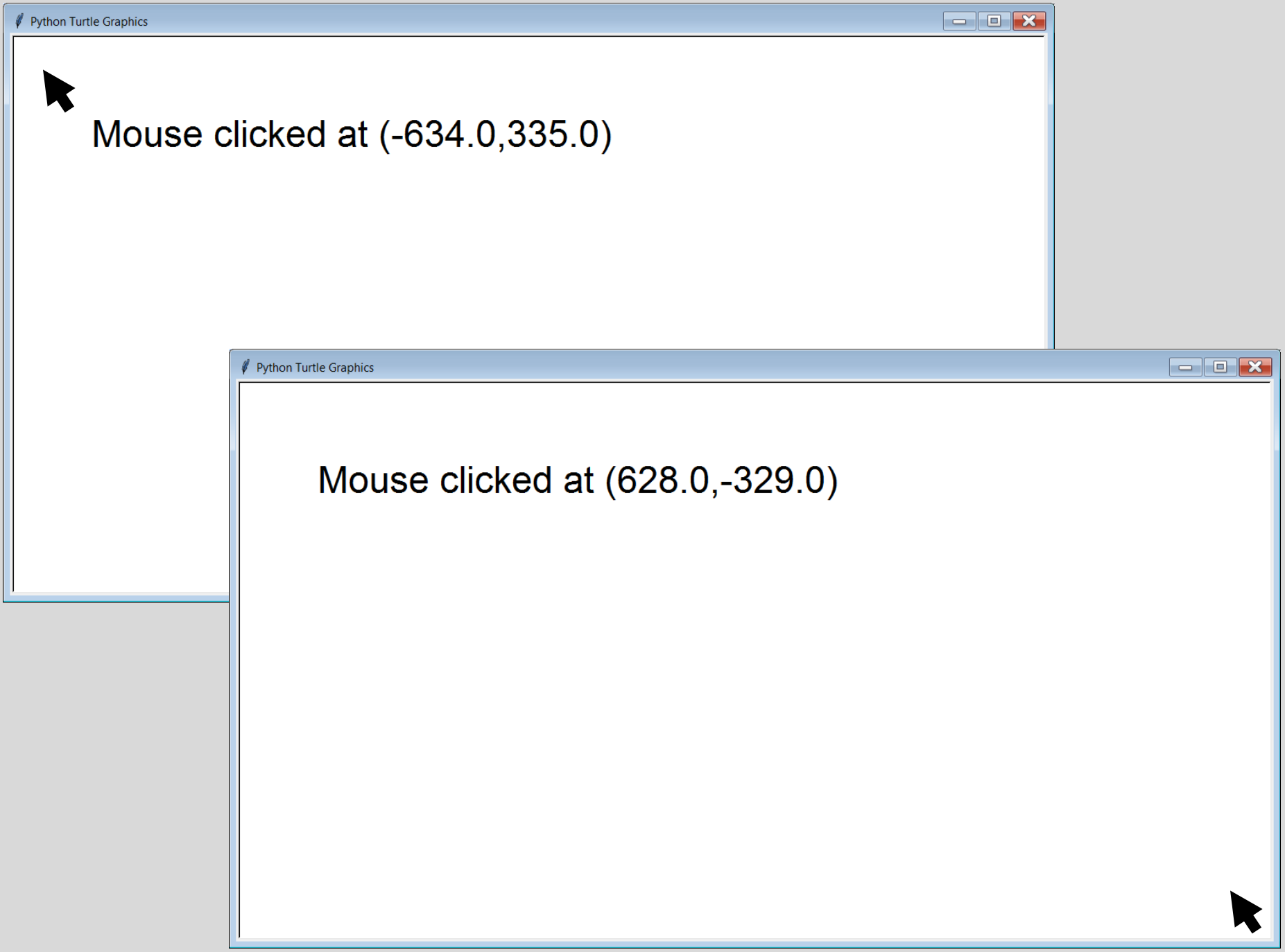

Python Turtle Graphics
Number of left clicks: 11
Number of middle clicks: 2
Number of right clicks: 7


Select your OS:
1 = Windows
2 = Mac
2
OK
Cancel


Python Turtle Graphics
Number of left clicks: 11
Number of middle clicks: N/A
Number of right clicks: 7

```
 1 # MouseEvents05.py
 2 # This program is more efficient and reliable that the
 3 # previous program because it can actually detect the
 4 # computer's operating system and use this information
 5 # to properly determine the button value of the right/
 6 # 2-finger click and whether or not a middle-click exists.
 7
 8
 9 from Graphics import *
10 import os
:
43 numLeftClicks = 0
44 numMiddleClicks = 0
45 numRightClicks = 0
46 right = 4
47 middle = 4
48 if os.name == "nt": # Windows
49    middle = 2
50    right = 3
51 if os.name == "posix": # Mac
52    right = 2
53    numMiddleClicks = "N/A"
54
55 beginGrfx(800,500)
56 onscreenclick(leftDisplay,btn=1)
57 onscreenclick(middleDisplay,btn=middle)
58 onscreenclick(rightDisplay,btn=right)
59 endGrfx()
```

```
 1 # MouseEvents06.py
 2 # This program demonstrates that the <x> and <y>
 3 # parameters store the location where the user
 4 # clicked on the screen.  The problem is, the
 5 # coordinate displayed is based on "Turtle Graphics"
 6 # where (0,0) is in the center of the screen.
 7 # This results in some negative coordinates.
 8
 9
10 from Graphics import *
11
12
13 def display(x,y):
14     clear()
15     drawString("Mouse clicked at ("+str(x)+","+str(y)+")",
100,150,"Arial",28,"normal")
16
17
18
19 ###########
20 #  MAIN   #
21 ###########
22
23 beginGrfx(1300,700)
24 onscreenclick(display)
25 endGrfx()
```


Python Turtle Graphics
Mouse clicked at (-634.0,335.0)
Python Turtle Graphics
Mouse clicked at (628.0,-329.0)

Python Turtle Graphics

Mouse clicked at (-634.0,335.0)

Python Turtle Graphics

Mouse clicked at (628.0,-329.0)

Python Turtle Graphics

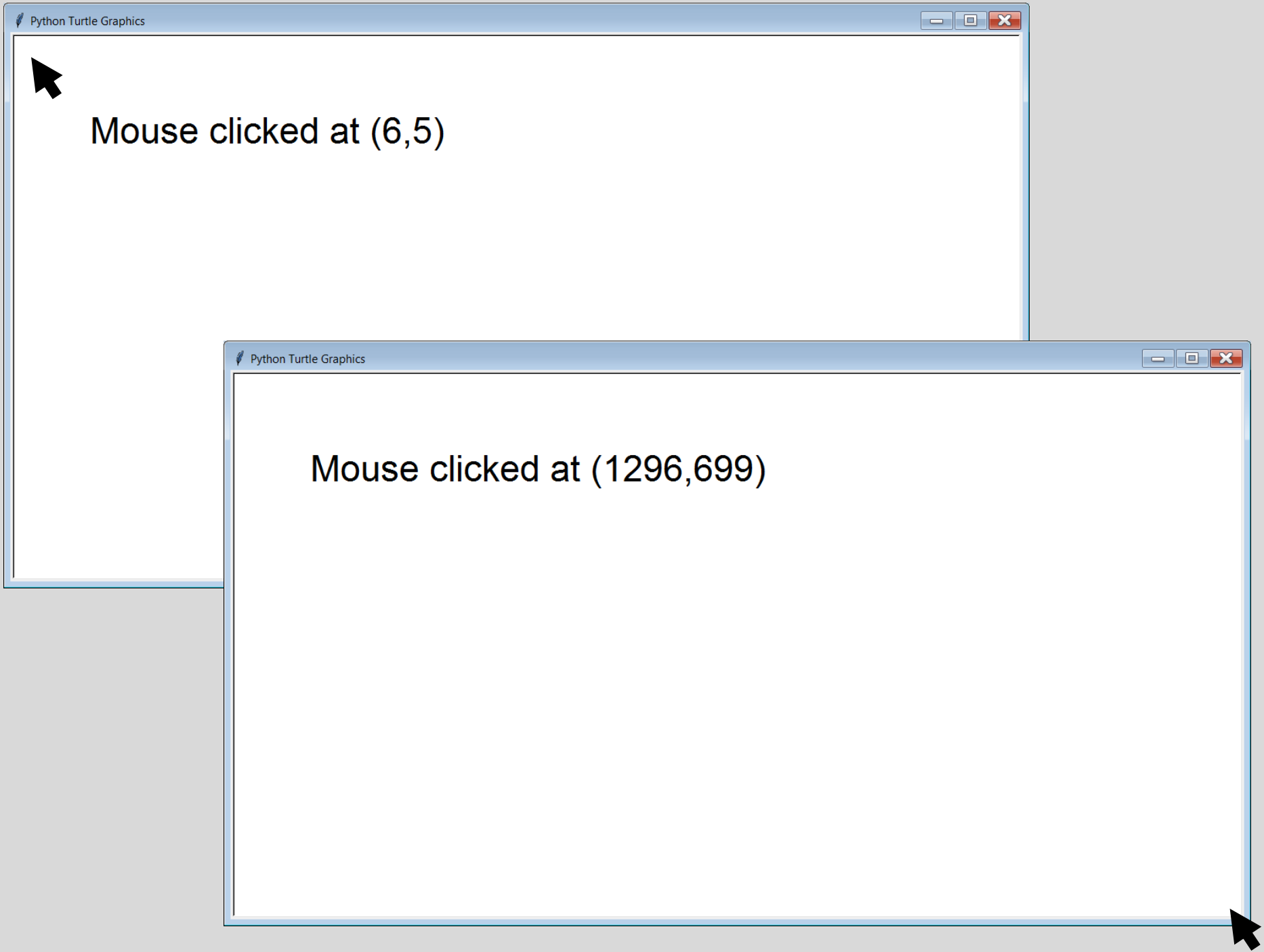
Mouse clicked at (-634.0,335.0)

Python Turtle Graphics

Mouse clicked at (628.0,-329.0)

The coordinate returned is based on the coordinate system used by *Turtle Graphics* where (0,0) is in the center of the window. Any y values below and x values to the left will be negative as the outputs show.

```
 1 # MouseEvents07.py
 2 # This program fixes the issue of the previous program
 3 # by using the <traditionalXY> function of the <Graphics>
 4 # library.  The result is the coordinate is converted to
 5 # "Traditional Graphics" where (0,0) is in the top-left
 6 # corner and all coordinate values are positive.
 7
 8
 9 from Graphics import *
10
11
12 def display(x,y):
13     x,y = traditionalXY(x,y)
14     clear()
15     drawString("Mouse clicked at ("+str(x)+","+str(y)+")",
100,150,"Arial",28,"normal")
16
17
18
19 ###########
20 #  MAIN   #
21 ###########
22
23 beginGrfx(1300,700)
24 onscreenclick(display)
25 endGrfx()
```


Python Turtle Graphics
Mouse clicked at (6,5)
Python Turtle Graphics
Mouse clicked at (1296,699)

Section 18.3

Creating Simple Paint Program Tools

Paint Program Tools vs. Entire Paint Program

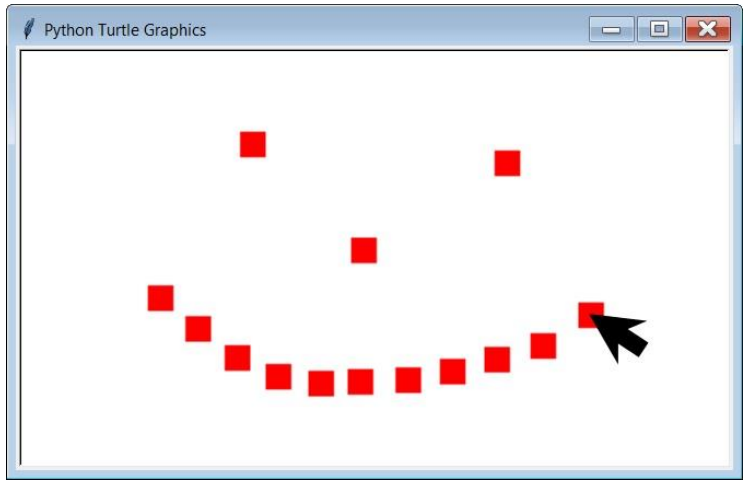
We are not creating an entire paint program. That assignment will have to wait for the *Advanced Graphics* class.

In this class, we are merely introducing some concepts by creating some of the simpler tools from a paint program.

This will lead into Lab 18A.

