

Exposure CS 2021 **for CS1**

Chapter 5 Slides

Introduction to Turtle Graphics

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



Section 5.1

Introduction

Turtle Graphics

In this chapter, we will start learning *Turtle Graphics*.

This is a simplistic type of graphics that was actually introduced in the language Logo.

Remember that Logo was designed for young children, so *Turtle Graphics* is fairly simplistic, even in Python.

LOGO



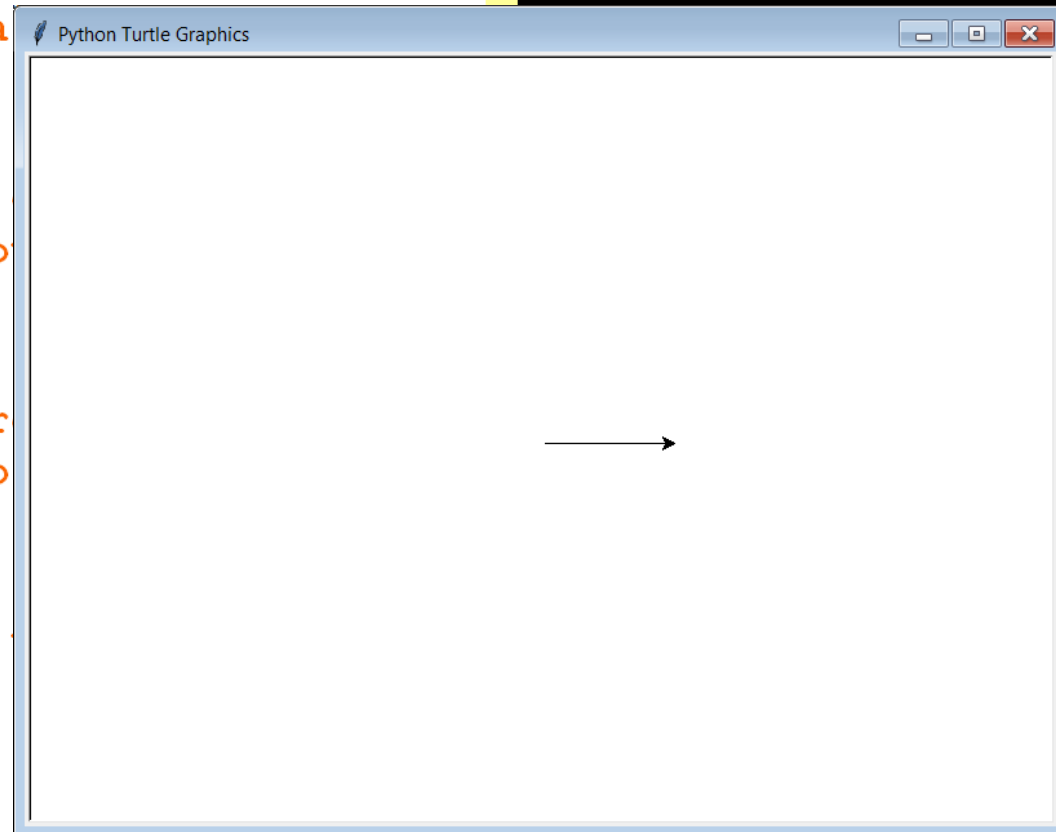
Section 5.2

Importing Libraries and

Turtle Graphics Setup

```
1 # TurtleGraphics01.py
2 # This program introduces "Turtle Graphics" by
3 # importing the <turtle> library and drawing a
4 # single line with the <forward> command.
5
6
7 # Required to have access to
8 # the turtle graphics commands.
9 import turtle
10
11 # Specifies the dimensions of
12 # the Turtle Graphics window.
13 turtle.setup(800,600)
14
15 # Moves the "turtle" forward 100 pixels
16 # and draws a line in the process.
17 turtle.forward(100)
18
19 # Required to "update" the window
20 # when everything is drawn.
21 turtle.update()
22
23 # Required to keep the graphics window
24 # open when the program is finished.
25 turtle.done()
```

```
1 # TurtleGraphics01.py
2 # This program introduces "Turtle Graphics" by
3 # importing the <turtle> library and drawing a
4 # single line with the <forward> command.
5
6
7 # Required to have access to
8 # the turtle graphics comma
9 import turtle
10
11 # Specifies the dimensions
12 # the Turtle Graphics windo
13 turtle.setup(800,600)
14
15 # Moves the "turtle" forwar
16 # and draws a line in the p
17 turtle.forward(100)
18
19 # Required to "update" the
20 # when everything is drawn.
21 turtle.update()
22
23 # Required to keep the graphics window
24 # open when the program is finished.
25 turtle.done()
```



```
1 # TurtleGraphics02.py
2 # This program has the exact same output as the
3 # previous one. By importing the individual
4 # <turtle> library command in this way, the
5 # rest of the code can be less "wordy".
6
7
8 from turtle import setup
9 from turtle import forward
10 from turtle import update
11 from turtle import done
12
13 setup(800, 600)
14 forward(100)
15 update()
16 done()
17
```

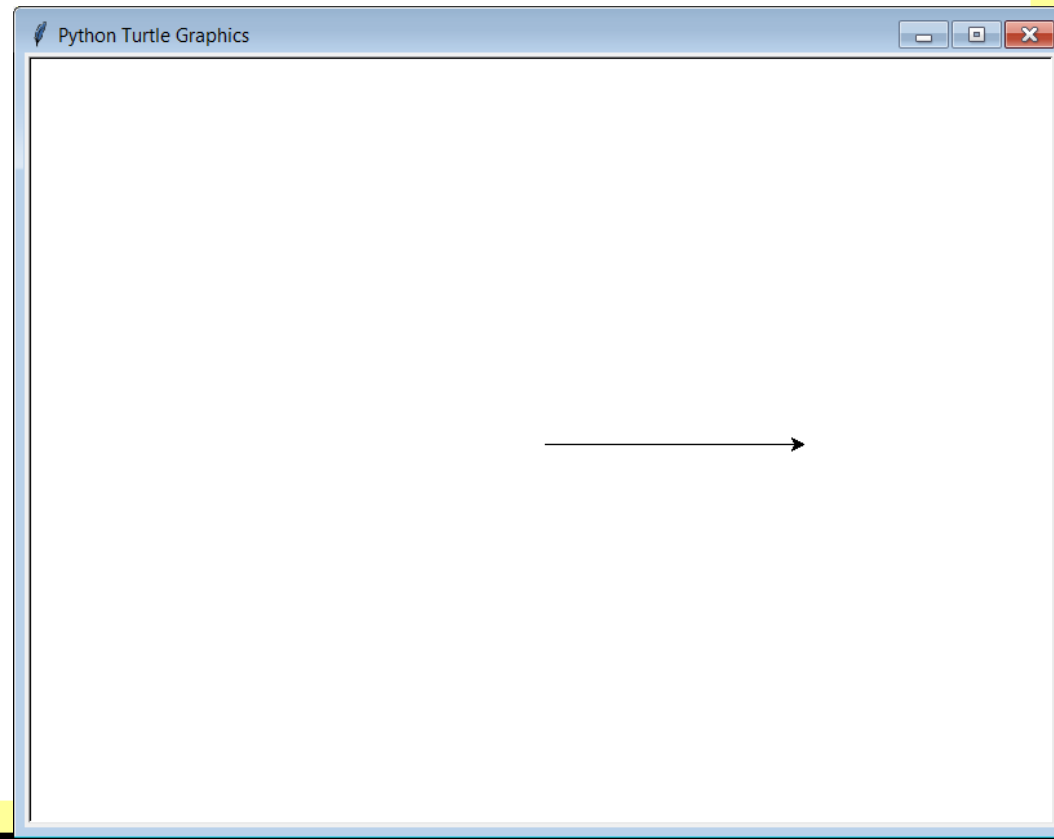
```
1 # TurtleGraphics03.py
2 # This program has the exact same output as the
3 # previous two. By using the <*> "wildcard" we
4 # can <import> all of the <turtle> library commands
5 # at once making the code even less "wordy".
6
7
8 from turtle import *
9
10 setup(800,600)
11 forward(100)
12 update()
13 done()
14
```


