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

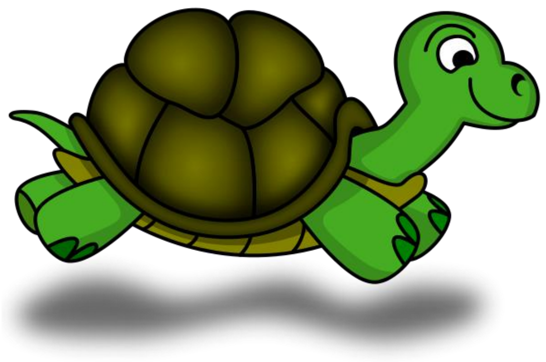
Turtle Graphics

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



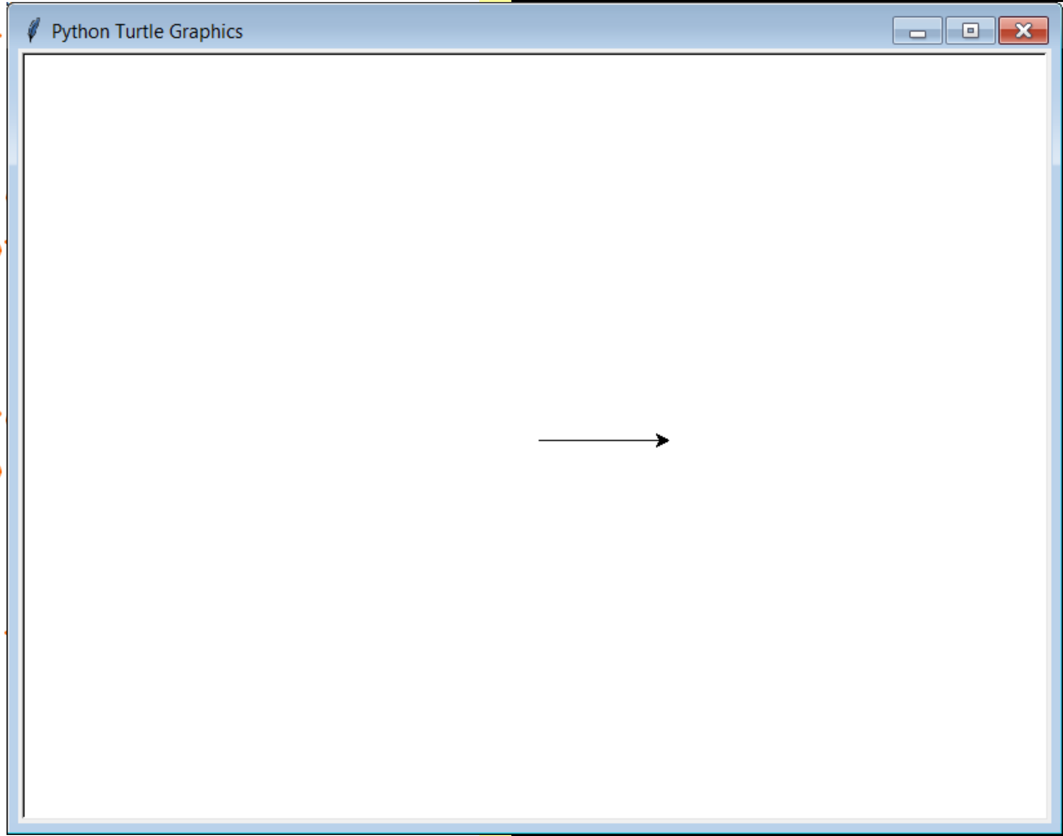
python

Section 5.2

Importing Libraries and Turtle Graphics Setup

```python
# TurtleGraphics01.py
# This program introduces "Turtle Graphics" by
# importing the <turtle> library and drawing a
# single line with the <forward> command.


# Required to have access to
# the turtle graphics commands.
import turtle

# Specifies the dimensions of
# the Turtle Graphics window.
turtle.setup(800,600)

# Moves the "turtle" forward 100 pixels
# and draws a line in the process.
turtle.forward(100)

# Required to "update" the window
# when everything is drawn.
turtle.update()

# Required to keep the graphics window
# open when the program is finished.
turtle.done()
```

```
# TurtleGraphics01.py
# This program introduces "Turtle Graphics" by
# importing the <turtle> library and drawing a
# single line with the <forward> command.


# Required to have access to
# the turtle graphics comma
import turtle

# Specifies the dimensions
# the Turtle Graphics windo
turtle.setup(800,600)

# Moves the "turtle" forwar
# and draws a line in the p
turtle.forward(100)

# Required to "update" the
# when everything is drawn.
turtle.update()

# Required to keep the graphics window
# open when the program is finished.
turtle.done()
```