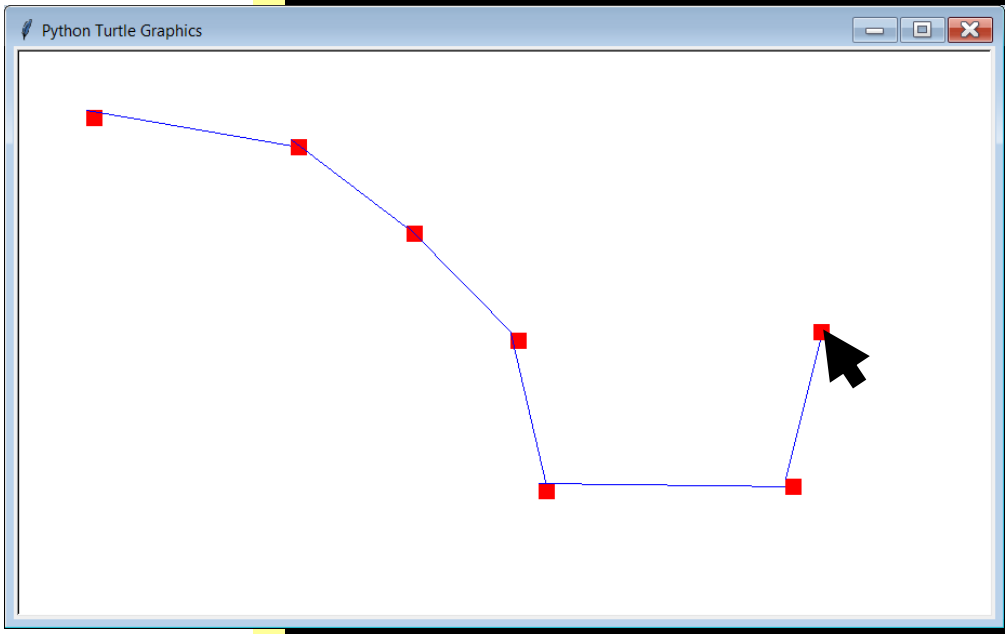
```
 1 # MouseEvents08.py
 2 # This program draws a small red
 3 # square at every click location.
 4
 5
 6 from Graphics import *
 7
 8
 9 def display(x,y):
10     x,y = traditionalXY(x,y)
11     fillRectangle(x-7,y-7,x+7,y+7)
12     update()
13
14
15
16 ##########
17 #  MAIN  #
18 ##########
19
20 beginGrfx(1300,700)
21 setColor("red")
22 onscreenclick(display)
23 endGrfx()
```

```
 1 # MouseEvents08.py
 2 # This program draws a small red
 3 # square at every click location.
 4
 5
 6 from Graphics import *
 7
 8
 9 def display(x,y):
10     x,y = traditionalXY(x,y)
11     fillRectangle(x-7,y-7,x+7,y+7)
12     update()
13
14
15
16 ##########
17 #  MAIN  #
18 ##########
19
20 beginGrfx(1300,700)
21 setColor("red")
22 onscreenclick(display)
23 endGrfx()
```

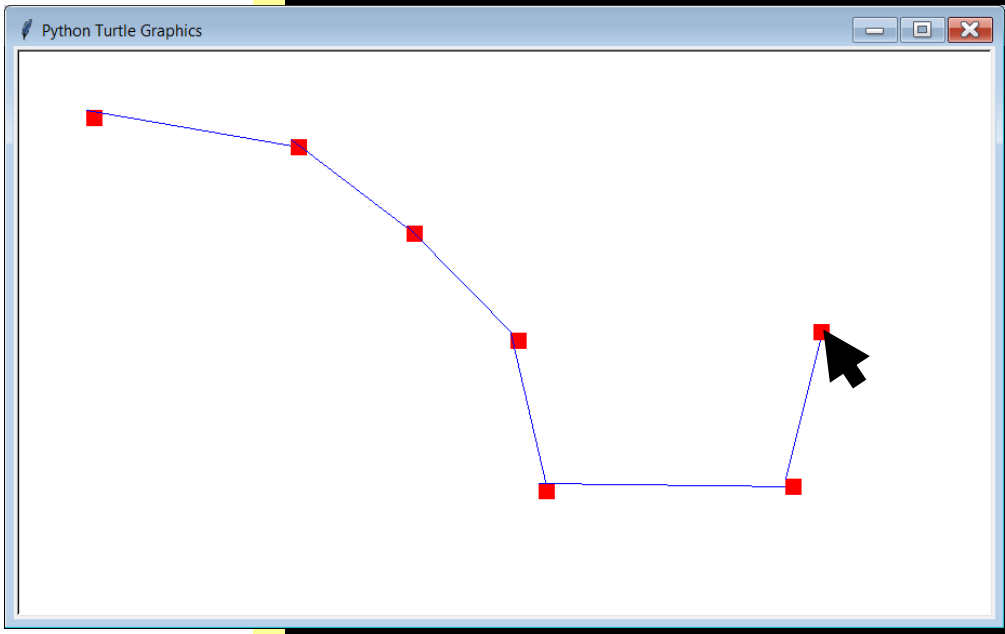


```
1  # MouseEvents09.py
2  # This program draws a small red square when
3  # the mouse is left-clicked and a blue line
4  # when the mouse in right-clicked.
5
6
7  from Graphics import *
8
9
10 def displayDot(x,y):
11     setColor("red")
12     x,y = traditionalXY(x,y)
13     fillRectangle(x-7,y-7,x+7,y+7)
14     update()
15
16 def displayLine(x,y):
17     setColor("blue")
18     goto(x,y)
19     update()
20
21
22
23 ##########
24 #  MAIN  #
25 ##########
26
27 beginGrfx(1300,700)
28 onscreenclick(displayDot,btn=1)
29 onscreenclick(displayLine,btn=3)
30 endGrfx()
```

```python
# MouseEvents09.py
# This program draws a small red square when
# the mouse is left-clicked and a blue line
# when the mouse in right-clicked.


from Graphics import *


def displayDot(x,y):
    setColor("red")
    x,y = traditionalXY(x,y)
    fillRectangle(x-7,y-7,x+7,y+7)
    update()

def displayLine(x,y):
    setColor("blue")
    goto(x,y)
    update()



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

beginGrfx(1300,700)
onscreenclick(displayDot,btn=1)
onscreenclick(displayLine,btn=3)
endGrfx()
```



```python
# MouseEvents09.py
# This program draws a small red square when
# the mouse is left-clicked and a blue line
# when the mouse in right-clicked.


from Graphics import *


def displayDot(x,y):
    setColor("red")
    x,y = traditionalXY(x,y)
    fillRectangle(x-7,y-7,x+7,y+7)
    update()

def displayLine(x,y):
    setColor("blue")
    goto(x,y)
    update()



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

beginGrfx(1300,700)
onscreenclick(displayDot,btn=1)
onscreenclick(displayLine,btn=3)
endGrfx()
```



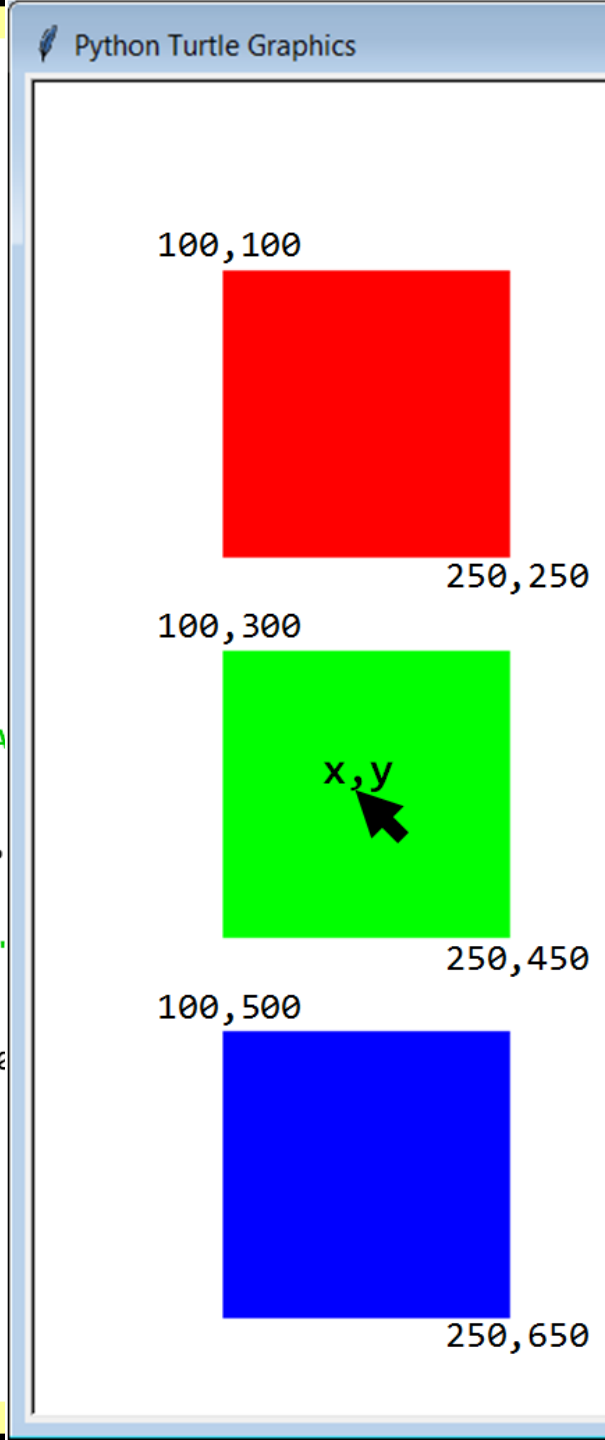
Unlike **drawLine**, with **goto**, you only specify the *end point*. The *start point* is simply the last coordinate that was used. If the program has just started, the *start point* is (0,0). Also, **goto** uses the coordinates from *Turtle Graphics*, so no conversion is necessary.