Section 5.3

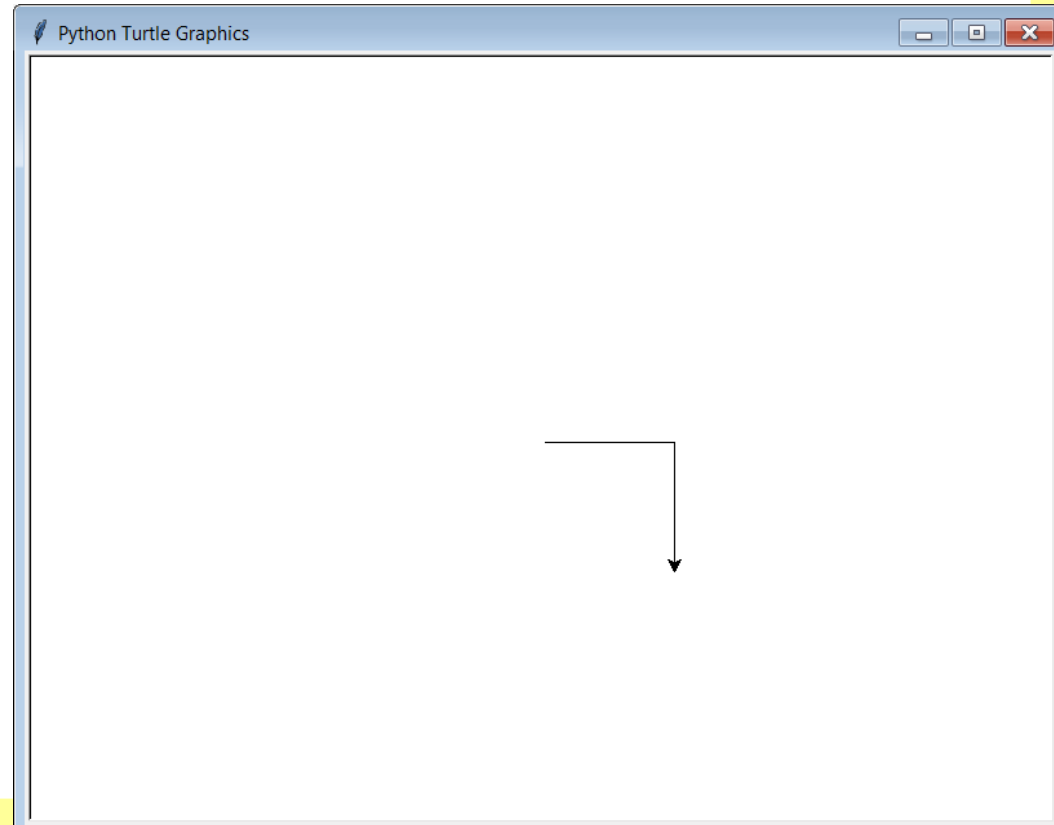
Drawing by Moving &

Turning the Turtle

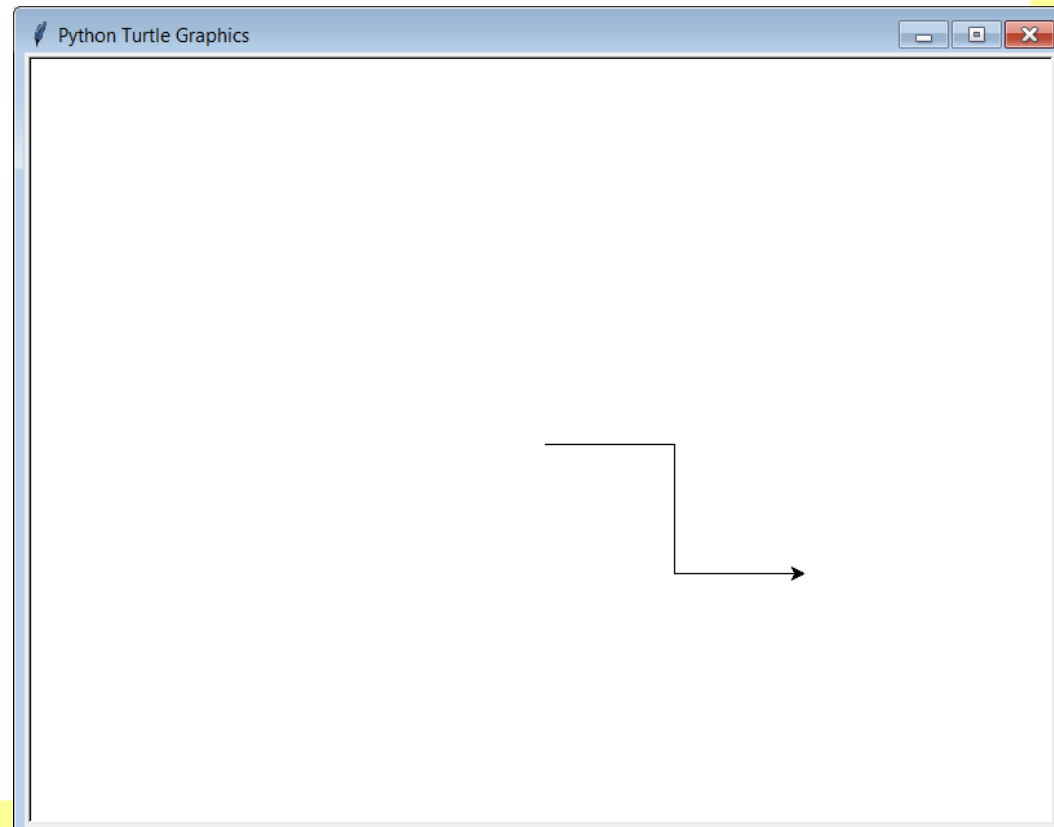
```
1 # TurtleGraphics04.py
2 # This program shows what happens when
3 # <forward> is called twice.
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 forward(100)
11 forward(100)
12
13 update()
14 done()
15
```



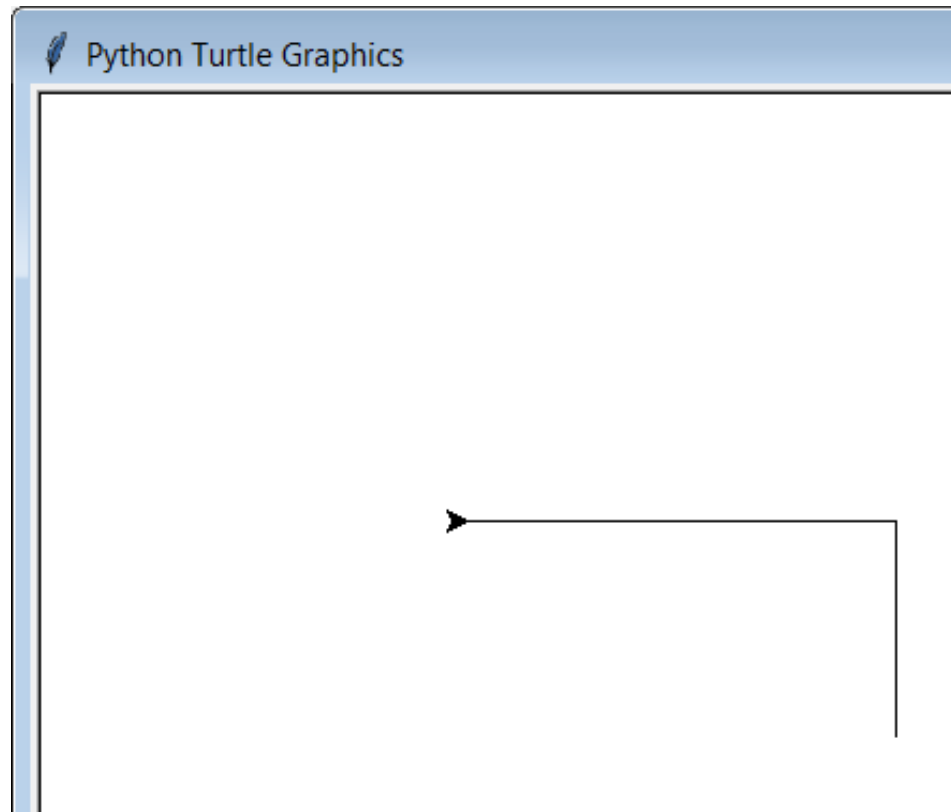
```
1 # TurtleGraphics05.py
2 # This program has the "turtle" make a
3 # 90 degree turn to the "right" before
4 # the second line is drawn.
5
6
7 from turtle import *
8
9 setup(800,600)
10
11 forward(100)
12 right(90)
13 forward(100)
14
15 update()
16 done()
```



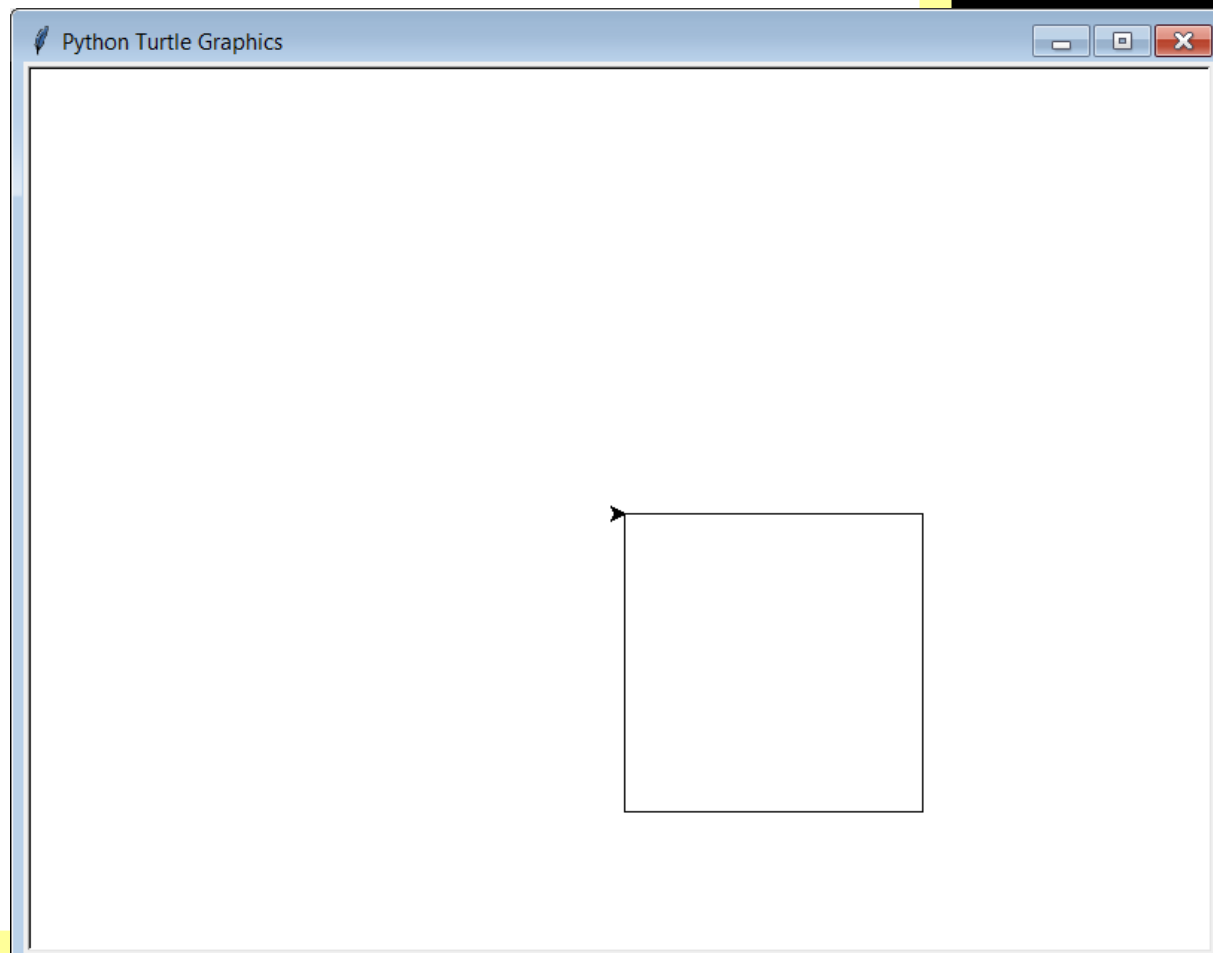
```
1 # TurtleGraphics06.py
2 # This program shows that "left" turns
3 # are also possible.
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 forward(100)
11 right(90)
12 forward(100)
13 left(90)
14 forward(100)
15
16 update()
17 done()
```