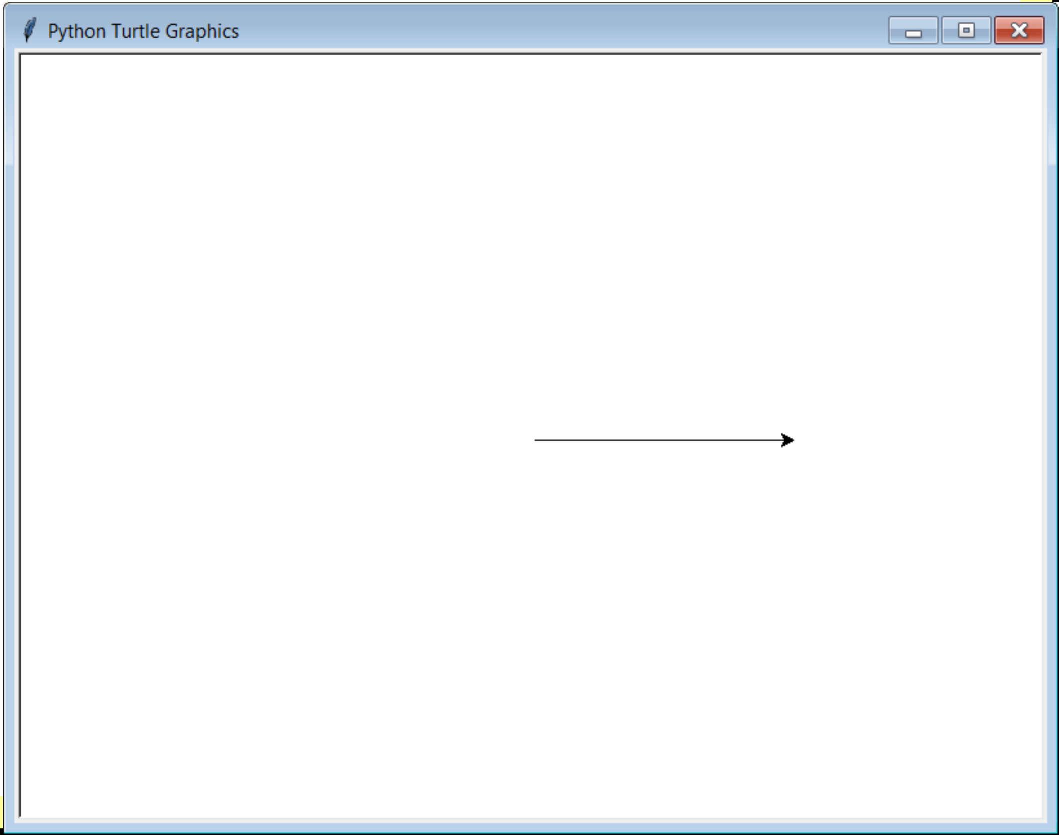
```python
# TurtleGraphics02.py
# This program has the exact same output as the
# previous one.  By importing the individual
# <turtle> library command in this way, the
# rest of the code can be less "wordy".


from turtle import setup
from turtle import forward
from turtle import update
from turtle import done

setup(800,600)
forward(100)
update()
done()

```

```python
# TurtleGraphics03.py
# This program has the exact same output as the
# previous two.  By using the <*> "wildcard" we
# can <import> all of the <turtle> library commands
# at once making the code even less "wordy".


from turtle import *

setup(800,600)
forward(100)
update()
done()