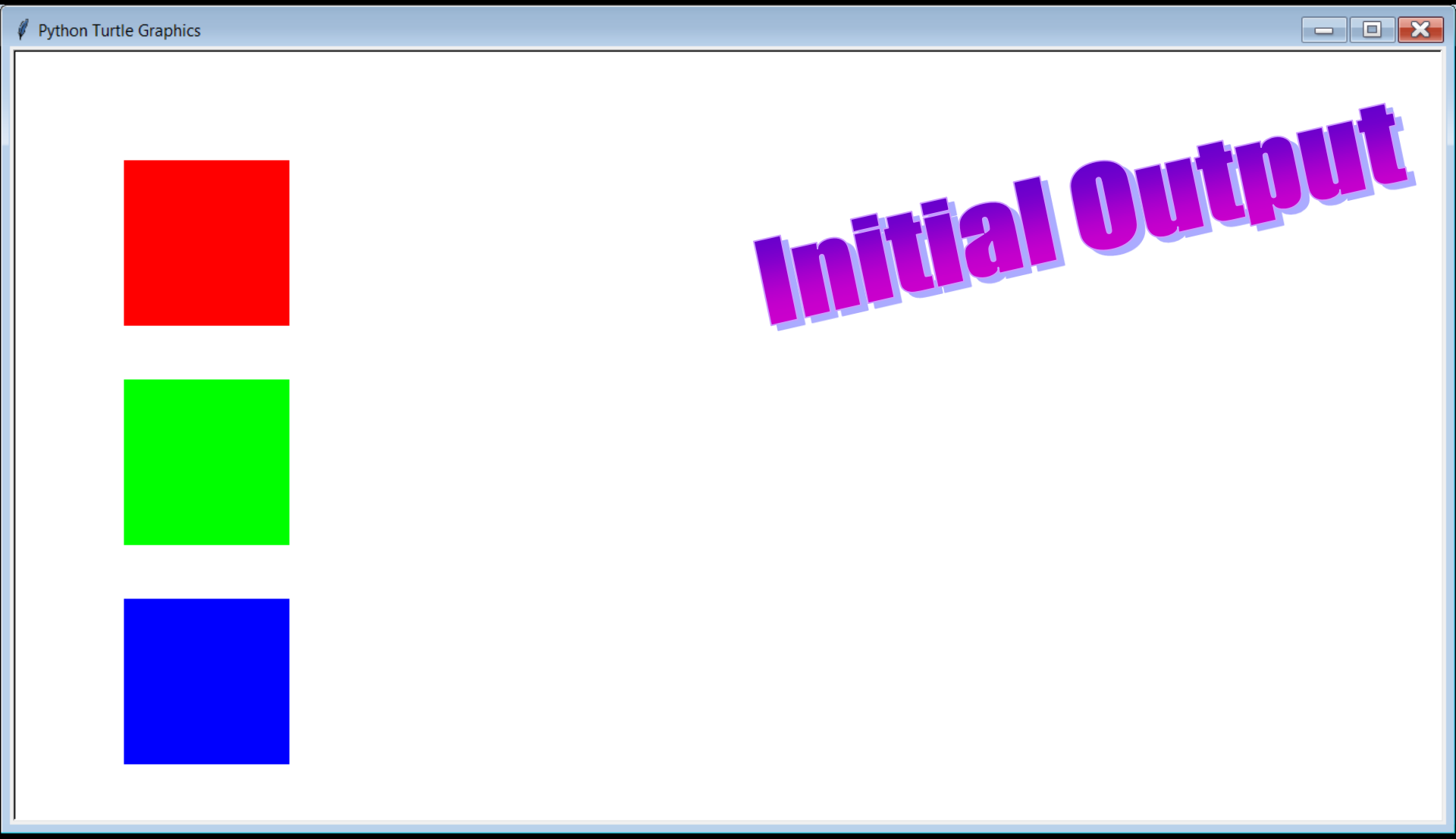
Section 18.4

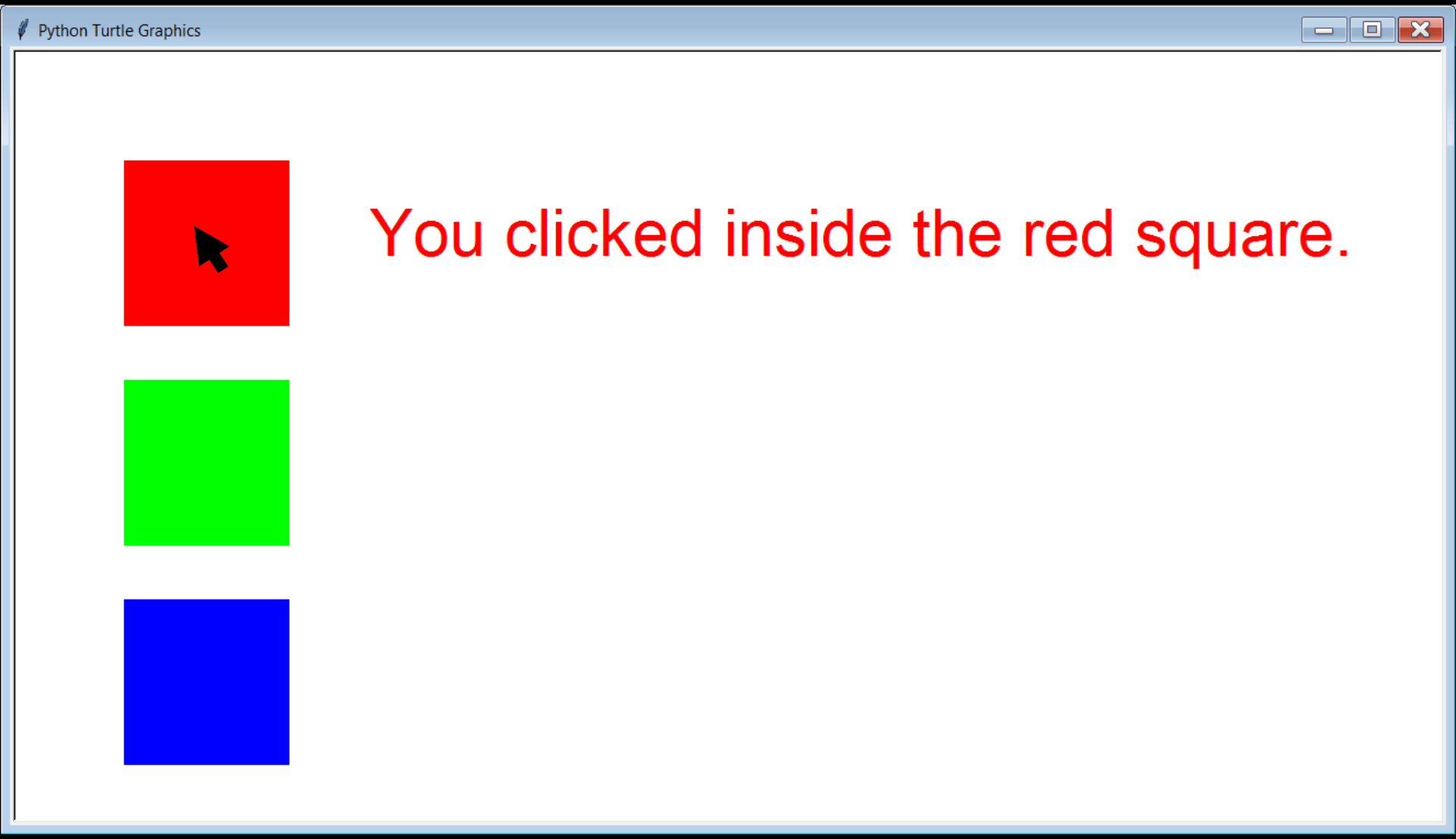
Creating Clickable Areas

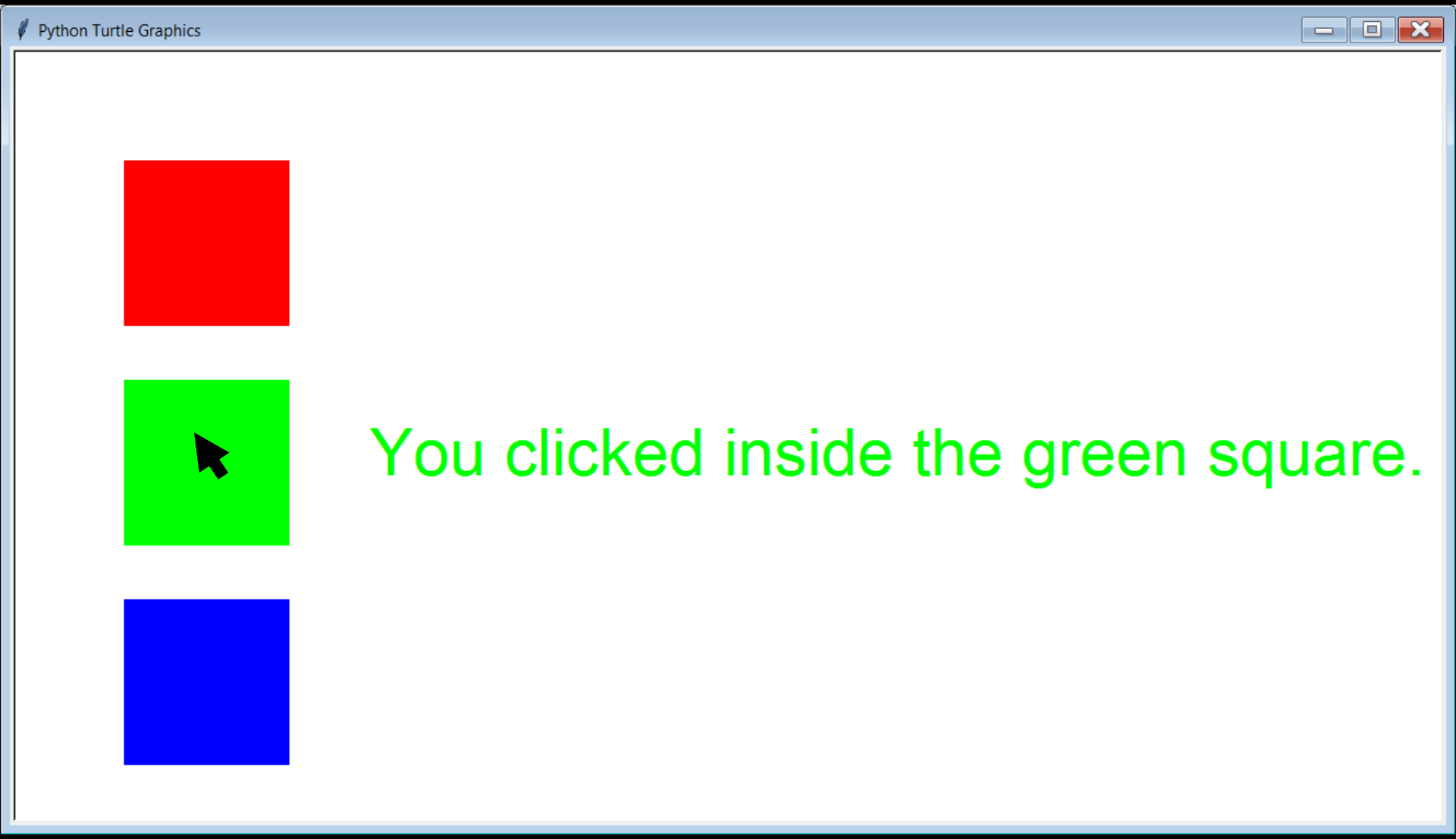
```
# MouseEvents10.py
# This program uses "ranges" to determine
# if the user clicked inside a certain
# rectangular (or square) area of the screen.

from Graphics import *

def displayBoxes():
    setColor("red")
    fillRectangle(100,100,250,250)
    setColor("green")
    fillRectangle(100,300,250,450)
    setColor("blue")
    fillRectangle(100,500,250,650)

def locate(x,y):
    clear()
    displayBoxes()
    x,y = traditionalXY(x,y)
    if 100 <= x <= 250 and 100 <= y <= 250:
        setColor("red")
        drawString("You clicked inside the red square.",325,200,"Arial",36,"normal")
    elif 100 <= x <= 250 and 300 <= y <= 450:
        setColor("green")
        drawString("You clicked inside the green square.",325,400,"Arial",36,"normal")
    elif 100 <= x <= 250 and 500 <= y <= 650:
        setColor("blue")
        drawString("You clicked inside the blue square.",325,600,"Arial",36,"normal")
    else:
        setColor("black")
        drawString("You did not click inside any of the squares.",150,80,"Arial",36,"normal")


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

beginGrfx(1300,700)
displayBoxes()
onscreenclick(locate)
endGrfx()
```

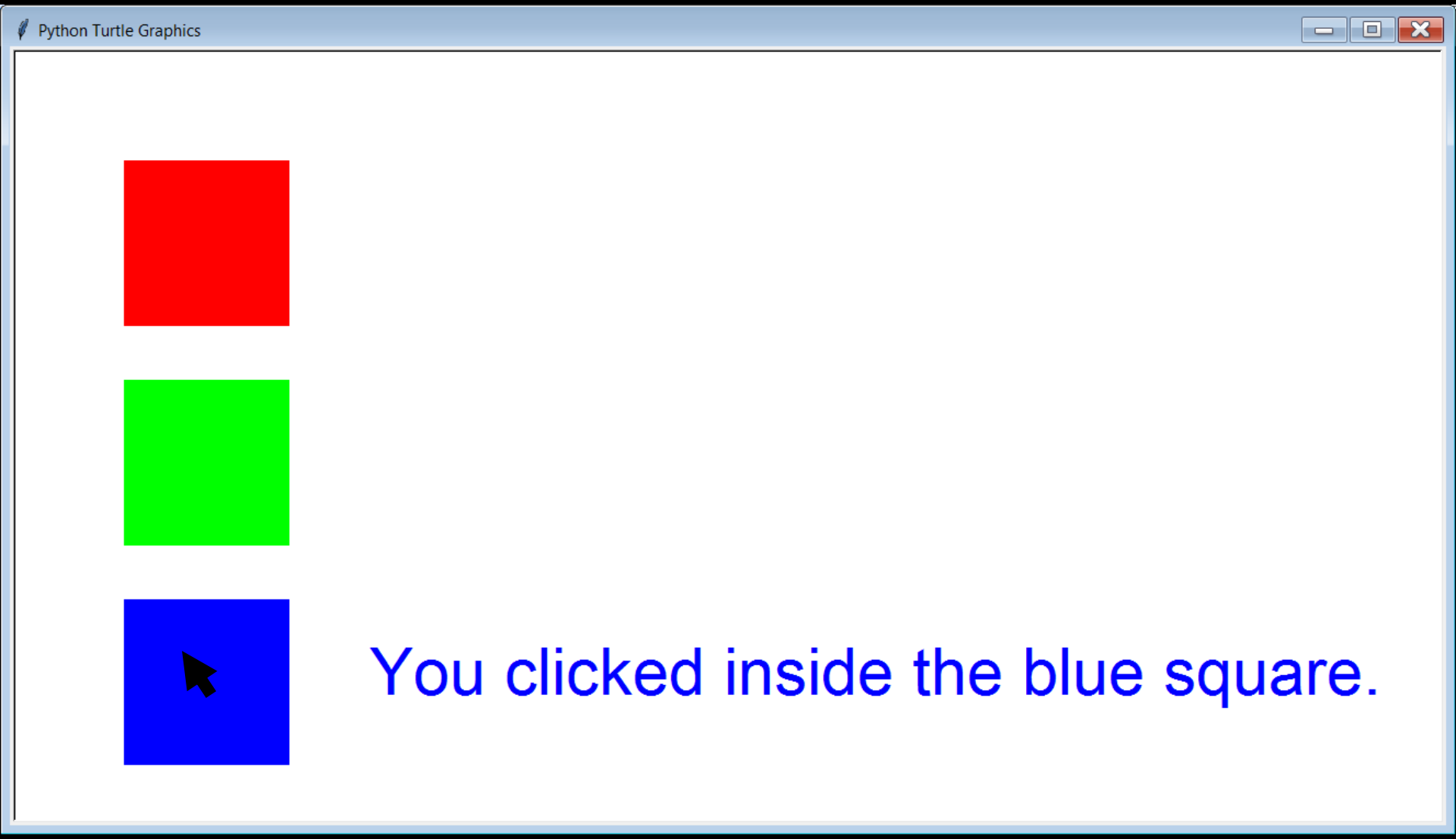
```python
# MouseEvents10.py
# This program uses "ranges" to determine
# if the user clicked inside a certain
# rectangular (or square) area of the screen.

from Graphics import *

def displayBoxes():
    setColor("red")
    fillRectangle(100,100,250,250)
    setColor("green")
    fillRectangle(100,300,250,450)
    setColor("blue")
    fillRectangle(100,500,250,650)

def locate(x,y):
    clear()
    displayBoxes()
    x,y = traditionalXY(x,y)
    if 100 <= x <= 250 and 100 <= y <= 250:
        setColor("red")
        drawString("You clicked inside the red square.",325,200,"A
    elif 100 <= x <= 250 and 300 <= y <= 450:
        setColor("green")
        drawString("You clicked inside the green square.",325,400,
    elif 100 <= x <= 250 and 500 <= y <= 650:
        setColor("blue")
        drawString("You clicked inside the blue square.",325,600,"
    else:
        setColor("black")
        drawString("You did not click inside any of the squares.",150,8


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

beginGrfx(1300,700)
displayBoxes()
onscreenclick(locate)
endGrfx()
```