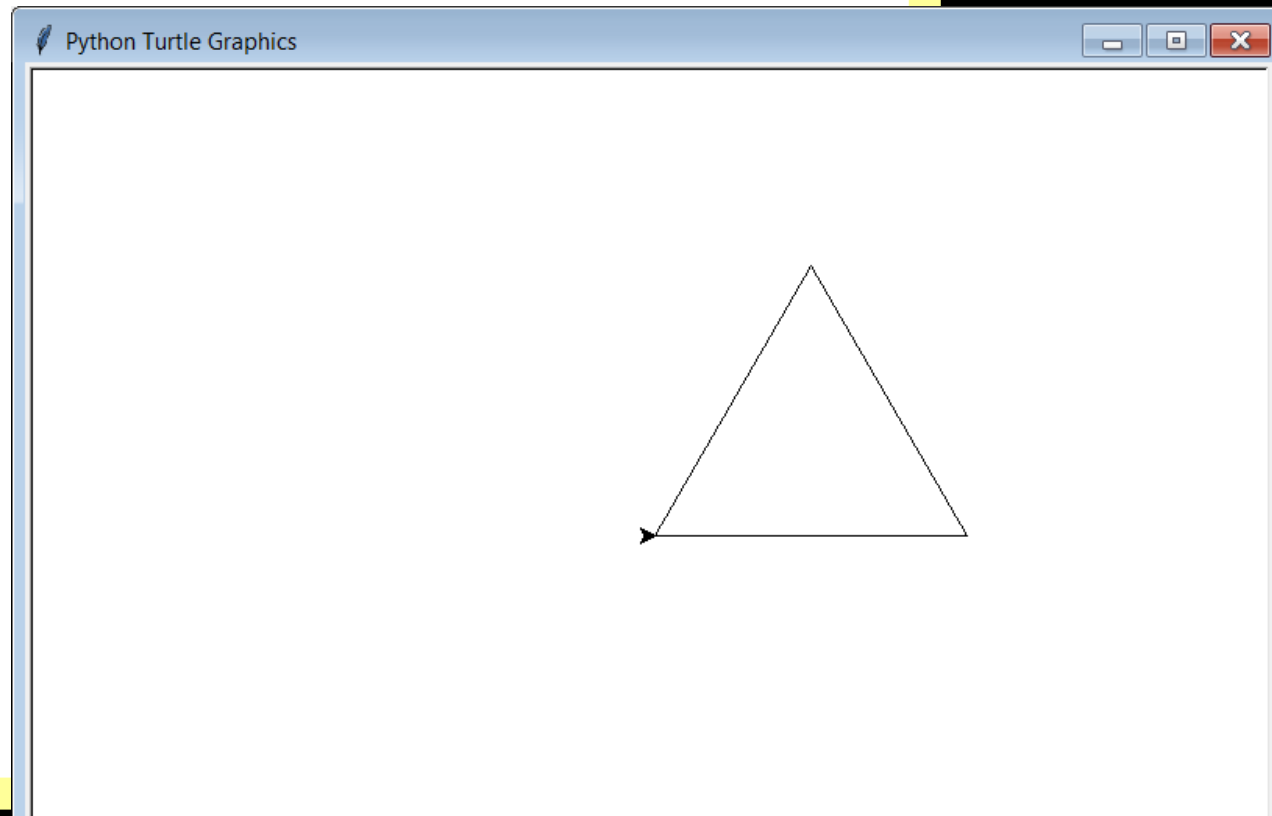
```
1 # TurtleGraphics07.py
2 # This program shows that the "turtle"
3 # can also move "backward".
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 left(90)
11 forward(100)
12 right(90)
13 backward(200)
14
15 update()
16 done()
```



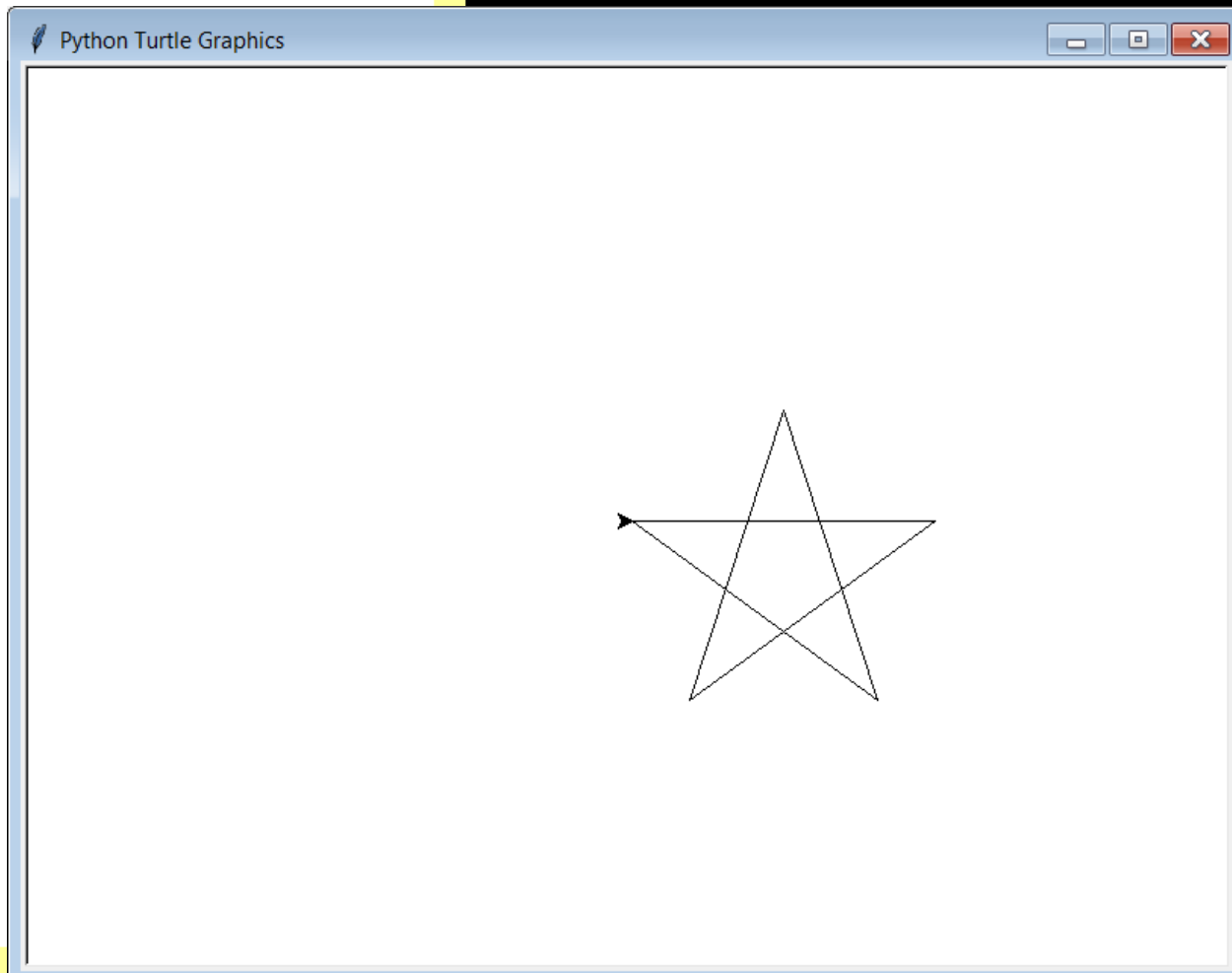
```
1 # TurtleGraphics08.py
2 # This program makes a square by going
3 # "forward" and turning "right" 4 times.
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 forward(200)
11 right(90)
12 forward(200)
13 right(90)
14 forward(200)
15 right(90)
16 forward(200)
17 right(90)
18
19 update()
20 done()
```



```
1 # TurtleGraphics09.py
2 # This program demonstrates that turns do
3 # not need to be 90 degree "right angles".
4 # You can turn any number of degrees.
5 # With 3 turns of 120 degrees, I can
6 # make an equilateral triangle.
7
8
9 from turtle import *
10
11 setup(800,600)
12
13 forward(200)
14 left(120)
15 forward(200)
16 left(120)
17 forward(200)
18 left(120)
19
20 update()
21 done()
```

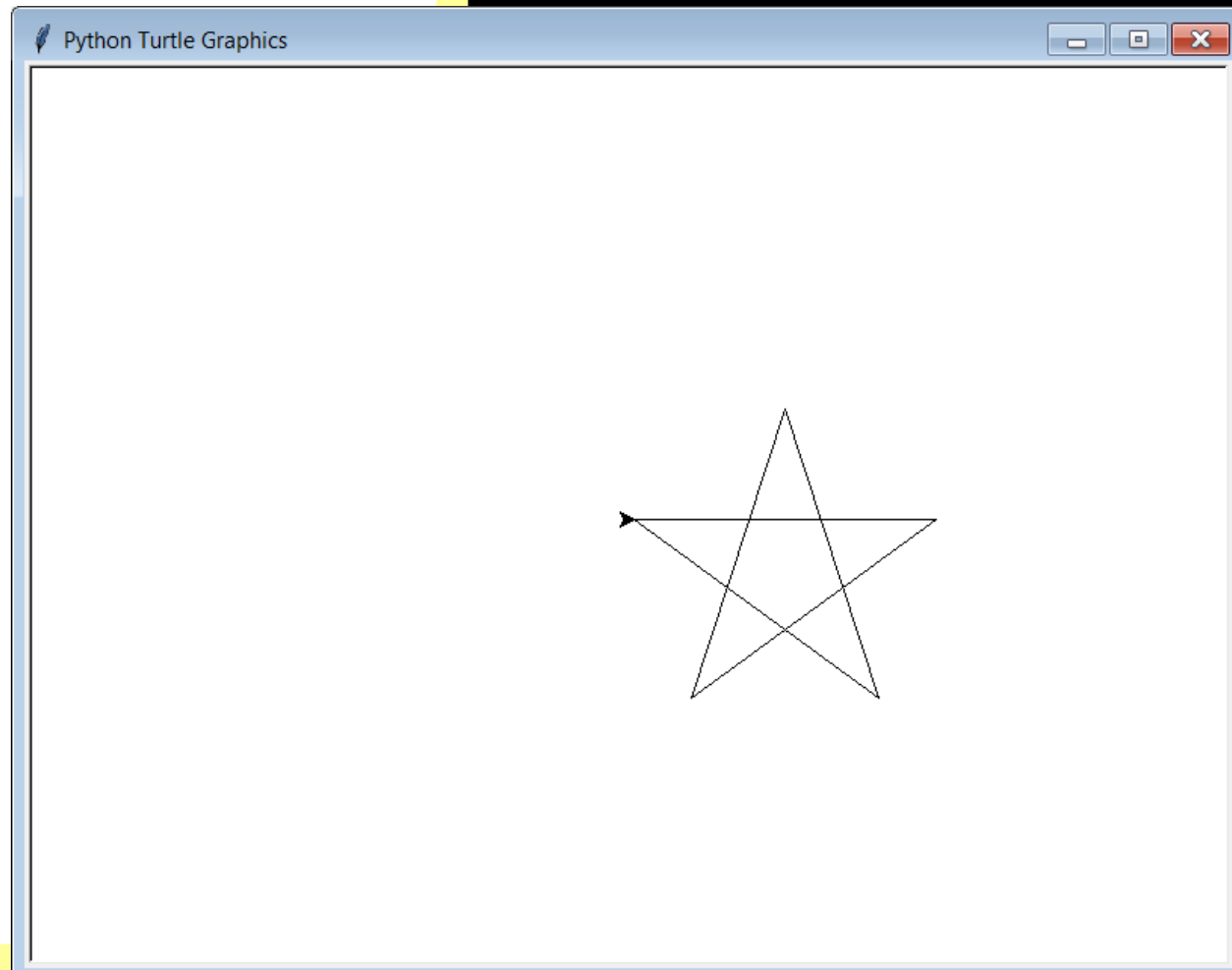


```
1 # TurtleGraphics10.py
2 # Even stars are possible.
3
4
5 from turtle import *
6
7 setup(800,600)
8
9 forward(200)
10 right(144)
11 forward(200)
12 right(144)
13 forward(200)
14 right(144)
15 forward(200)
16 right(144)
17 forward(200)
18 right(144)
19
20 update()
21 done()
```