```

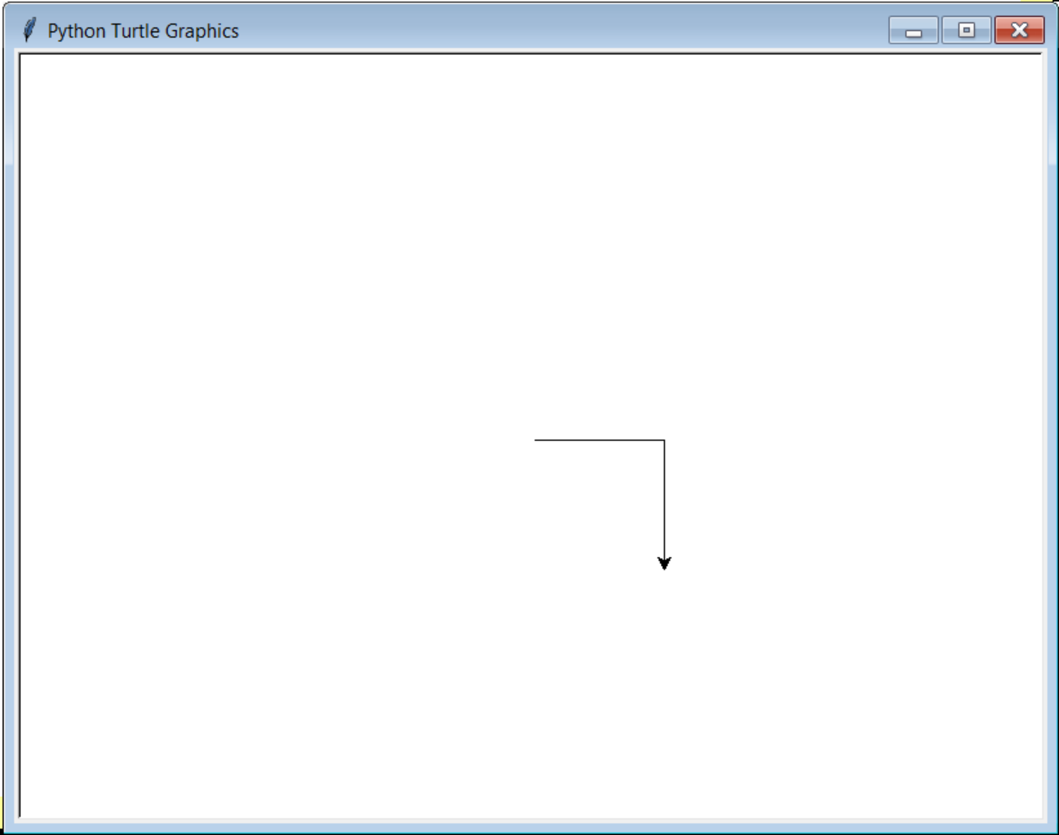
Section 5.3

Drawing by Moving & Turning the Turtle

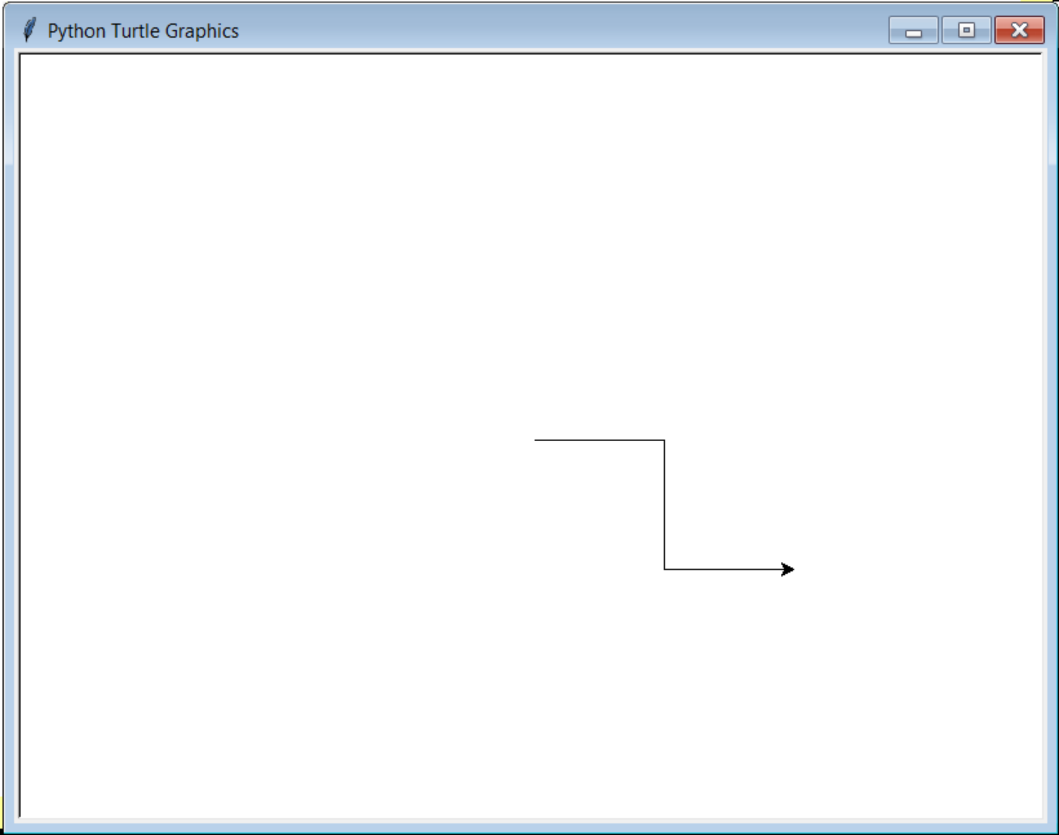
```python
# TurtleGraphics04.py
# This program shows what happens when
# <forward> is called twice.


from turtle import *

setup(800,600)

forward(100)
forward(100)