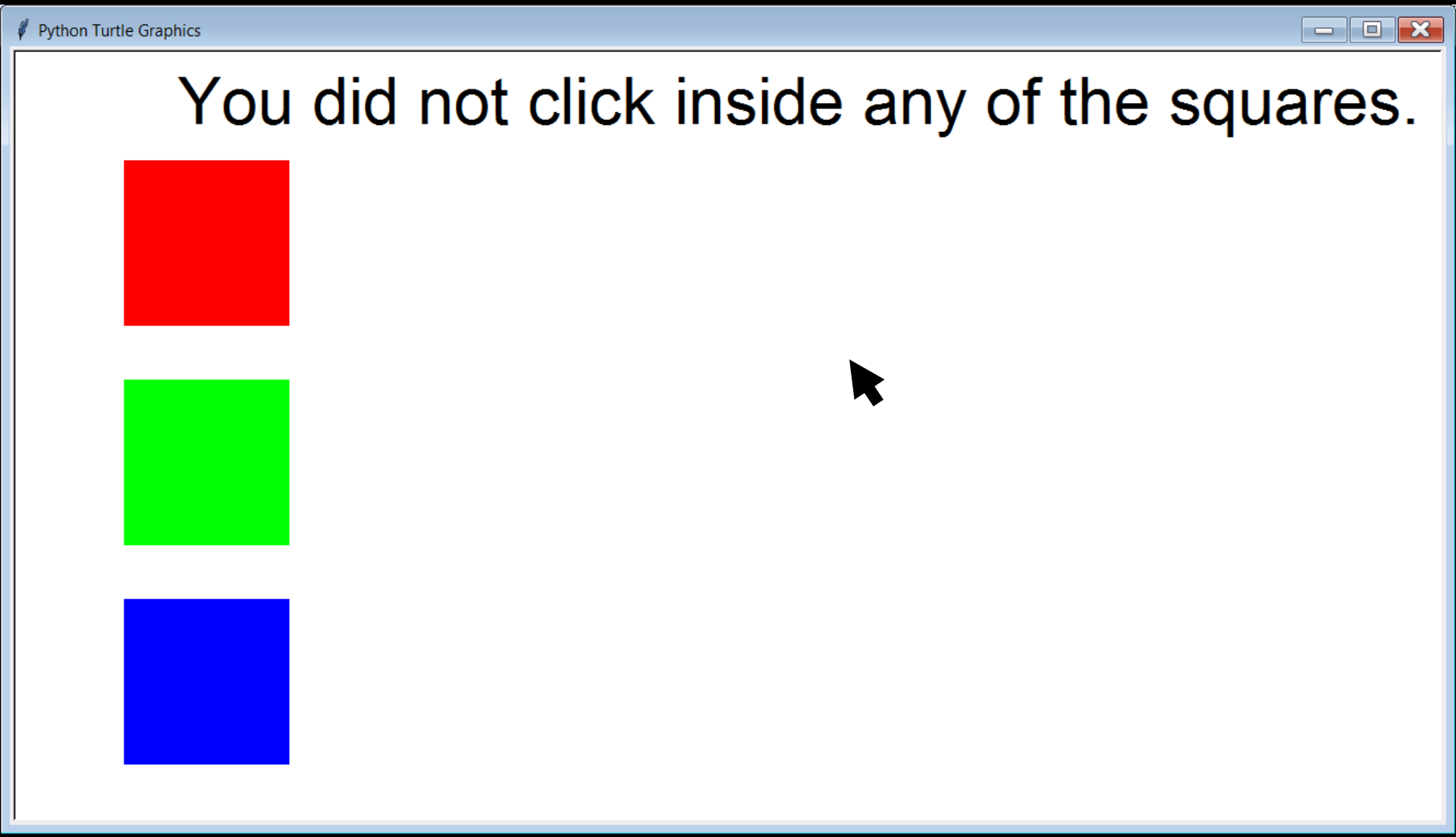



Python Turtle Graphics
Initial Output


Python Turtle Graphics
You clicked inside the red square.

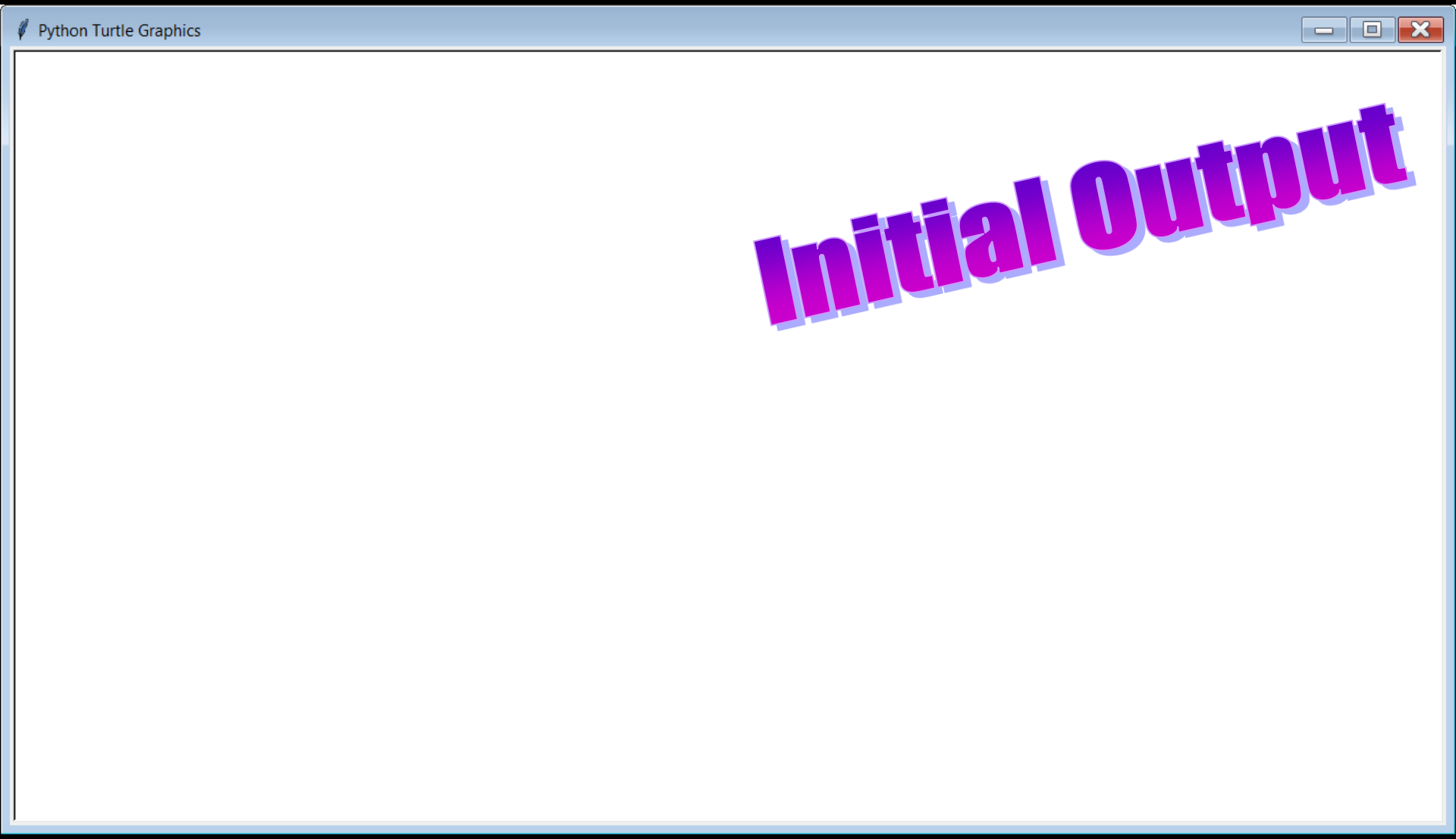

Python Turtle Graphics
You clicked inside the green square.

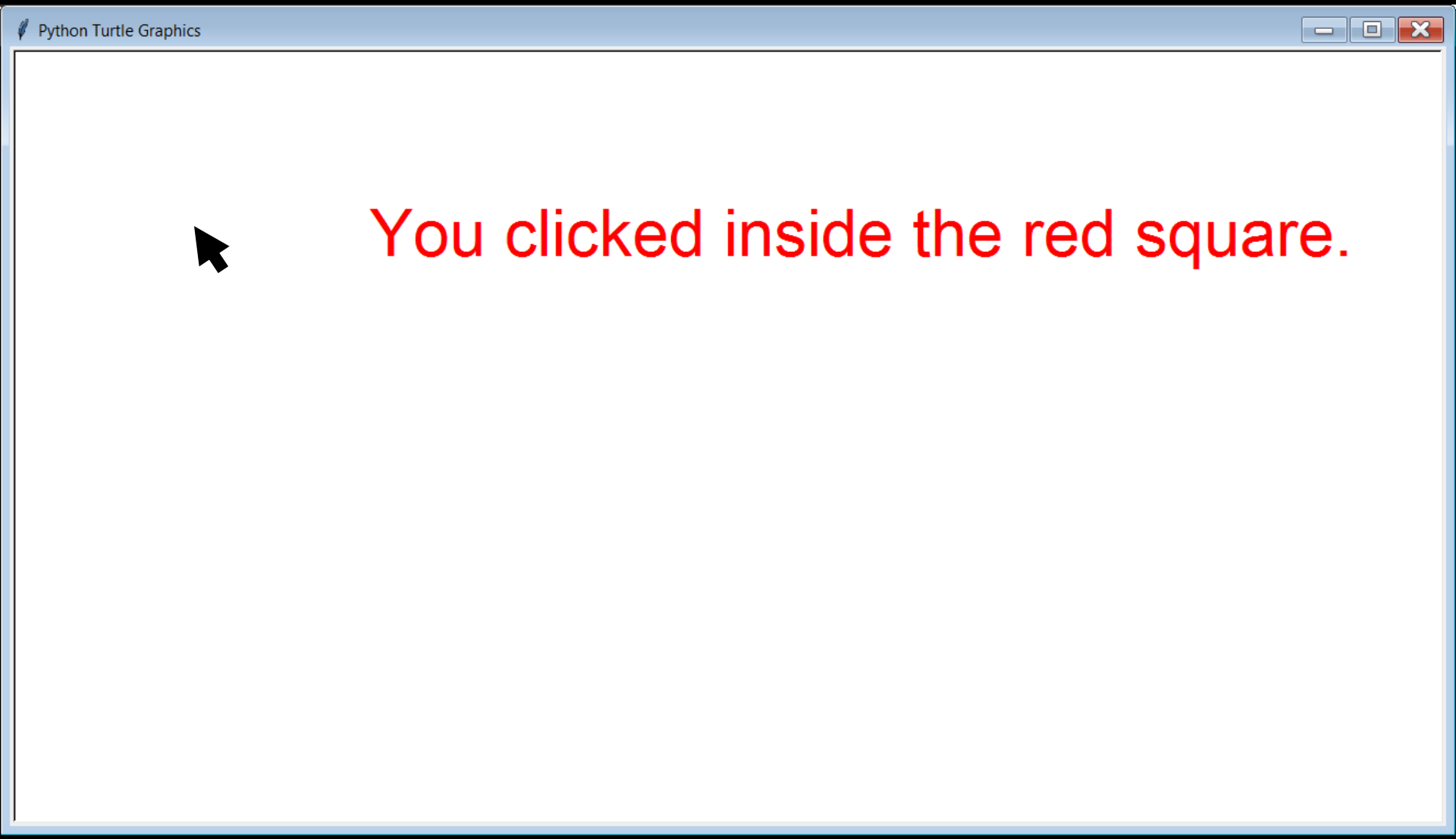

Python Turtle Graphics
You clicked inside the blue square.