NOTE: When trying to draw stars or regular polygons, there are mathematical formulas that you can use to determine the exact number of degrees to turn; however, you may find it quicker and easier to just use simple *trial-and-error*.

```
7  setup(800,600)
8
9  forward(200)
10 right(144)
11 forward(200)
12 right(144)
13 forward(200)
14 right(144)
15 forward(200)
16 right(144)
17 forward(200)
18 right(144)
19
20 update()
21 done()
```



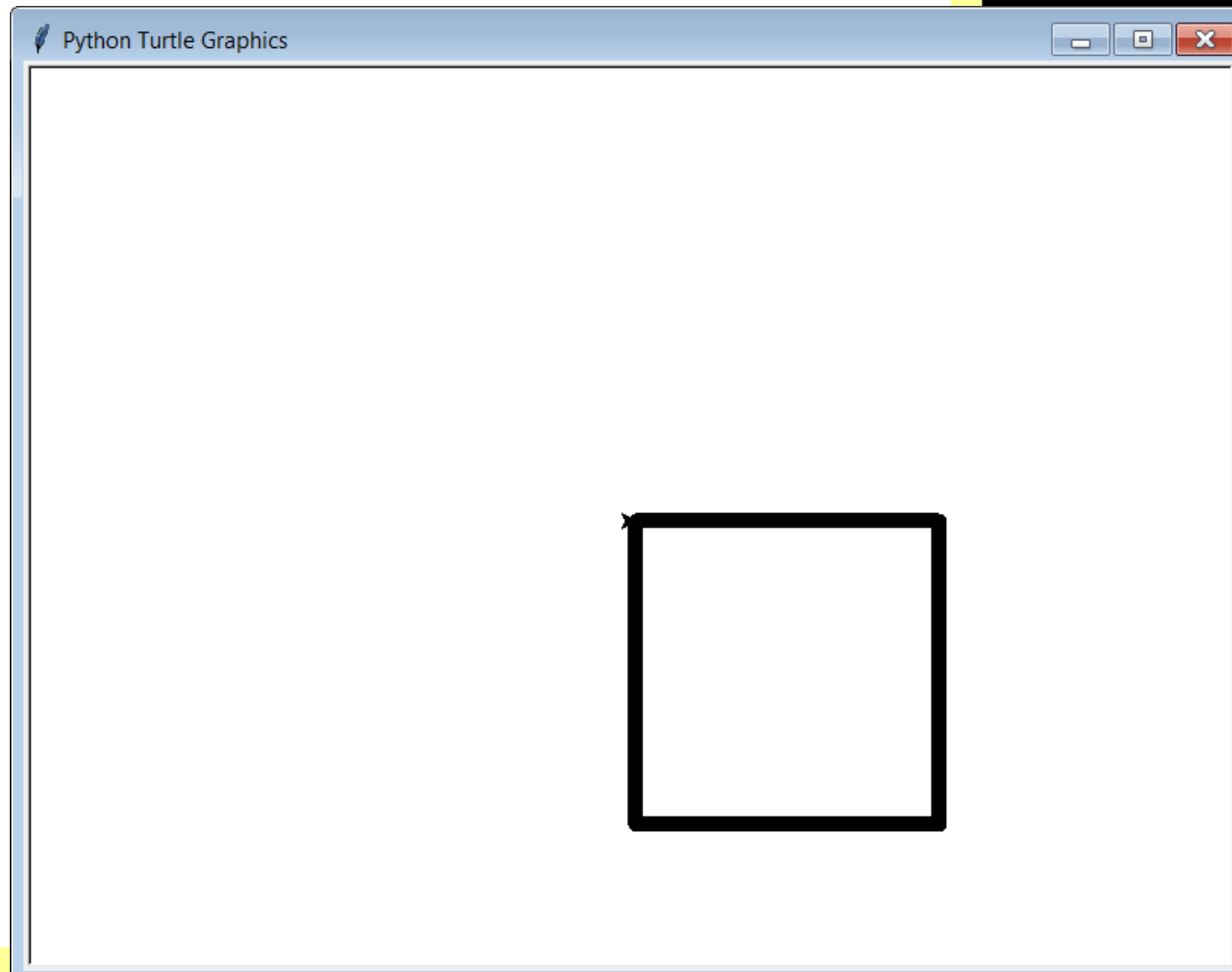
Section 5.4

Drawing Thick

Or Solid Images

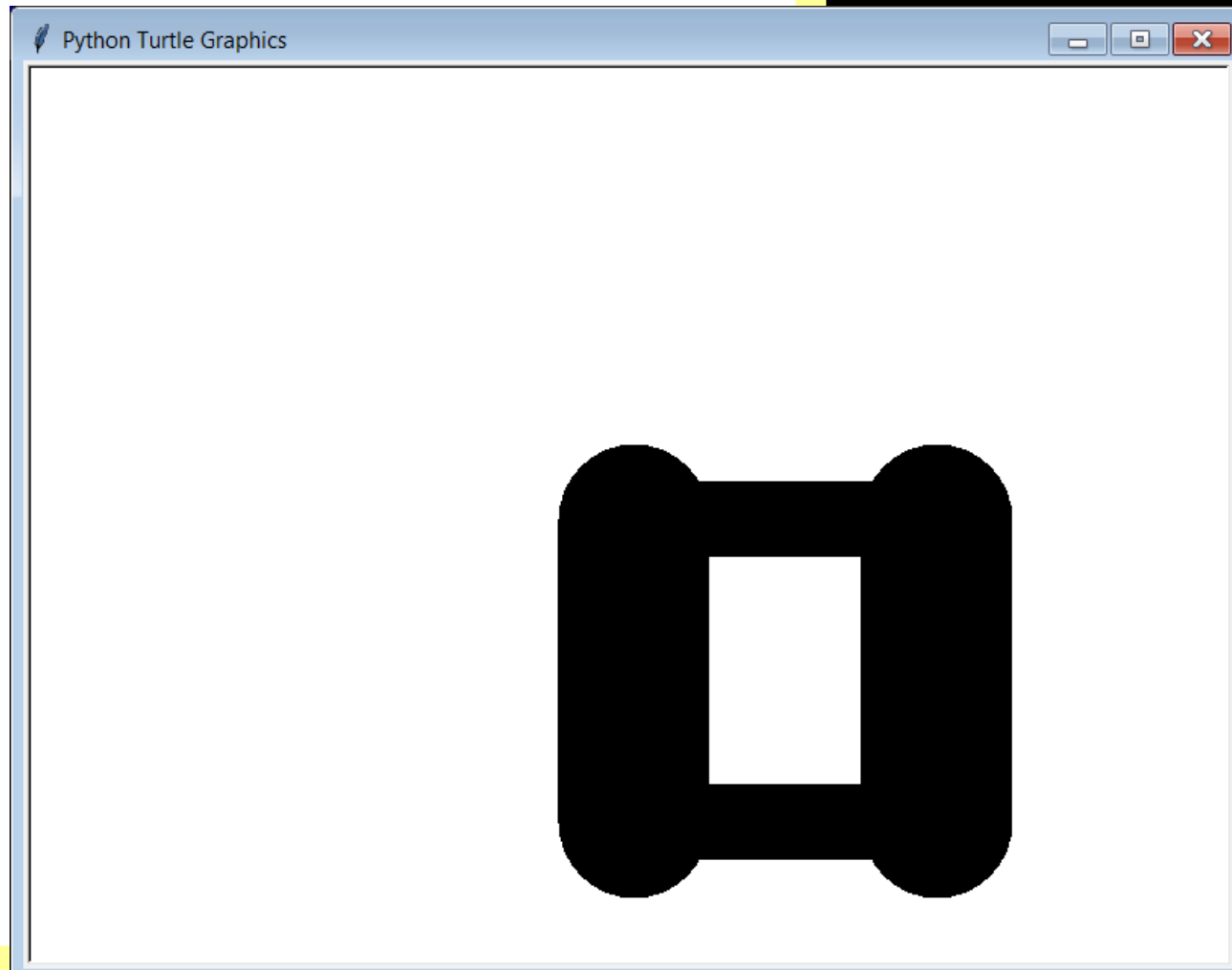
```
1 # TurtleGraphics11.py
2 # This program demonstrates how you can change
3 # the thickness or <width> of the lines.
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 width(10)
11 forward(200)
12 right(90)
13 forward(200)
14 right(90)
15 forward(200)
16 right(90)
17 forward(200)
18 right(90)
19
20 update()
21 done()
```

```
1 # TurtleGraphics11.py
2 # This program demonstrates how you can change
3 # the thickness or <width> of the lines.
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 width(10)
11 forward(200)
12 right(90)
13 forward(200)
14 right(90)
15 forward(200)
16 right(90)
17 forward(200)
18 right(90)
19
20 update()
21 done()
```