update()
done()

```



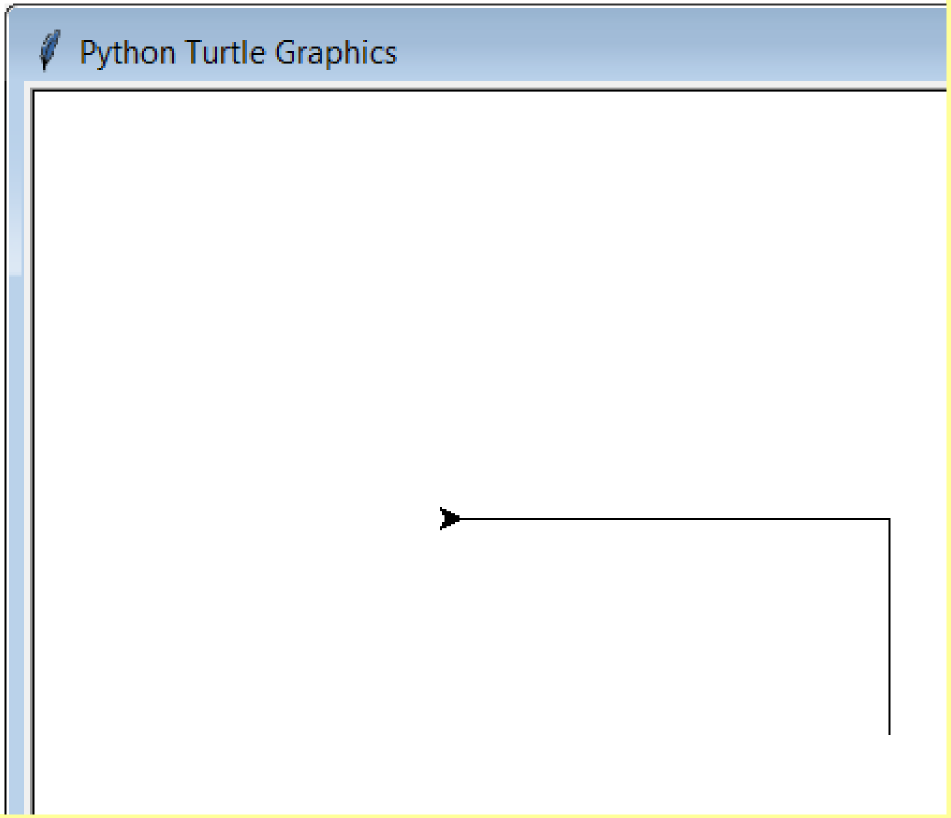
```
# TurtleGraphics05.py
# This program has the "turtle" make a
# 90 degree turn to the "right" before
# the second line is drawn.


from turtle import *

setup(800,600)

forward(100)
right(90)
forward(100)

update()
done()
```