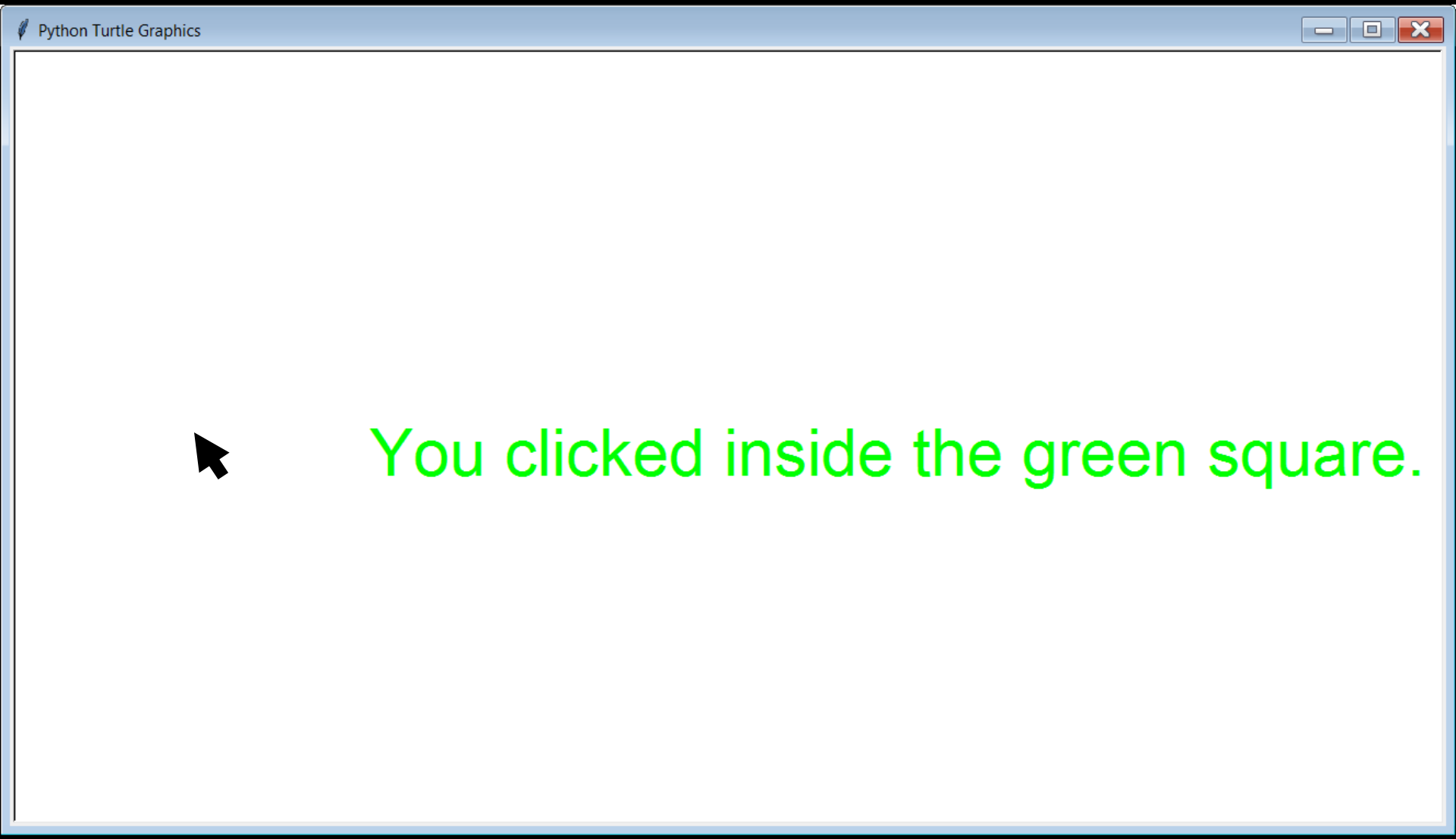

Python Turtle Graphics
You did not click inside any of the squares.

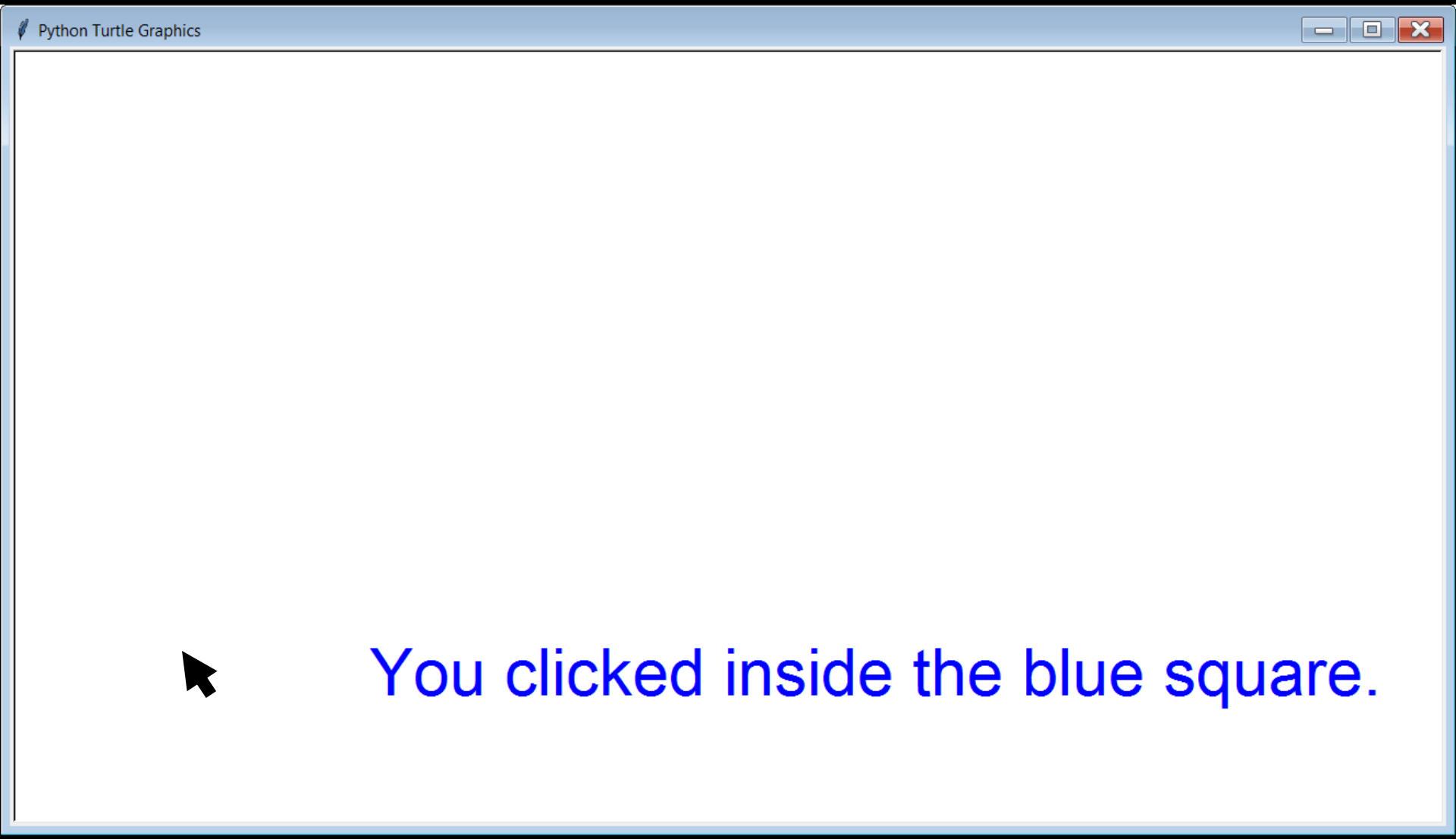
```
 1 # MouseEvents11.py
 2 # This program does the exact same thing as the previous program.
 3 # The code is now a little more efficient because it uses the <inside>
 4 # function of the <Graphics> library.
 5
 6 from Graphics import *
 7
 8 def displayBoxes():
 9     setColor("red")
10     fillRectangle(100,100,250,250)
11     setColor("green")
12     fillRectangle(100,300,250,450)
13     setColor("blue")
14     fillRectangle(100,500,250,650)
15
16 def locate(x,y):
17     clear()
18     displayBoxes()
19     if inside(x,y,100,100,250,250):
20         setColor("red")
21         drawString("You clicked inside the red square.",325,200,"Arial",36,"normal")
22     elif inside(x,y,100,300,250,450):
23         setColor("green")
24         drawString("You clicked inside the green square.",325,400,"Arial",36,"normal")
25     elif inside(x,y,100,500,250,650):
26         setColor("blue")
27         drawString("You clicked inside the blue square.",325,600,"Arial",36,"normal")
28     else:
29         setColor("black")
30         drawString("You did not click inside any of the squares.",150,80,"Arial",36,"normal")
31
32
33 ###########
34 #  MAIN   #
35 ###########
36
37 beginGrfx(1300,700)
38 displayBoxes()
39 onscreenclick(locate)
40 endGrfx()
```

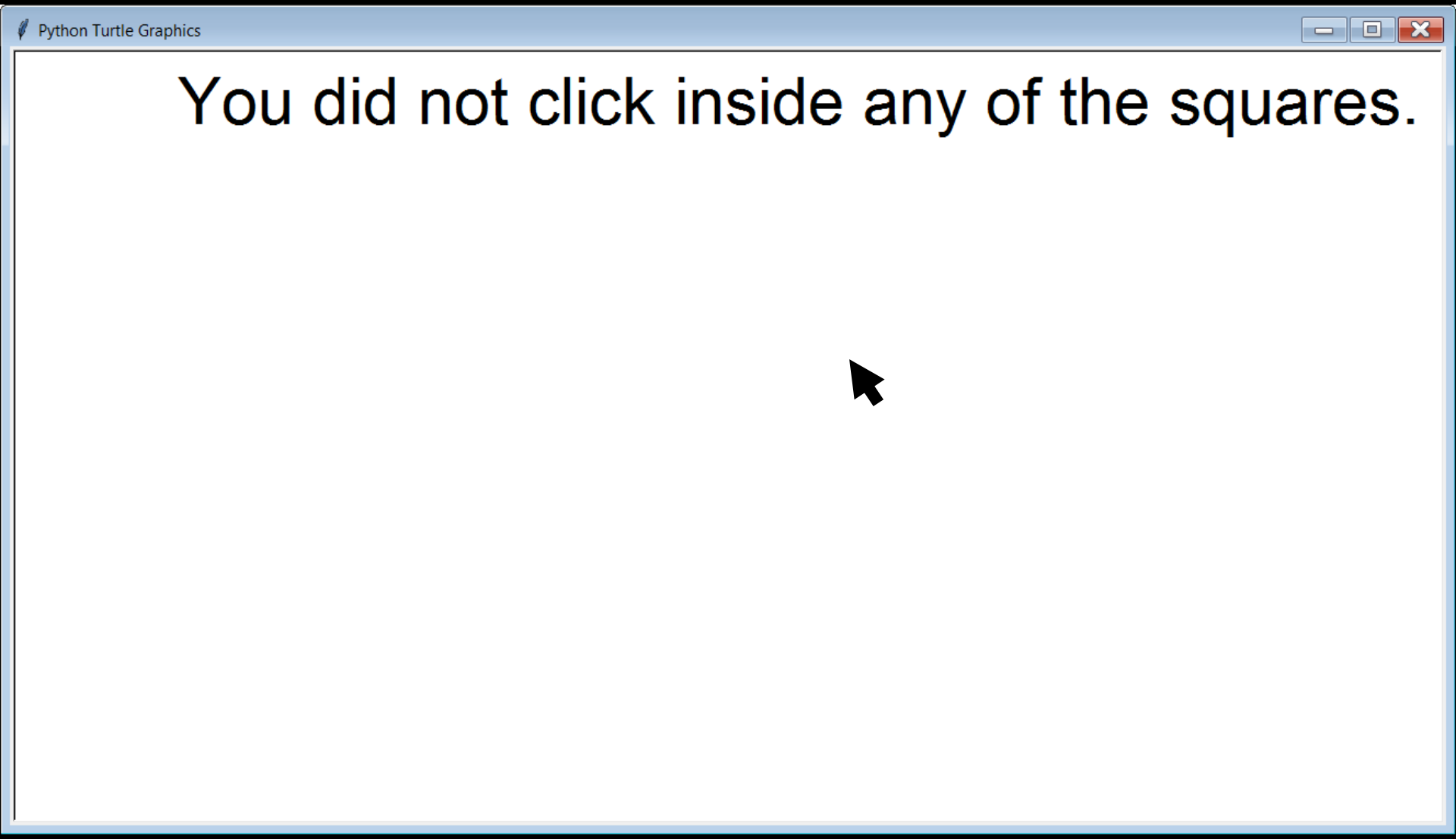
```
 1 # MouseEvents12.py
 2 # The only difference between this program and the previous program is that
 3 # in this one the calls to procedure <displayBoxes> have been commented out.
 4 # Note that even though the boxes are not displayed, you can still click on
 5 # them... or at least, you can click on the area they occupied.
 6
 7 from Graphics import *
 8
 9 def displayBoxes():
10     setColor("red")
11     fillRectangle(100,100,250,250)
12     setColor("green")
13     fillRectangle(100,300,250,450)
14     setColor("blue")
15     fillRectangle(100,500,250,650)
16
17 def locate(x,y):
18     clear()
19     #displayBoxes()
20     if inside(x,y,100,100,250,250):
21         setColor("red")
22         drawString("You clicked inside the red square.",325,200,"Arial",36,"normal")
23     elif inside(x,y,100,300,250,450):
24         setColor("green")
25         drawString("You clicked inside the green square.",325,400,"Arial",36,"normal")
26     elif inside(x,y,100,500,250,650):
27         setColor("blue")
28         drawString("You clicked inside the blue square.",325,600,"Arial",36,"normal")
29     else:
30         setColor("black")
31         drawString("You did not click inside any of the squares.",150,80,"Arial",36,"normal")
32
33
34 ##########
35 #  MAIN  #
36 ##########
37
38 beginGrfx(1300,700)
39 #displayBoxes()
40 onscreenclick(locate)
41 endGrfx()
```


Python Turtle Graphics
Initial Output


Python Turtle Graphics
You clicked inside the red square.

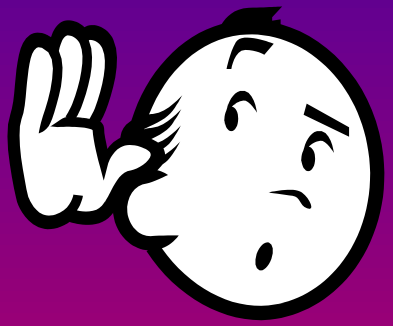

Python Turtle Graphics
You clicked inside the green square.