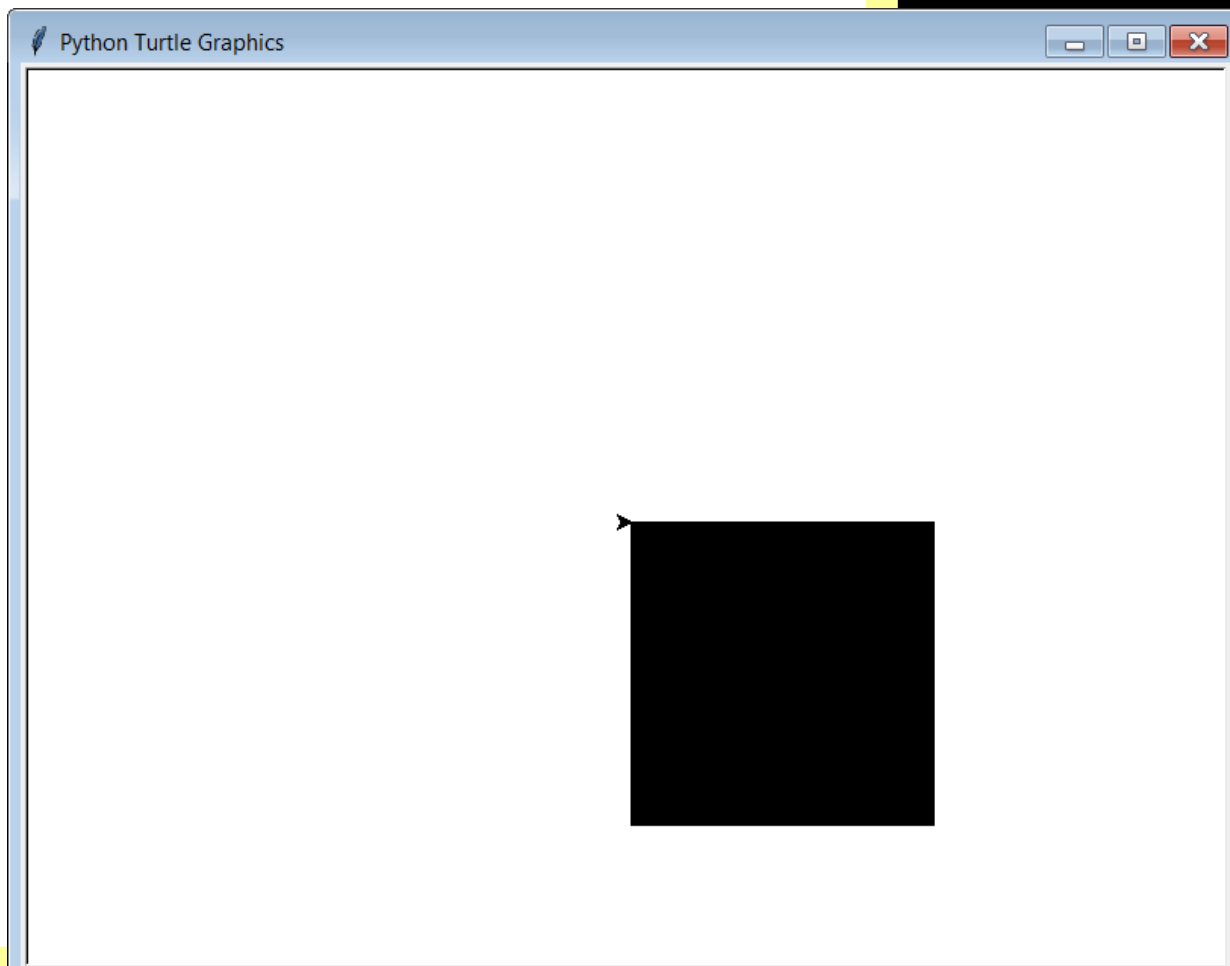
```
1 # TurtleGraphics12.py
2 # This program demonstrates that different
3 # lines can have different widths -- which
4 # can be quite thick.
5
6
7 from turtle import *
8
9 setup(800, 600)
10
11 width(50)
12 forward(200)
13 right(90)
14 width(100)
15 forward(200)
16 right(90)
17 width(50)
18 forward(200)
19 right(90)
20 width(100)
21 forward(200)
22 right(90)
23
24 update()
25 done()
```

```
1 # TurtleGraphics12.py
2 # This program demonstrates that different
3 # lines can have different widths -- which
4 # can be quite thick.
5
6
7 from turtle import *
8
9 setup(800, 600)
10
11 width(50)
12 forward(200)
13 right(90)
14 width(100)
15 forward(200)
16 right(90)
17 width(50)
18 forward(200)
19 right(90)
20 width(100)
21 forward(200)
22 right(90)
23
24 update()
25 done()
```

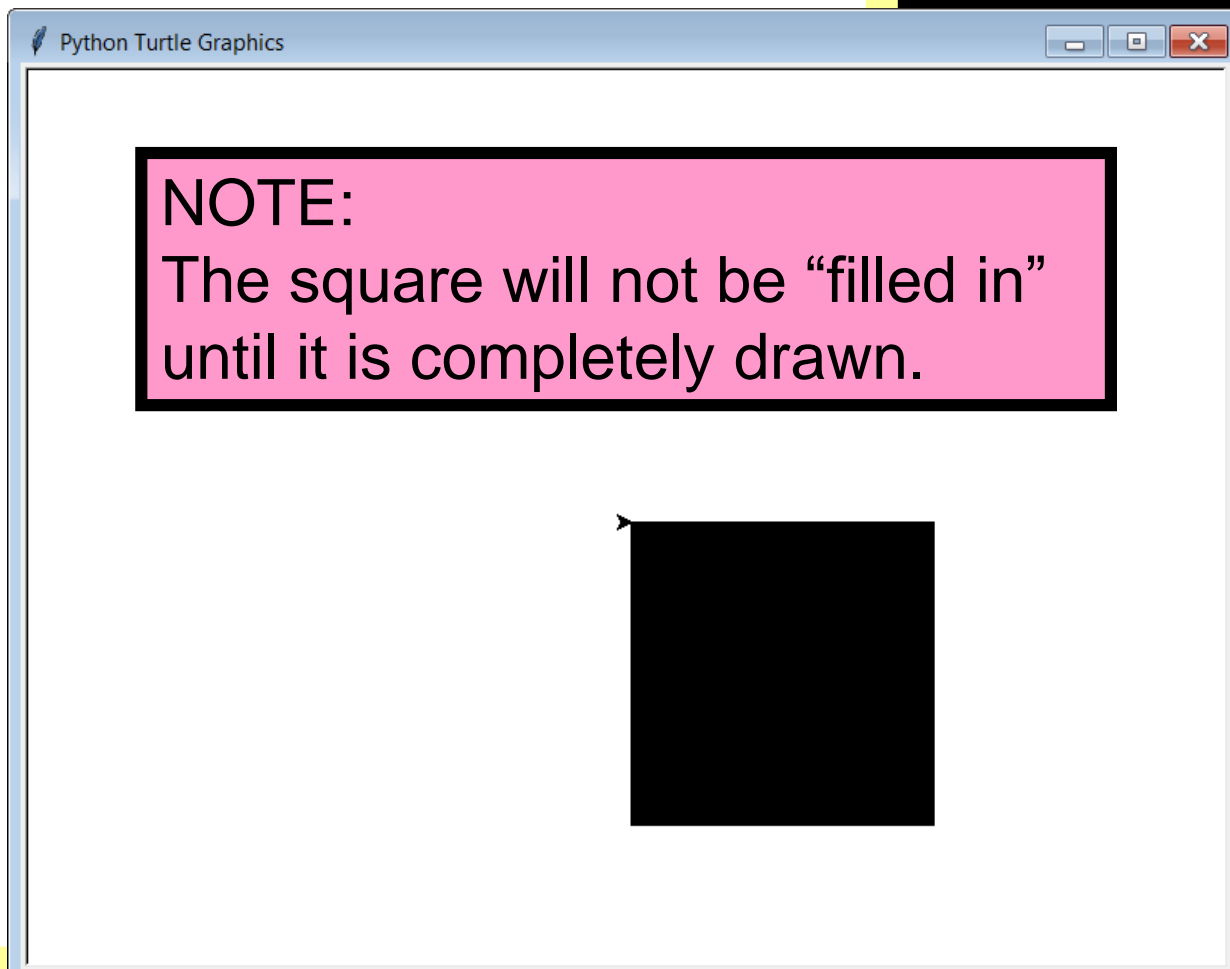


```
1 # TurtleGraphics13.py
2 # This program demonstrates that you can
3 # "fill" enclosed shapes.
4
5 from turtle import *
6
7 setup(800,600)
8
9 begin_fill()
10 forward(200)
11 right(90)
12 forward(200)
13 right(90)
14 forward(200)
15 right(90)
16 forward(200)
17 right(90)
18 end_fill()
19
20 update()
21 done()
```

```
1 # TurtleGraphics13.py
2 # This program demonstrates that you can
3 # "fill" enclosed shapes.
4
5 from turtle import *
6
7 setup(800,600)
8
9 begin_fill()
10 forward(200)
11 right(90)
12 forward(200)
13 right(90)
14 forward(200)
15 right(90)
16 forward(200)
17 right(90)
18 end_fill()
19
20 update()
21 done()
```




```
1 # TurtleGraphics13.py
2 # This program demonstrates that you can
3 # "fill" enclosed shapes.
4
5 from turtle import *
6
7 setup(800,600)
8
9 begin_fill()
10 forward(200)
11 right(90)
12 forward(200)
13 right(90)
14 forward(200)
15 right(90)
16 forward(200)
17 right(90)
18 end_fill()
19
20 update()
21 done()
```

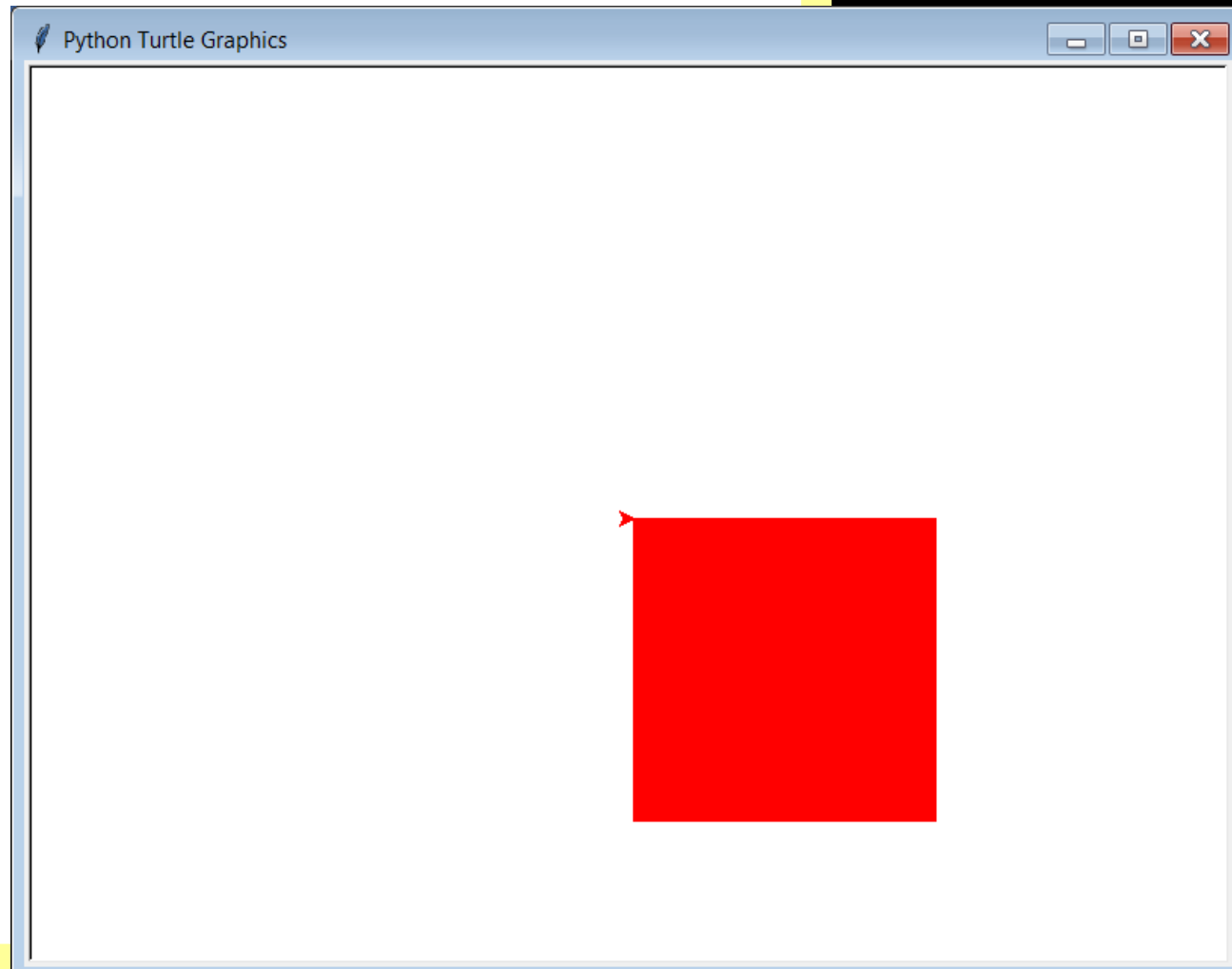


Section 5.5

Adding Color

```
1 # TurtleGraphics14.py
2 # This program demonstrates that you can
3 # draw in other colors besides "black".
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 color("red")
11 begin_fill()
12 forward(200)
13 right(90)
14 forward(200)
15 right(90)
16 forward(200)
17 right(90)
18 forward(200)
19 right(90)
20 end_fill()
21
22 update()
23 done()
```

```
1 # TurtleGraphics14.py
2 # This program demonstrates that you can
3 # draw in other colors besides "black".
4
5
6 from turtle import *
7
8 setup(800,600)
9
10 color("red")
11 begin_fill()
12 forward(200)
13 right(90)
14 forward(200)
15 right(90)
16 forward(200)
17 right(90)
18 forward(200)
19 right(90)
20 end_fill()
21
22 update()
23 done()
```