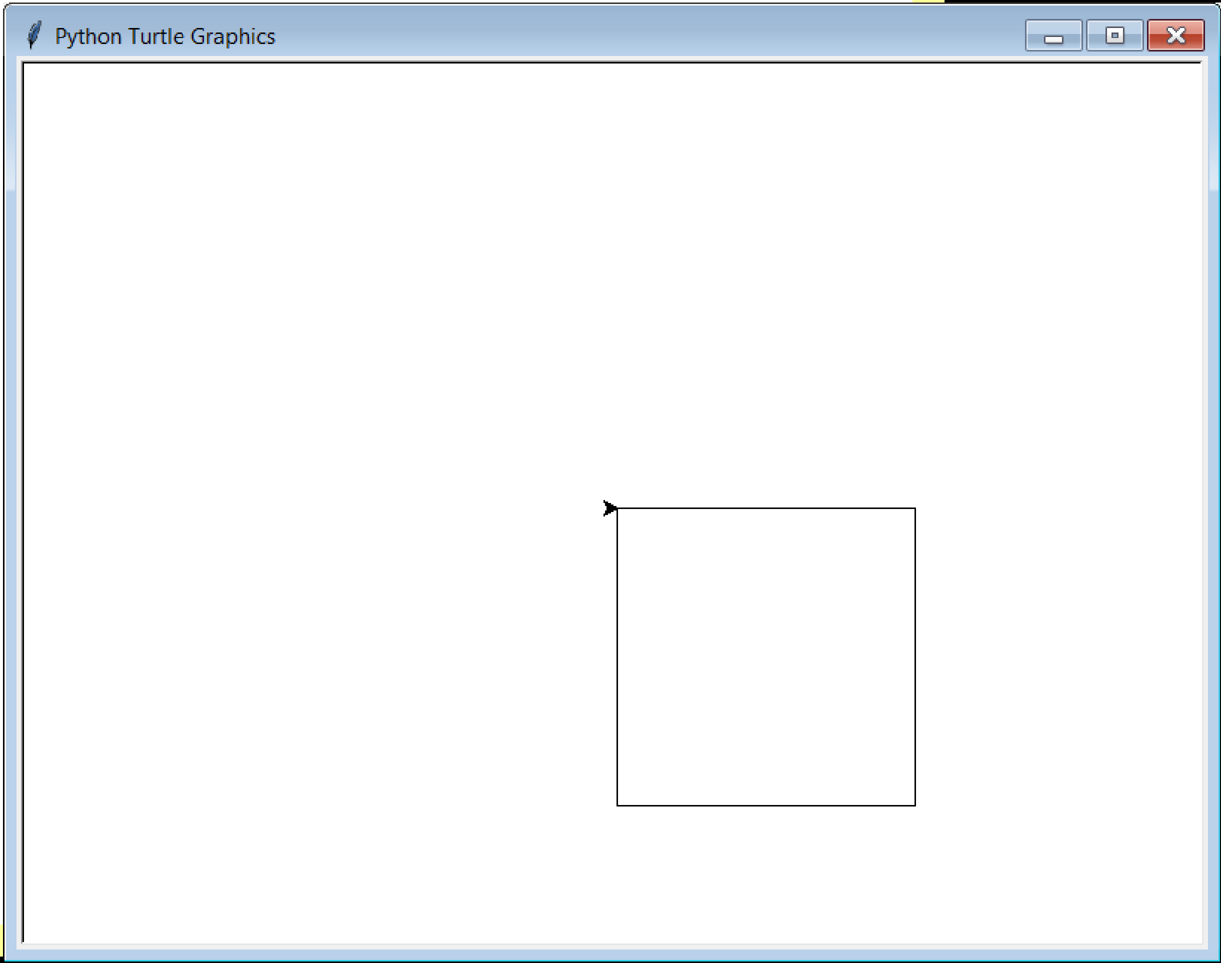
```
# TurtleGraphics06.py
# This program shows that "left" turns
# are also possible.


from turtle import *

setup(800,600)

forward(100)
right(90)
forward(100)
left(90)
forward(100)

update()
done()
```