Python Turtle Graphics
You clicked inside the blue square.


Python Turtle Graphics
You did not click inside any of the squares.

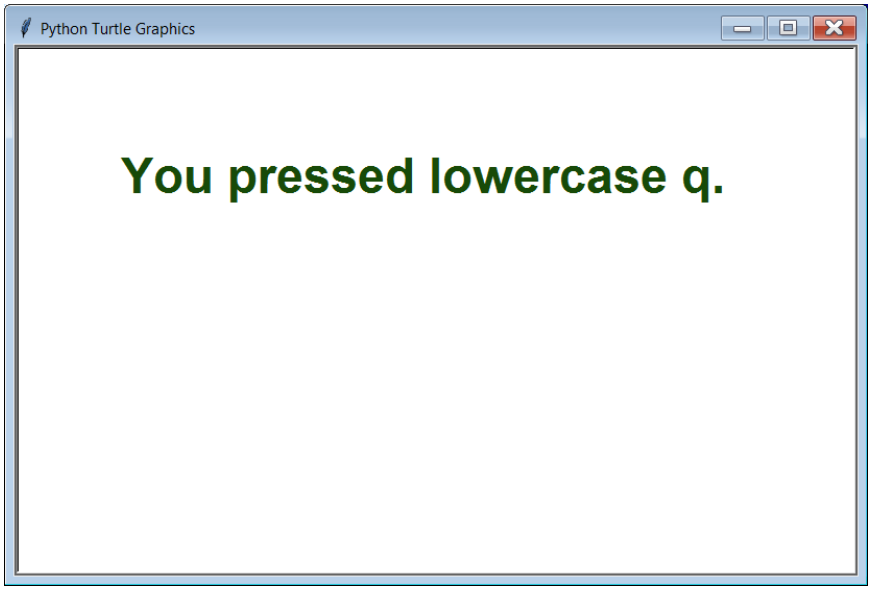
Section 18.5

Key Events



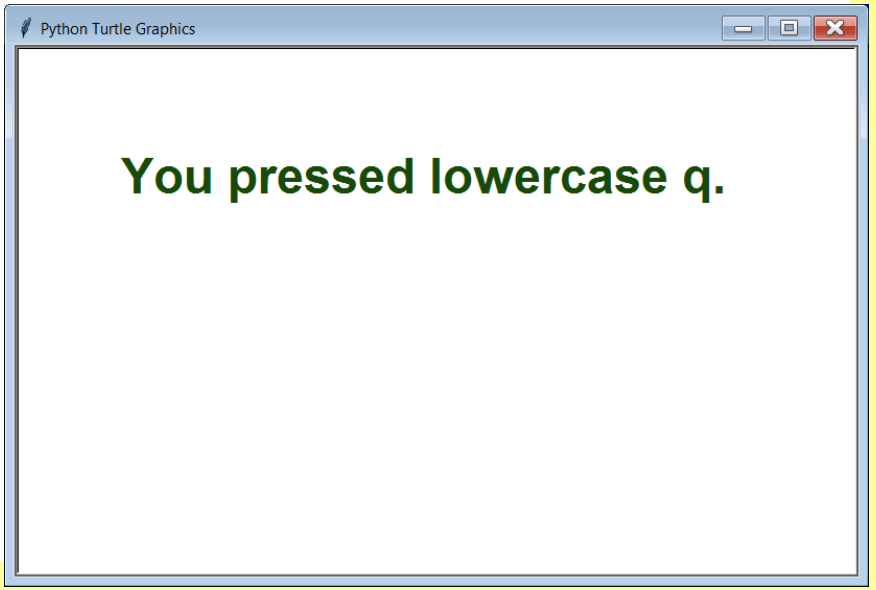
```
1  # KeyEvents01.py
2  # This program introduces "Key Interactivity"
3  # by displaying a randomly colored message
4  # anytime lowercase 'q' is typed.
5  # NOTE: This not the same as "Keyboard Input".
6  # ALSO: The <listen> command is necessary for
7  # the <onkey> command to function properly.
8
9
10 from Graphics import *
11
12
13 def displayq():
14     clear()
15     setRandomColor()
16     drawString("You pressed lowercase q.",100,150,"Arial",28,"bold")
17     update()
18
19
20
21 ##########
22 #  MAIN  #
23 ##########
24
25
26 beginGrfx(800,500)
27 listen()
28 onkey(displayq,"q")
29 endGrfx()
```

```
1  # KeyEvents01.py
2  # This program introduces "Key Interactivity"
3  # by displaying a randomly colored message
4  # anytime lowercase 'q' is typed.
5  # NOTE: This not the same as "Keyboard Input".
6  # ALSO: The <listen> command is necessary for
7  # the <onkey> command to function properly.
8
9
10 from Graphics import *
11
12
13 def displayq():
14     clear()
15     setRandomColor()
16     drawString("You pressed lowercase q.",100,150,"Arial",28,"bold")
17     update()
18
19
20
21 ##########
22 #  MAIN  #
23 ##########
24
25
26 beginGrfx(800,500)
27 listen()
28 onkey(displayq,"q")
29 endGrfx()
```

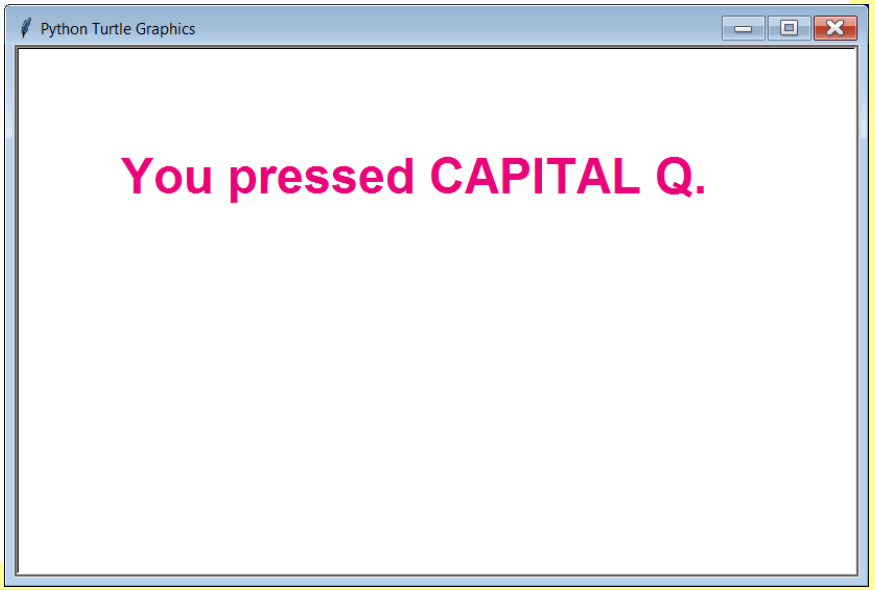


```
 1 # KeyEvents02.py
 2 # This program demonstrates that the computer
 3 # can "listen" for multiple different keys.
 4 # It also shows that CAPITAL 'Q' and lowercase 'q'
 5 # are treated as different keys.
 6
 7
 8 from Graphics import *
 9
10
11 def displayq():
12     clear()
13     setRandomColor()
14     drawString("You pressed lowercase q.",100,150,"Arial",28,"bold")
15     update()
16
17 def displayQ():
18     clear()
19     setRandomColor()
20     drawString("You pressed CAPITAL Q.",100,150,"Arial",28,"bold")
21     update()
22
23
24
25 ###########
26 #  MAIN   #
27 ###########
28
29
30 beginGrfx(800,500)
31 listen()
32 onkey(displayq,"q")
33 onkey(displayQ,"Q")
34 endGrfx()
```

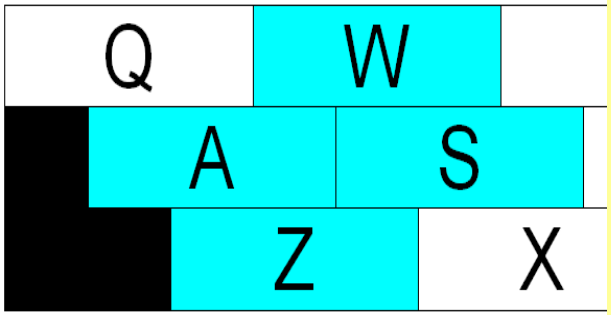
```
1  # KeyEvents02.py
2  # This program demonstrates that the computer
3  # can "listen" for multiple different keys.
4  # It also shows that CAPITAL 'Q' and lowercase 'q'
5  # are treated as different keys.
6
7
8  from Graphics import *
9
10
11 def displayq():
12     clear()
13     setRandomColor()
14     drawString("You pressed lowercase q.",100,150,"Arial",28,"bold")
15     update()
16
17 def displayQ():
18     clear()
19     setRandomColor()
20     drawString("You pressed CAPITAL Q.",100,150,"Arial",28,"bold")
21     update()
22
23
24
25 ##########
26 #  MAIN  #
27 ##########
28
29
30 beginGrfx(800,500)
31 listen()
32 onkey(displayq,"q")
33 onkey(displayQ,"Q")
34 endGrfx()
```



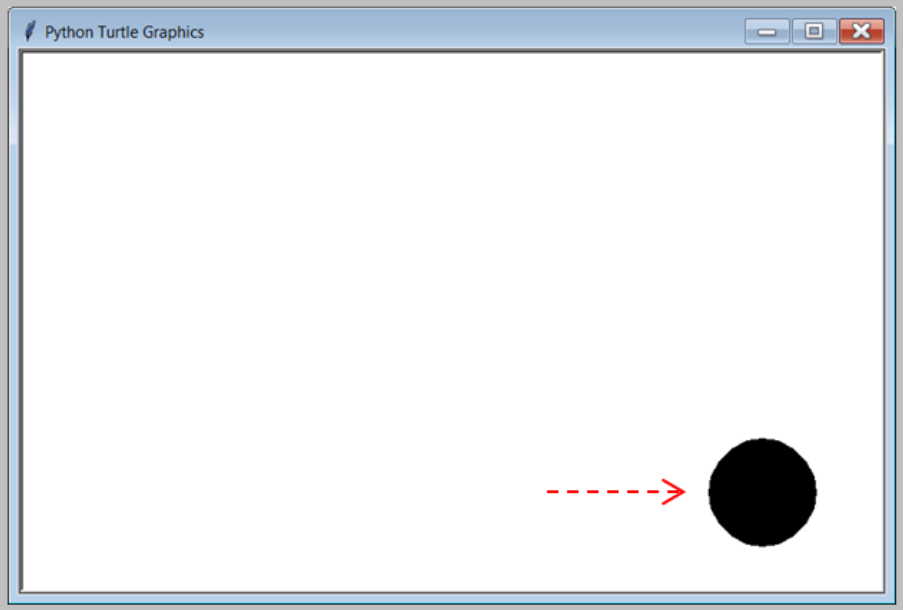
```
1  # KeyEvents02.py
2  # This program demonstrates that the computer
3  # can "listen" for multiple different keys.
4  # It also shows that CAPITAL 'Q' and lowercase 'q'
5  # are treated as different keys.
6
7
8  from Graphics import *
9
10
11 def displayq():
12     clear()
13     setRandomColor()
14     drawString("You pressed lowercase q.",100,150,"Arial",28,"bold")
15     update()
16
17 def displayQ():
18     clear()
19     setRandomColor()
20     drawString("You pressed CAPITAL Q.",100,150,"Arial",28,"bold")
21     update()
22
23
24
25 ##########
26 #  MAIN  #
27 ##########
28
29
30 beginGrfx(800,500)
31 listen()
32 onkey(displayq,"q")
33 onkey(displayQ,"Q")
34 endGrfx()
```



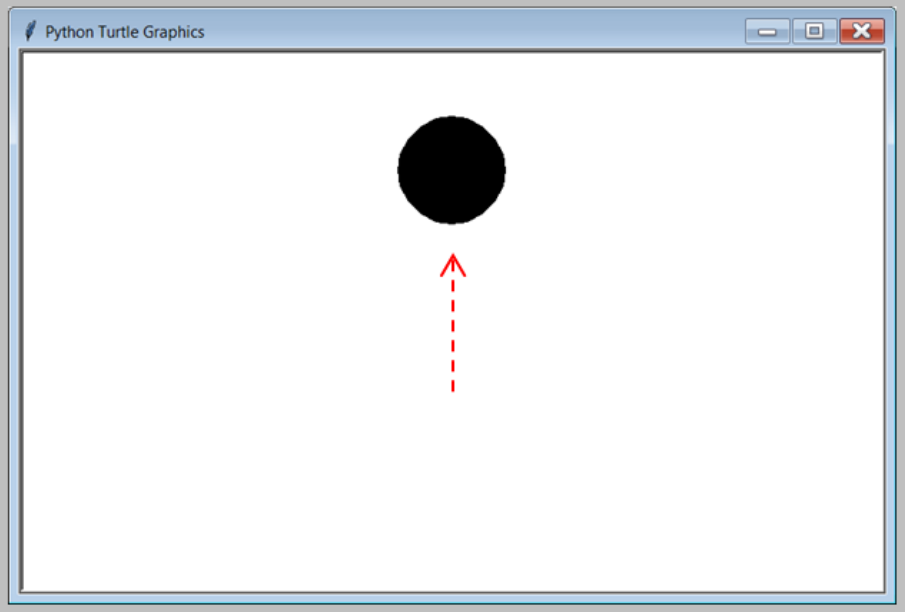
```
# KeyEvents03.py
# This program controls the movement of a
# circle on the screen using the letters:
#     w
#   a s
#     z


from Graphics import *


def moveUp():
    global y
    y -= 10
    clear()
    fillCircle(x,y,50)
    update()

def moveDown():
    global y
    y += 10
    clear()
    fillCircle(x,y,50)
    update()

def moveLeft():
    global x
    x -= 10
    clear()
    fillCircle(x,y,50)
    update()

def moveRight():
    global x
    x += 10
    clear()
    fillCircle(x,y,50)
    update()



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


x = 400
y = 250

beginGrfx(800,500)
listen()
onkey(moveUp,"w")
onkey(moveDown,"z")
onkey(moveLeft,"a")
onkey(moveRight,"s")
fillCircle(x,y,50)
endGrfx()
```