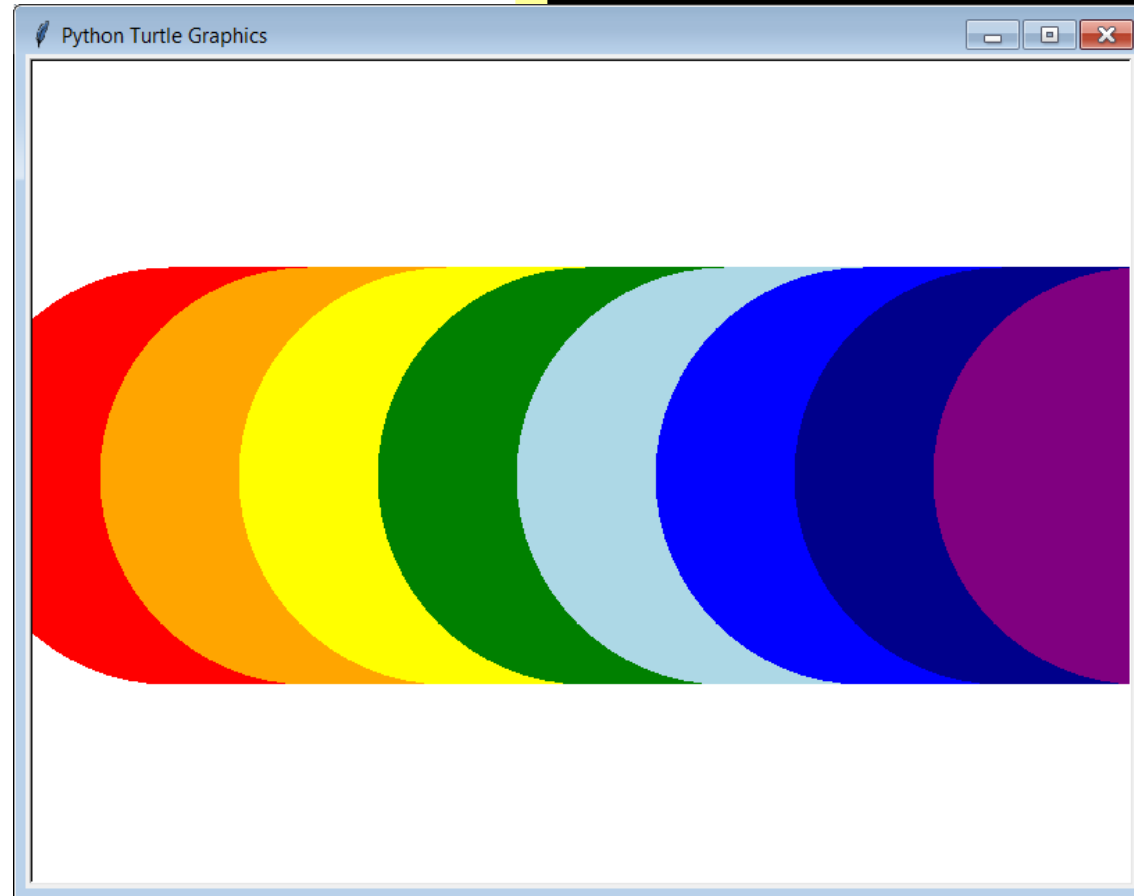


```
1 # TurtleGraphics15.py
2 # This program demonstrates several colors.
3
4 from turtle import *
5
6 setup(800, 600)
7
8 backward(300)
9 width(300)
10 color("red")
11 forward(100)
12 color("orange")
13 forward(100)
14 color("yellow")
15 forward(100)
16 color("green")
17 forward(100)
18 color("light blue")
19 forward(100)
20 color("blue")
21 forward(100)
22 color("dark blue")
23 forward(100)
24 color("purple")
25 forward(100)
26
27 update()
28 done()
```

```
1 # TurtleGraphics15.py
2 # This program demonstrates several colors.
3
4 from turtle import *
5
6 setup(800, 600)
7
8 backward(300)
9 width(300)
10 color("red")
11 forward(100)
12 color("orange")
13 forward(100)
14 color("yellow")
15 forward(100)
16 color("green")
17 forward(100)
18 color("light blue")
19 forward(100)
20 color("blue")
21 forward(100)
22 color("dark blue")
23 forward(100)
24 color("purple")
25 forward(100)
26
27 update()
28 done()
```



```
1 # TurtleGraphics15.py
2 # This program demonstrates several colors.
3
4 from turtle import *
5
6 setup(800, 600)
7
8 backward(300)
9 width(300)
10 color("red")
11 forward(100)
12 color("orange")
13 forward(100)
14 color("yellow")
15 forward(100)
16 color("green")
17 forward(100)
18 color("light blue")
19 forward(100)
20 color("blue")
21 forward(100)
22 color("dark blue")
23 forward(100)
24 color("purple")
25 forward(100)
26
27 update()
28 done()
```



NOTE:

Python has 140 different colors.
These are the same colors used
in HTML to create Webpages.

Section 5.6

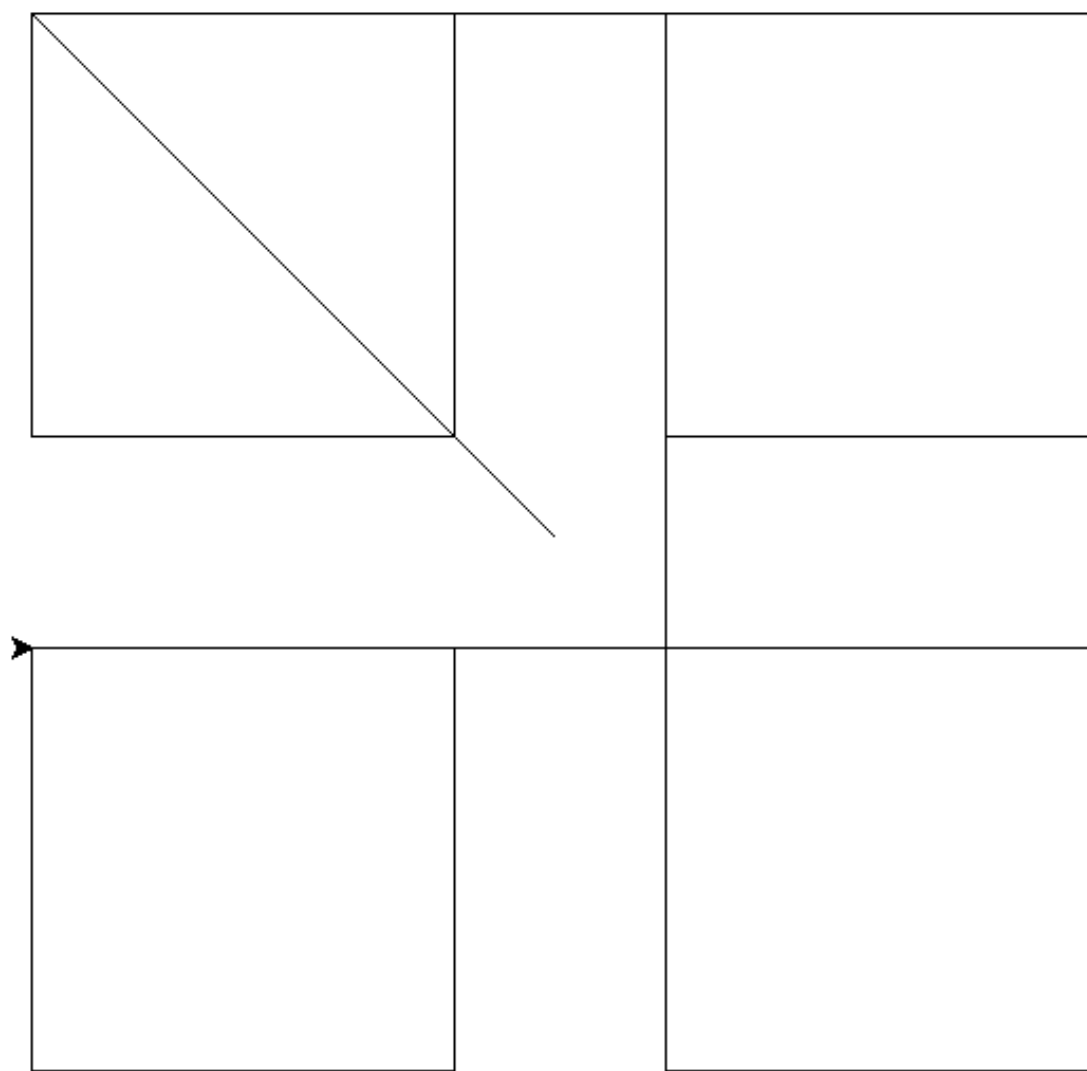
Lifting the Pen


```
1 # TurtleGraphics16.py
2 # This program tries to draw 4 separate squares.
3 # Before each square is drawn; the "turtle" moves
4 # to one of the corners of the graphics window.
5 # The problem is the turtle is always drawing
6 # a several lines are drawn that we do not want.
7
8
9 from turtle import *
10
11 setup(800,600)
12
13 left(135)      # face North-West
14 forward(350)   # Move to top-left corner
15 right(135)     # face East again
16
17 # draw square
18 forward(200)
19 right(90)
20 forward(200)
21 right(90)
22 forward(200)
23 right(90)
24 forward(200)
25 right(90)
26
27 forward(300)   # Move to top-right corner
28
29 # draw square
30 forward(200)
31 right(90)
32 forward(200)
33 right(90)
```

```

1 # TurtleGraphics16.py
2 # This program tries to draw a square
3 # Before each square is drawn, the turtle moves
4 # to one of the corners of the square
5 # The problem is the turtle moves to the corner
6 # a several lines are drawn
7
8
9 from turtle import *
10
11 setup(800,600)
12
13 left(135)      # face North
14 forward(350)   # Move to top-left corner
15 right(135)     # face East
16
17 # draw square
18 forward(200)
19 right(90)
20 forward(200)
21 right(90)
22 forward(200)
23 right(90)
24 forward(200)
25 right(90)
26
27 forward(300)   # Move to top-right corner
28
29 # draw square
30 forward(200)
31 right(90)
32 forward(200)
33 right(90)
34 forward(200)
35 right(90)
36 forward(200)
37 right(90)
38
39 right(90)      # face South
40 forward(300)   # Move to bottom-right corner
41 left(90)       # face East again
42
43 # draw square
44 forward(200)
45 right(90)
46 forward(200)
47 right(90)
48 forward(200)
49 right(90)
50 forward(200)
51 right(90)
52
53 backward(300)  # Move to bottom-left corner
54
55 # draw square
56 forward(200)
57 right(90)
58 forward(200)
59 right(90)
60 forward(200)
61 right(90)
62 forward(200)
63 right(90)
64
65 update()
66 done()