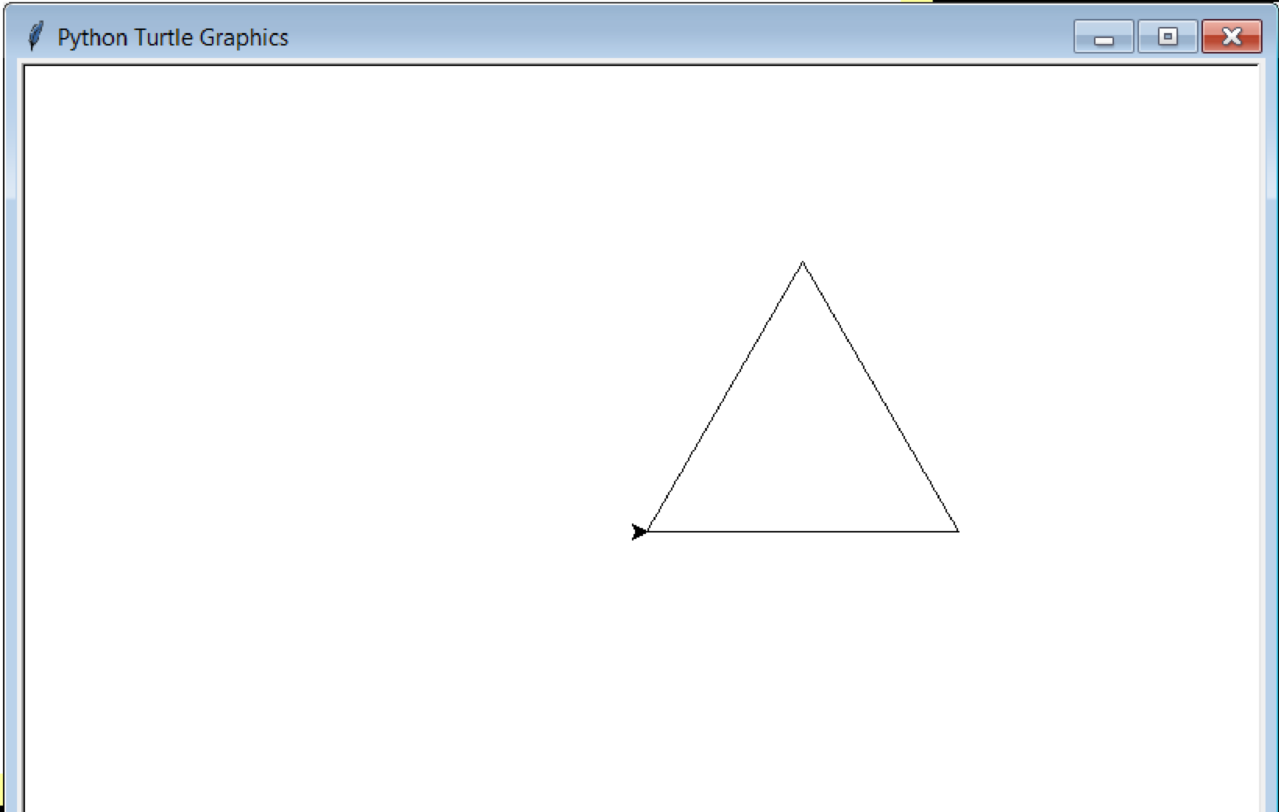
```
# TurtleGraphics07.py
# This program shows that the "turtle"
# can also move "backward".


from turtle import *

setup(800,600)

left(90)
forward(100)
right(90)
backward(200)

update()
done()
```