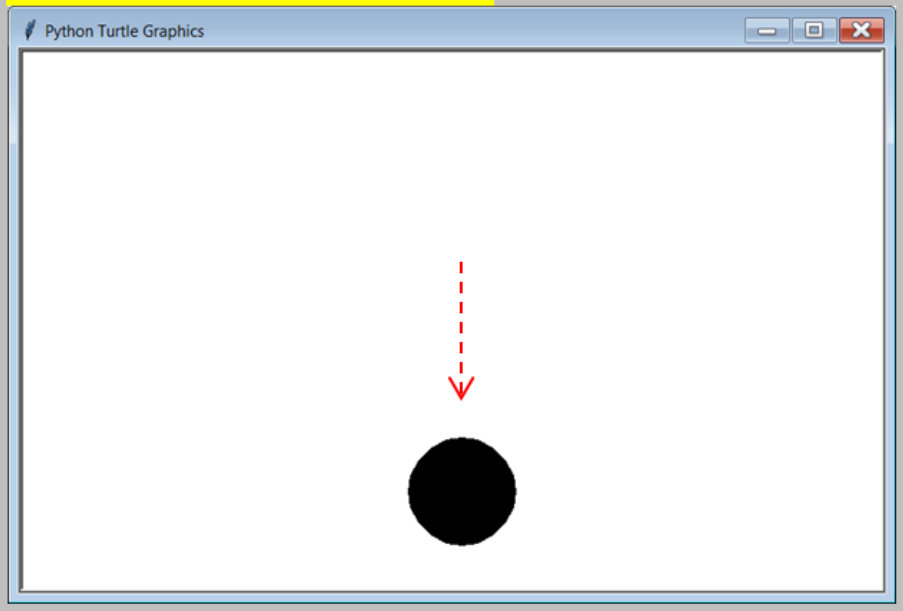
Press '**s**' to move right.



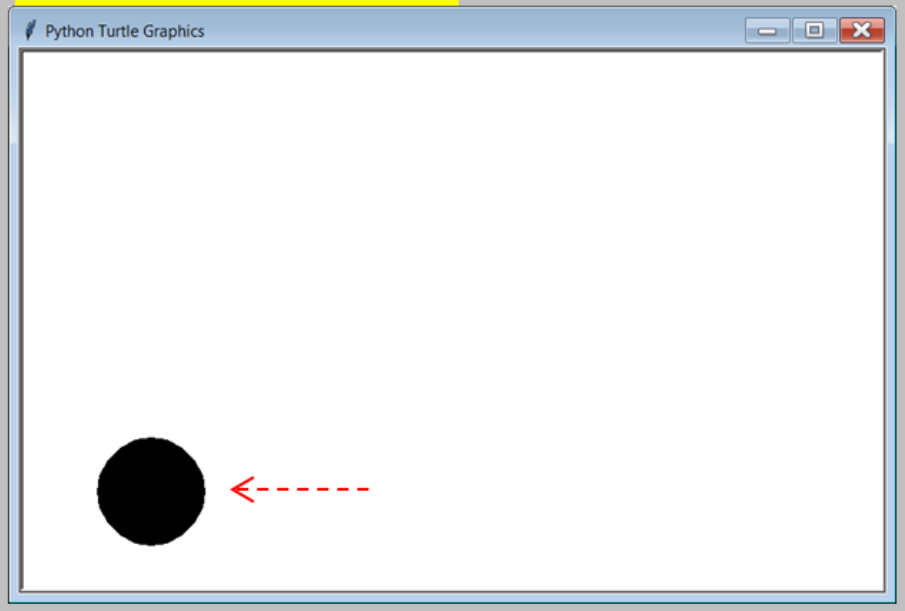
Press '**w**' to move up.



Press '**z**' to move down.

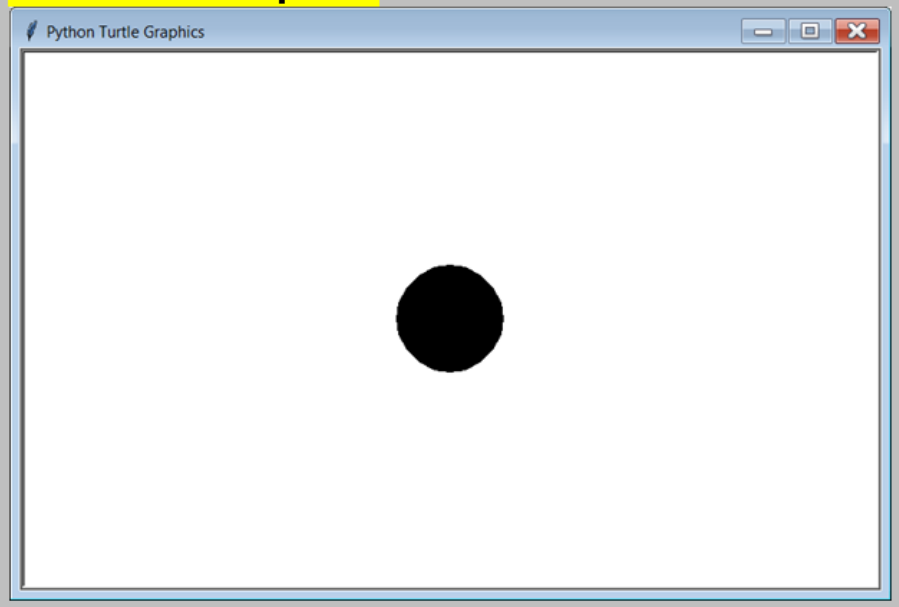


Press '**a**' to move left.

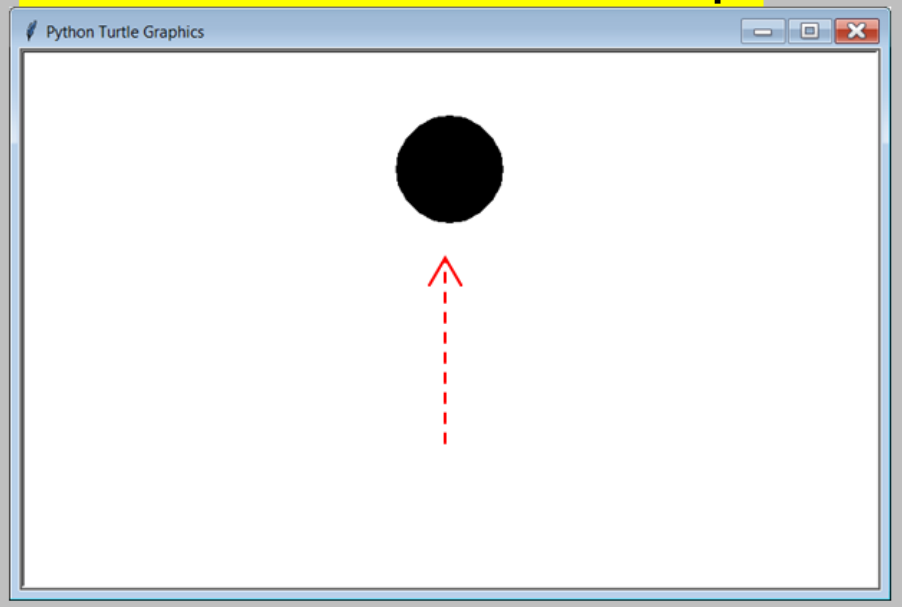


```
# KeyEvents04.py
# This program shows that <onkey> works with
# "Special Keys" as well like the arrow keys,
# as well as "Insert", "Delete" and "Home".


from Graphics import *


def moveUp():
    global y
    y -= 10
    clear()
    fillCircle(x,y,r)
    update()

def moveDown():
    global y
    y += 10
    clear()
    fillCircle(x,y,r)
    update()

def moveLeft():
    global x
    x -= 10
    clear()
    fillCircle(x,y,r)
    update()

def moveRight():
    global x
    x += 10
    clear()
    fillCircle(x,y,r)
    update()

def bigger():
    global r
    r += 10
    clear()
    fillCircle(x,y,r)
    update()

def smaller():
    global r
    r -= 10
    if r < 0:
        r = 0
    clear()
    fillCircle(x,y,r)
    update()

def center():
    global x,y,r
    x = 400
    y = 250
    r = 50
    clear()
    fillCircle(x,y,r)
    update()


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


x = 400
y = 250
r = 50

beginGrfx(800,500)
listen()
onkey(moveUp,"Up")
onkey(moveDown,"Down")
onkey(moveLeft,"Left")
onkey(moveRight,"Right")
onkey(bigger,"Insert")
onkey(smaller,"Delete")
onkey(center,"Home")
fillCircle(x,y,r)
endGrfx()
```

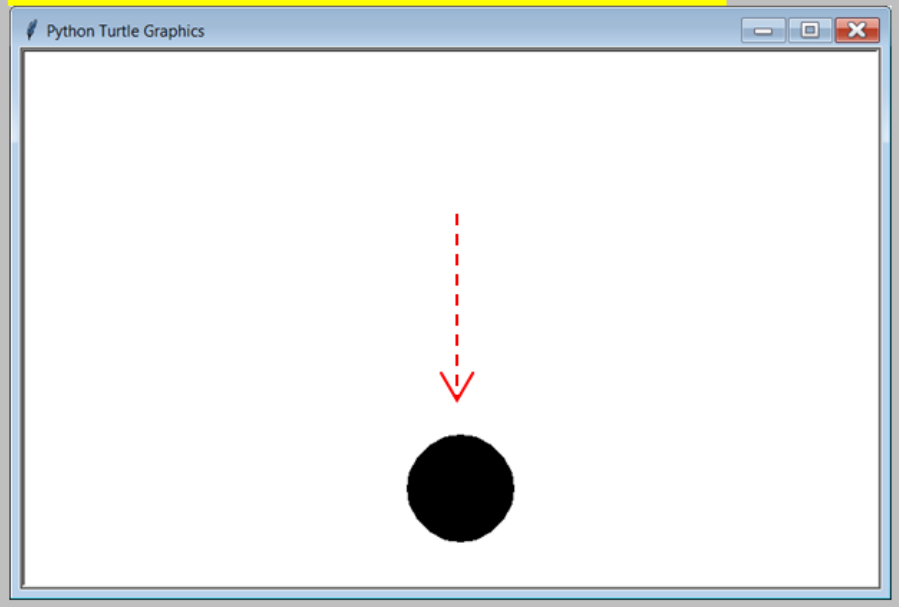
Initial Output:



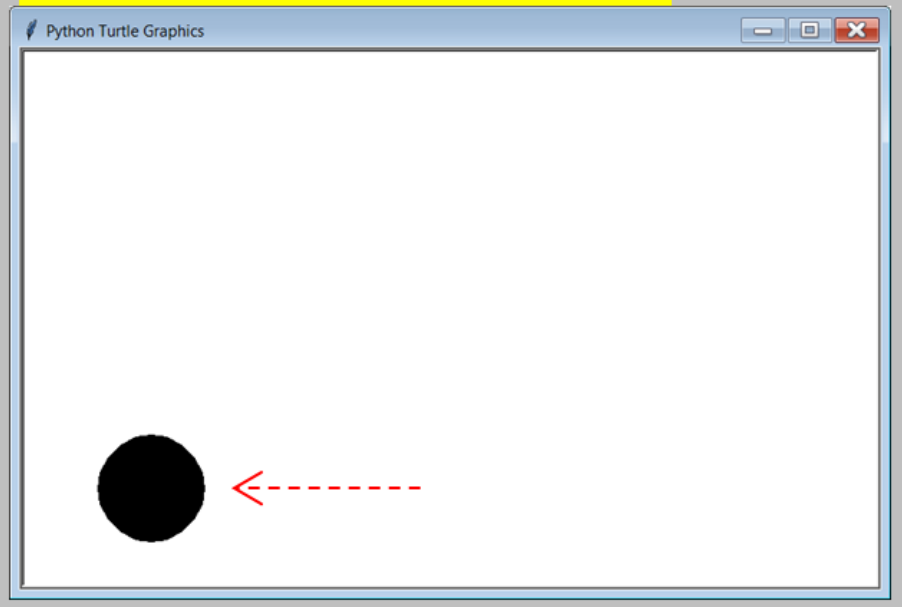
Press the <↑> to move up.