```



```
1 # TurtleGraphics17.py
2 # This program improves on the previous program
3 # by adding strategic <penup> and <pendown>
4 # commands at the appropriate places.
5 # Now, only the 4 squares are drawn.
6
7
8 from turtle import *
9
10 setup(800,600)
11
12 penup()
13 left(135)      # face North-West
14 forward(350)   # Move to top-left corner
15 right(135)     # face East again
16 pendown()
17
18 # draw square
19 forward(200)
20 right(90)
21 forward(200)
22 right(90)
23 forward(200)
24 right(90)
25 forward(200)
26 right(90)
27
28 penup()
29 forward(300)   # Move to top-right corner
30 pendown()
31
32 # draw square
33 forward(200)
34 right(90)
35 forward(200)
36 right(90)
37 forward(200)
```

```

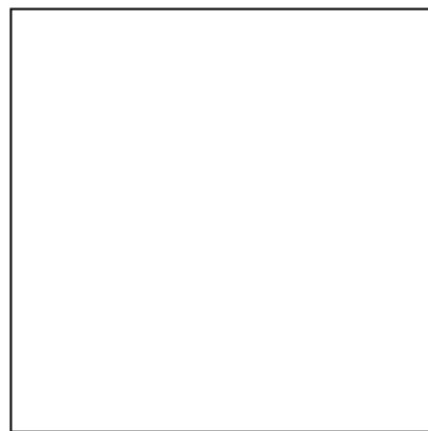
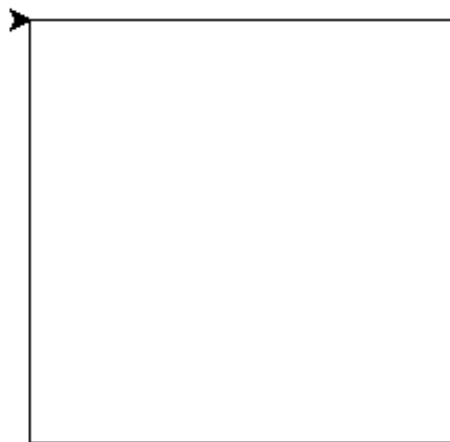
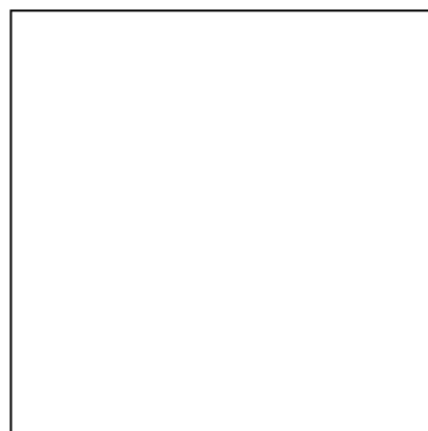
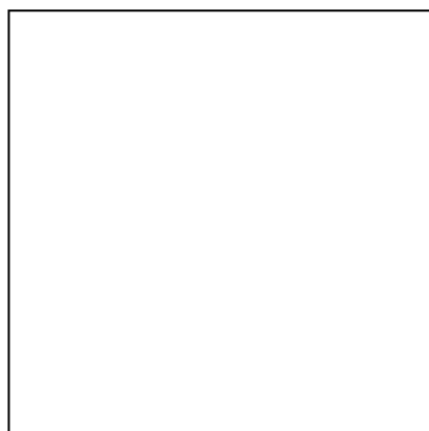
1 # TurtleGraphics17.py
2 # This program improves on the previ
3 # by adding strategic <penup> and <p
4 # commands at the appropriate places
5 # Now, only the 4 squares are drawn.
6
7
8 from turtle import *
9
10 setup(800,600)
11
12 penup()
13 left(135)      # face North-West
14 forward(350)   # Move to top-left cor
15 right(135)     # face East again
16 pendown()
17
18 # draw square
19 forward(200)
20 right(90)
21 forward(200)
22 right(90)
23 forward(200)
24 right(90)
25 forward(200)
26 right(90)
27
28 penup()
29 forward(300)   # Move to top-right co
30 pendown()
31
32 # draw square
33 forward(200)
34 right(90)
35 forward(200)
36 right(90)
37 forward(200)

```

```

38 right(90)
39 forward(200)
40 right(90)
41
42 penup()
43 right(90)      # face South
44 forward(300)   # Move to bottom-right corner
45 left(90)       # face East again
46 pendown()
47
48 # draw square
49 forward(200)
50 right(90)
51 forward(200)
52 right(90)
53 forward(200)
54 right(90)
55 forward(200)
56 right(90)
57
58 penup()
59 backward(300)  # Move to bottom-left corner
60 pendown()
61
62 # draw square
63 forward(200)
64 right(90)
65 forward(200)
66 right(90)
67 forward(200)
68 right(90)
69 forward(200)
70 right(90)
71
72 update()
73 done()
74

```



Section 5.7

clearing the
Window

```

1 # TurtleGraphics18.py
2 # This program puts a <clear> command after each
3 # square is drawn. Now we only see one square
4 # (briefly) at a time.
5
6 from turtle import *
7
8 setup(800,600)
9
10 penup()
11 left(135)      # face North-West
12 forward(350)   # Move to top-left corner
13 right(135)     # face East again
14 pendown()
15
16 # draw square
17 forward(200)
18 right(90)
19 forward(200)
20 right(90)
21 forward(200)
22 right(90)
23 forward(200)
24 right(90)
25
26 clear()
27
28 penup()
29 forward(300)   # Move to top-right corner
30 pendown()
31
32 # draw square
33 forward(200)
34 right(90)
35 forward(200)
36 right(90)
37 forward(200)
38 right(90)
39 forward(200)
40 right(90)
41

```

```

42 clear()
43
44 penup()
45 right(90)      # face South
46 forward(300)   # Move to bottom-right corner
47 left(90)       # face East again
48 pendown()
49
50 # draw square
51 forward(200)
52 right(90)
53 forward(200)
54 right(90)
55 forward(200)
56 right(90)
57 forward(200)
58 right(90)
59
60 clear()
61
62 penup()
63 backward(300)  # Move to bottom-left corner
64 pendown()
65
66 # draw square
67 forward(200)
68 right(90)
69 forward(200)
70 right(90)
71 forward(200)
72 right(90)
73 forward(200)
74 right(90)
75
76 clear()
77
78 update()
79 done()

```


NOTE:

I cannot really display the output of this program as each square is erased the moment it is drawn. You really need to execute this program on your computer to see and appreciate the output.



```

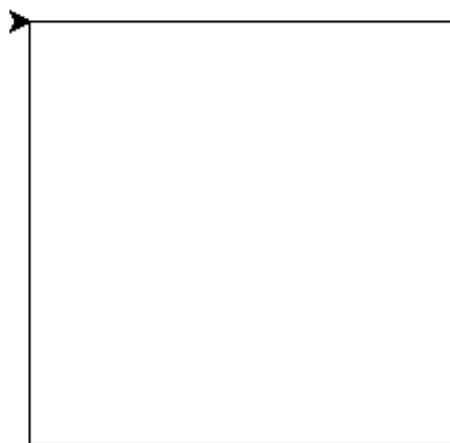
1 # TurtleGraphics19.py
2 # This program imports the <sleep> command from
3 # the <time> library which allows you to make
4 # the turtle pause or "sleep" for a specified
5 # number of seconds. Now each square stays on
6 # the screen for a full second before it is erased
7
8
9 from turtle import *
10 from time import sleep
11
12 setup(800,600)
13
14 penup()
15 left(135)      # face North-West
16 forward(350)   # Move to top-left corner
17 right(135)     # face East again
18 pendown()
19
20 # draw square
21 forward(200)
22 right(90)
23 forward(200)
24 right(90)
25 forward(200)
26 right(90)
27 forward(200)
28 right(90)
29
30 sleep(1)
31 clear()
32
33 penup()
34 forward(300)   # Move to top-right corner
35 pendown()
36
37 # draw square
38 forward(200)
39 right(90)
40 forward(200)
41 right(90)
42 forward(200)
43 right(90)
44 forward(200)
45 right(90)

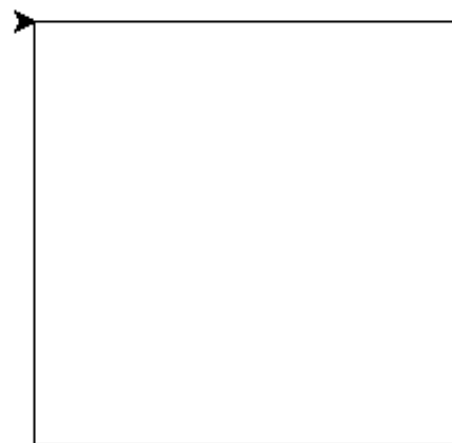
```