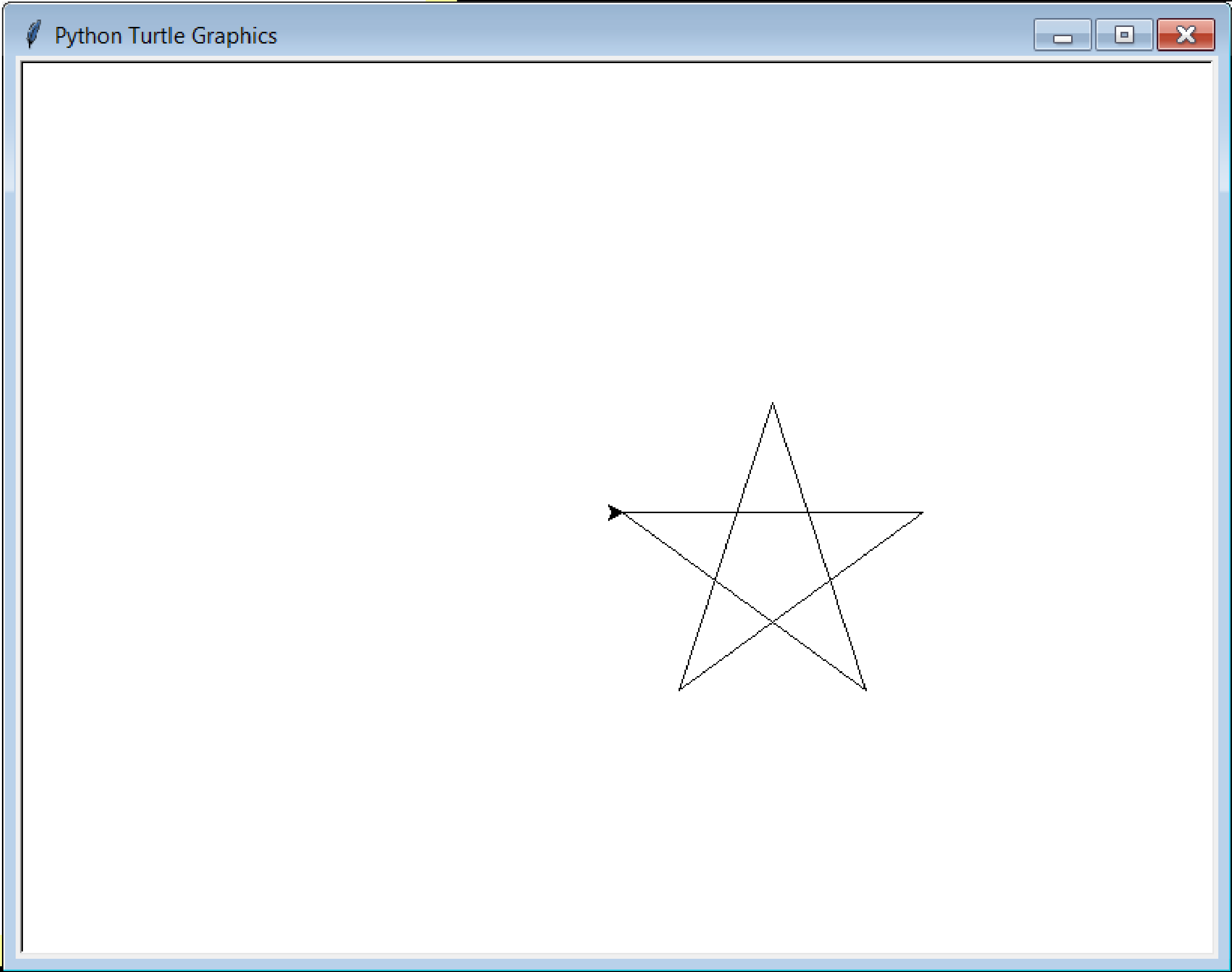
```python
# TurtleGraphics08.py
# This program makes a square by going
# "forward" and turning "right" 4 times.


from turtle import *

setup(800,600)

forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

update()
done()
```



```python
# TurtleGraphics09.py
# This program demonstrates that turns do
# not need to be 90 degree "right angles".
# You can turn any number of degrees.
# With 3 turns of 120 degrees, I can
# make an equilateral triangle.


from turtle import *

setup(800,600)

forward(200)
left(120)
forward(200)
left(120)
forward(200)
left(120)

update()
done()
```