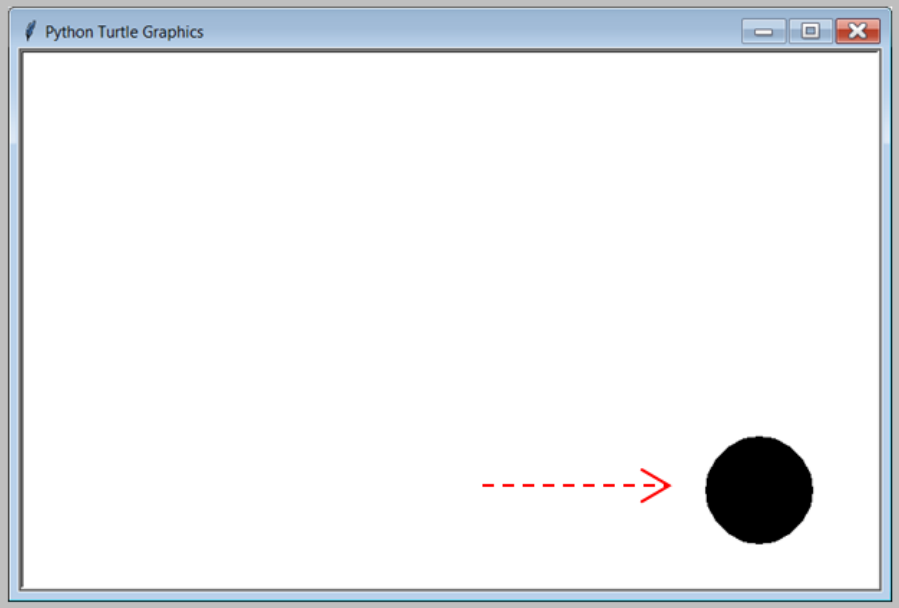
Press <↓> to move down.



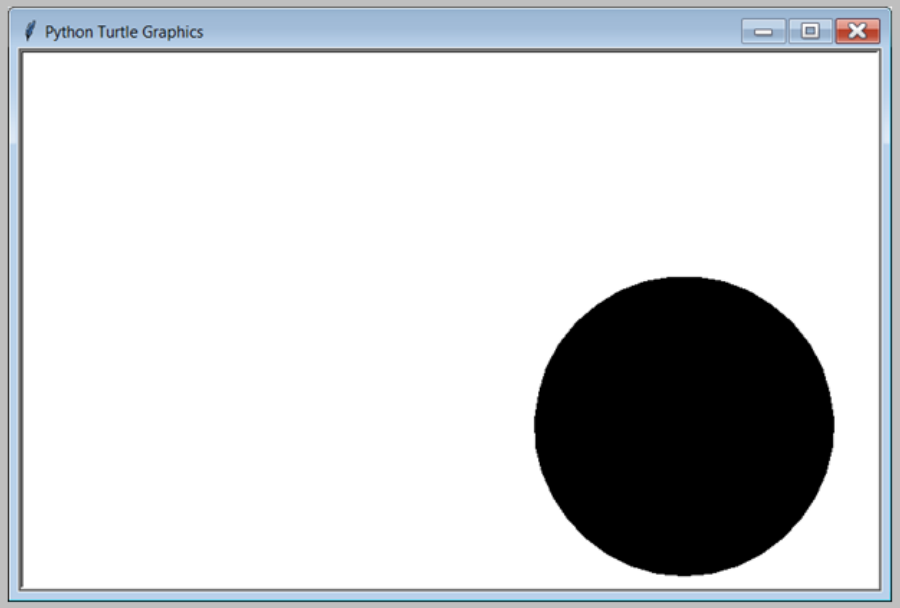
Press <←> to move left.



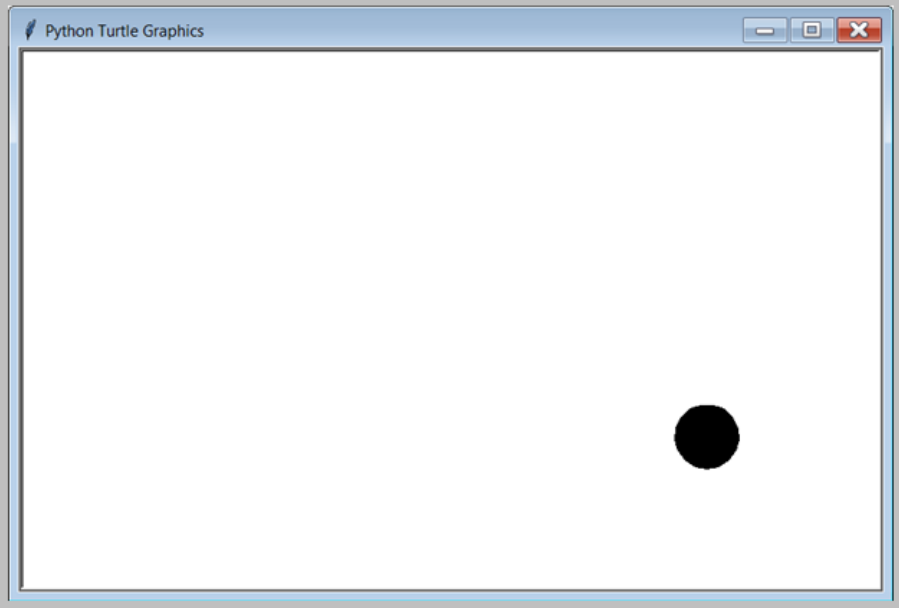
Press <→> to move right.



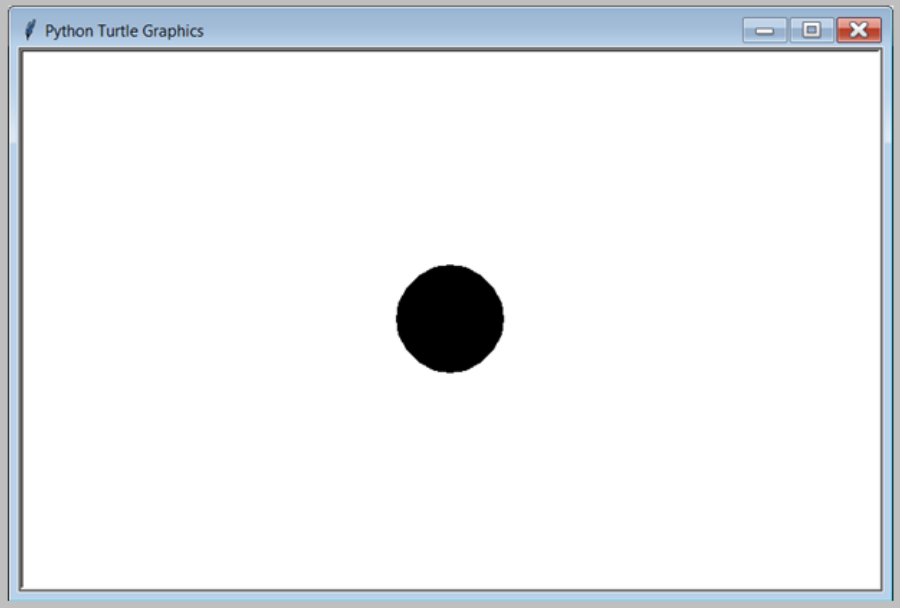
<Insert> makes circle bigger.



<Delete> makes circle smaller.



<Home> resets the screen.



```
1 # KeyEvents05.py
2 # This program replaces most of the <onkey>
3 # commands with <onkeypress>.  This allows
4 # the user to simply hold the key down rather
5 # than having to type it repeatedly.
6
```

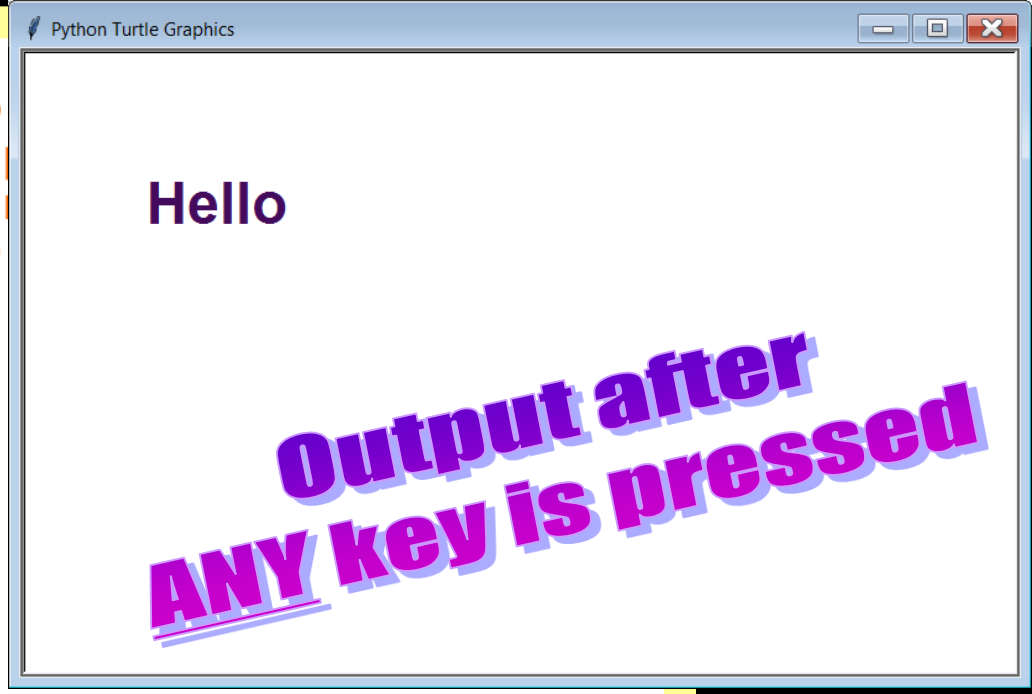
```
74
75 beginGrfx(800,500)
76 listen()
77 onkeypress(moveUp,"Up")
78 onkeypress(moveDown,"Down")
79 onkeypress(moveLeft,"Left")
80 onkeypress(moveRight,"Right")
81 onkeypress(bigger,"Insert")
82 onkeypress(smaller,"Delete")
83 onkey(center,"Home")
84 fillCircle(x,y,r)
85 endGrfx()
```

```
 1 # KeyEvents06.py
 2 # This program demonstrates that the string literal
 3 # "key symbols" used for the "Special Keys" are
 4 # "Case-Sensitive".
    :       :       :       :       :       :       :       :
74 beginGrfx(800,500)
75 listen()
76 onkeypress(moveUp,"up")  # should be "Up"
77 onkeypress(moveDown,"Down")
78 onkeypress(moveLeft,"Left")
79 onkeypress(moveRight,"Right")
80 onkeypress(bigger,"Insert")
81 onkeypress(smaller,"Delete")
82 onkey(center,"Home")
83 fillCircle(x,y,r)
84 endGrfx()
```

```
 ----jGRASP exec: python KeyEvents06.py
Traceback (most recent call last):
  File "KeyEvents06.py", line 76, in <module>
    onkeypress(moveUp,"up")
  File "<string>", line 8, in onkeypress
  File "turtle.py", line 1426, in onkeypress
    self._onkeypress(fun, key)
  File "turtle.py", line 705, in _onkeypress
    self.cv.bind("<KeyPress-%s>" % key, eventfun)
  File "turtle.py", line 416, in bind
    self._canvas.bind(*args, **kwargs)
  File "__init__.py", line 1245, in bind
    return self._bind(('bind', self._w), sequence, func, add)
  File "__init__.py", line 1200, in _bind
    self.tk.call(what + (sequence, cmd))
_tkinter.TclError: bad event type or keysym "up"

 ----jGRASP wedge2: exit code for process is 1.
 ----jGRASP: operation complete.
```

```
 1 # KeyEvents07.py
 2 # This program attempts to use the "space bar"
 3 # to make the computer display "Hello".  It may
 4 # seem to work at first, until you realize that
 5 # typing ANY key will make the computer display
 6 # "Hello".
 7
 8
 9 from Graphics import *
10
11
12 def display():
13     clear()
14     setRandomColor()
15     drawString("Hello",100,150,"Arial",28,"bold")
16     update()
17
18
19
20 ##########
21 #  MAIN  #
22 ##########
23
24
25 beginGrfx(800,500)
26 listen()
27 onkeypress(display," ")
28 endGrfx()
```

```
 1 # KeyEvents07.py
 2 # This program attempts to
 3 # to make the computer dis
 4 # seem to work at first, u
 5 # typing ANY key will make
 6 # "Hello".
 7
 8
 9 from Graphics import *
10
11
12 def display():
13     clear()
14     setRandomColor()
15     drawString("Hello",100,150,"Arial",28,"bold")
16     update()
17
18
19
20 ##########
21 #  MAIN  #
22 ##########
23
24
25 beginGrfx(800,500)
26 listen()
27 onkeypress(display," ")
28 endGrfx()
```



```
1 # KeyEvents07.py
2 # This program corrects the issue of the previous
3 # program by using the string literal key symbol
4 # "space" instead of " ".
    :     :     :     :     :     :     :     :
23 beginGrfx(800,500)
24 listen()
25 onkeypress(display,"space")
26 endGrfx()
```