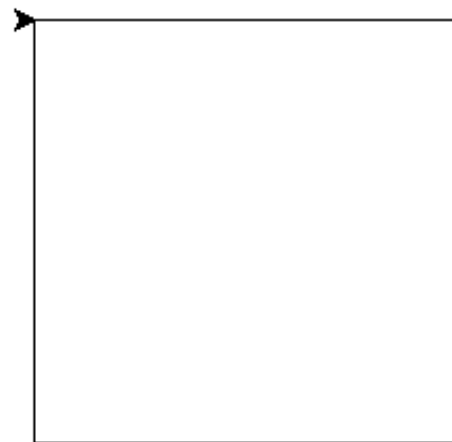
```

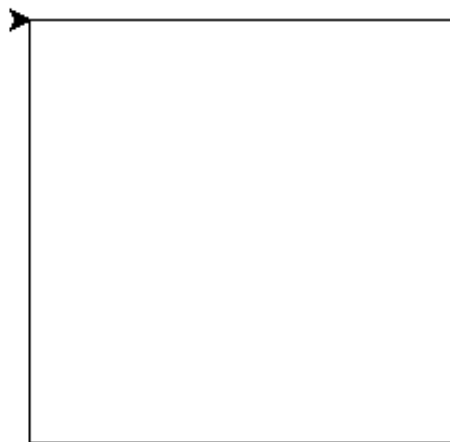
46
47 sleep(1)
48 clear()
49
50 penup()
51 right(90)      # face South
52 forward(300)   # Move to bottom-right corner
53 left(90)       # face East again
54 pendown()
55
56 # draw square
57 forward(200)
58 right(90)
59 forward(200)
60 right(90)
61 forward(200)
62 right(90)
63 forward(200)
64 right(90)
65
66 sleep(1)
67 clear()
68
69 penup()
70 backward(300)  # Move to bottom-left corner
71 pendown()
72
73 # draw square
74 forward(200)
75 right(90)
76 forward(200)
77 right(90)
78 forward(200)
79 right(90)
80 forward(200)
81 right(90)
82
83 sleep(1)
84 clear()
85
86 update()
87 done()
88


```









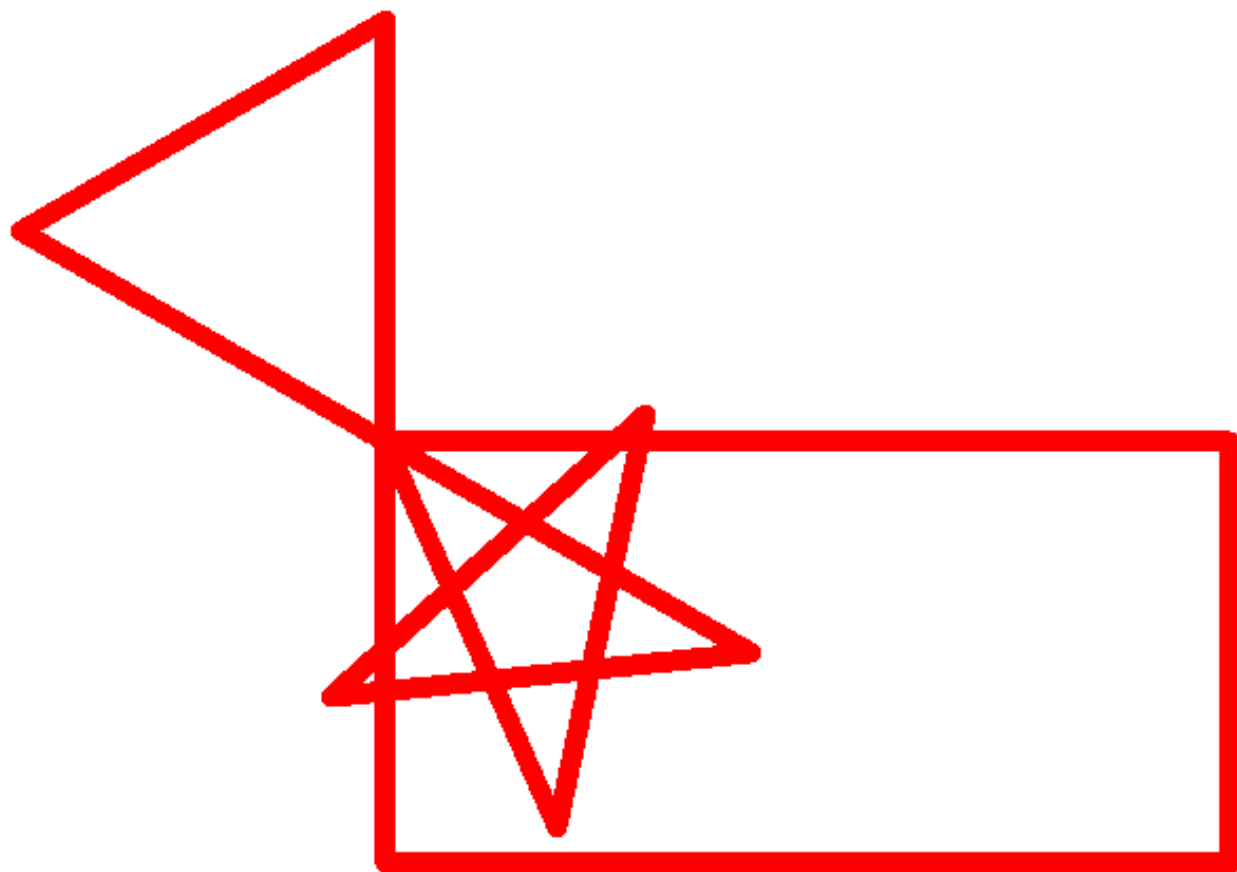
 Python Turtle Graphics



```
1 # TurtleGraphics20.py
2 # This program draws a rectangle, a triangle,
3 # and a star. Note how the way one shape
4 # finishes affects the location and rotation
5 # of the next shape.
6
7 from turtle import *
8
9 setup(800,600)
10 width(10)
11 color("red")
12
13 # rectangle
14 backward(200)
15 forward(400)
16 right(90)
17 forward(200)
18 right(90)
19 forward(400)
20 right(90)
21 forward(200)
```

```
22
23 #triangle
24 forward(200)
25 left(120)
26 forward(200)
27 left(120)
28 forward(200)
29
```

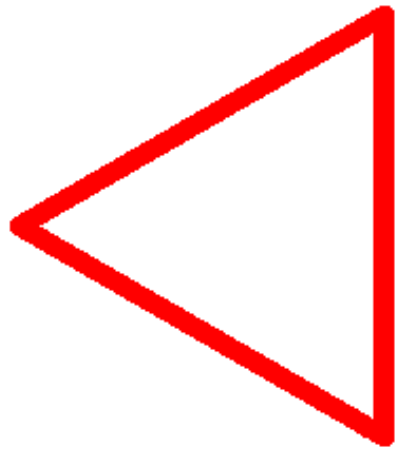
```
30 # star
31 forward(200)
32 right(144)
33 forward(200)
34 right(144)
35 forward(200)
36 right(144)
37 forward(200)
38 right(144)
39 forward(200)
40 right(144)
41
42 update()
43 done()
```

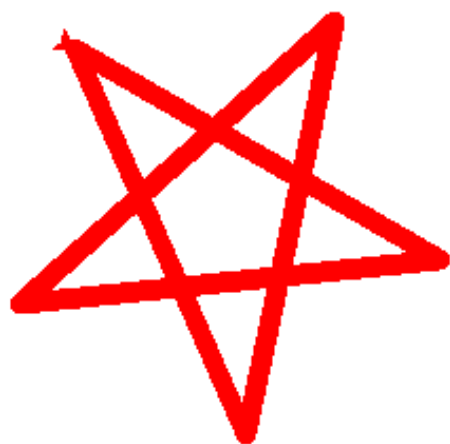



```
1 # TurtleGraphics21.py
2 # This program draws the same shapes from the
3 # previous program, but uses a <clear> command
4 # after each shape is drawn. Note how clearing
5 # the window this way has no effect on the
6 # shape's location, orientation, color or width.
7
8 from turtle import *
9 from time import sleep
10
11 setup(800,600)
12 width(10)
13 color("red")
14
15 # rectangle
16 backward(200)
17 forward(400)
18 right(90)
19 forward(200)
20 right(90)
21 forward(400)
22 right(90)
23 forward(200)
24
25 sleep(1)
26 clear()
```

```
27
28 #triangle
29 forward(200)
30 left(120)
31 forward(200)
32 left(120)
33 forward(200)
34
35 sleep(1)
36 clear()
37
38 # star
39 forward(200)
40 right(144)
41 forward(200)
42 right(144)
43 forward(200)
44 right(144)
45 forward(200)
46 right(144)
47 forward(200)
48 right(144)
49
50 sleep(1)
51 clear()
52
53 update()
54 done()
```



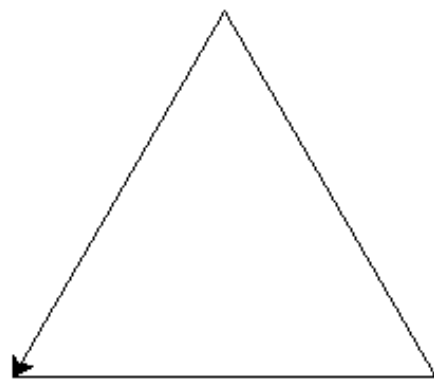


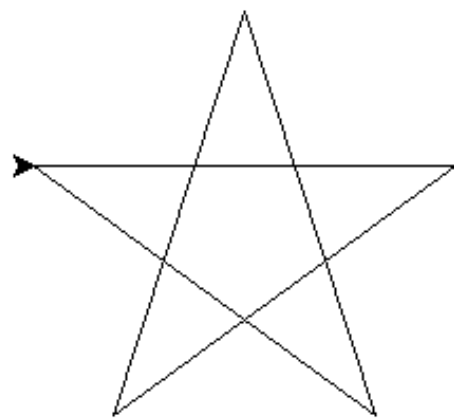


```
1 # TurtleGraphics22.py
2 # This program replaces the <clear> commands
3 # of the previous program with <reset>.
4 # The <reset> command does more than clear the
5 # screen. It also returns the turtle back to
6 # the center of the screen and makes it face
7 # right, sets the <width> value back to 1,
8 # and sets the <color> value back to "black".
9
10 from turtle import *
11 from time import sleep
12
13 setup(800, 600)
14 width(10)
15 color("red")
16
17 # rectangle
18 backward(200)
19 forward(400)
20 right(90)
21 forward(200)
22 right(90)
23 forward(400)
24 right(90)
25 forward(200)
26
27 sleep(1)
28 reset()
```

```
29
30 #triangle
31 forward(200)
32 left(120)
33 forward(200)
34 left(120)
35 forward(200)
36
37 sleep(1)
38 reset()
39
40 # star
41 forward(200)
42 right(144)
43 forward(200)
44 right(144)
45 forward(200)
46 right(144)
47 forward(200)
48 right(144)
49 forward(200)
50 right(144)
51
52 sleep(1)
53 reset()
54
55 update()
56 done()
```







clear vs. reset

clear	reset
<p>Clears the screen only</p>	<p>Clears the screen <i>and</i></p> <p>Returns the turtle to the center of the screen <i>and</i></p> <p>Makes the turtle face right <i>and</i></p> <p>Returns the width back to its default value of 1 <i>and</i></p> <p>Returns the color back to its default value of "black"</p>

More Turtle Graphics Commands?

There are actually many more commands available in the **turtle** graphics library. We are not going to cover those commands in this class. The reason is we want to focus more on using “Traditional Graphics” which is coordinate based. This will be introduced in the next chapter and used throughout the entire school year. This chapter on “Turtle Graphics” was simply meant to be an introduction.