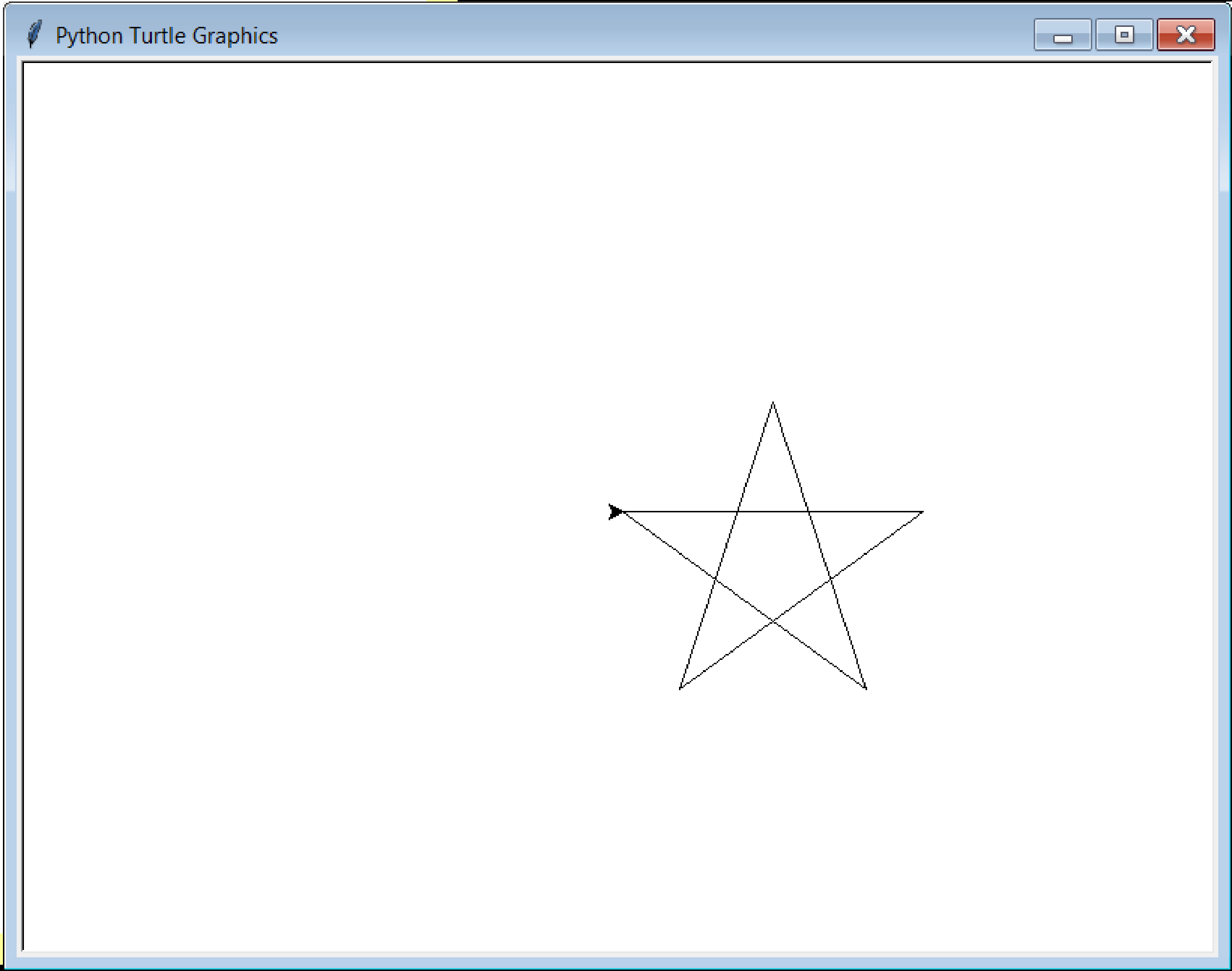


```python
# TurtleGraphics10.py
# Even stars are possible.


from turtle import *

setup(800,600)

forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)

update()
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*.

```
setup(800,600)

forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)

update()
done()
```

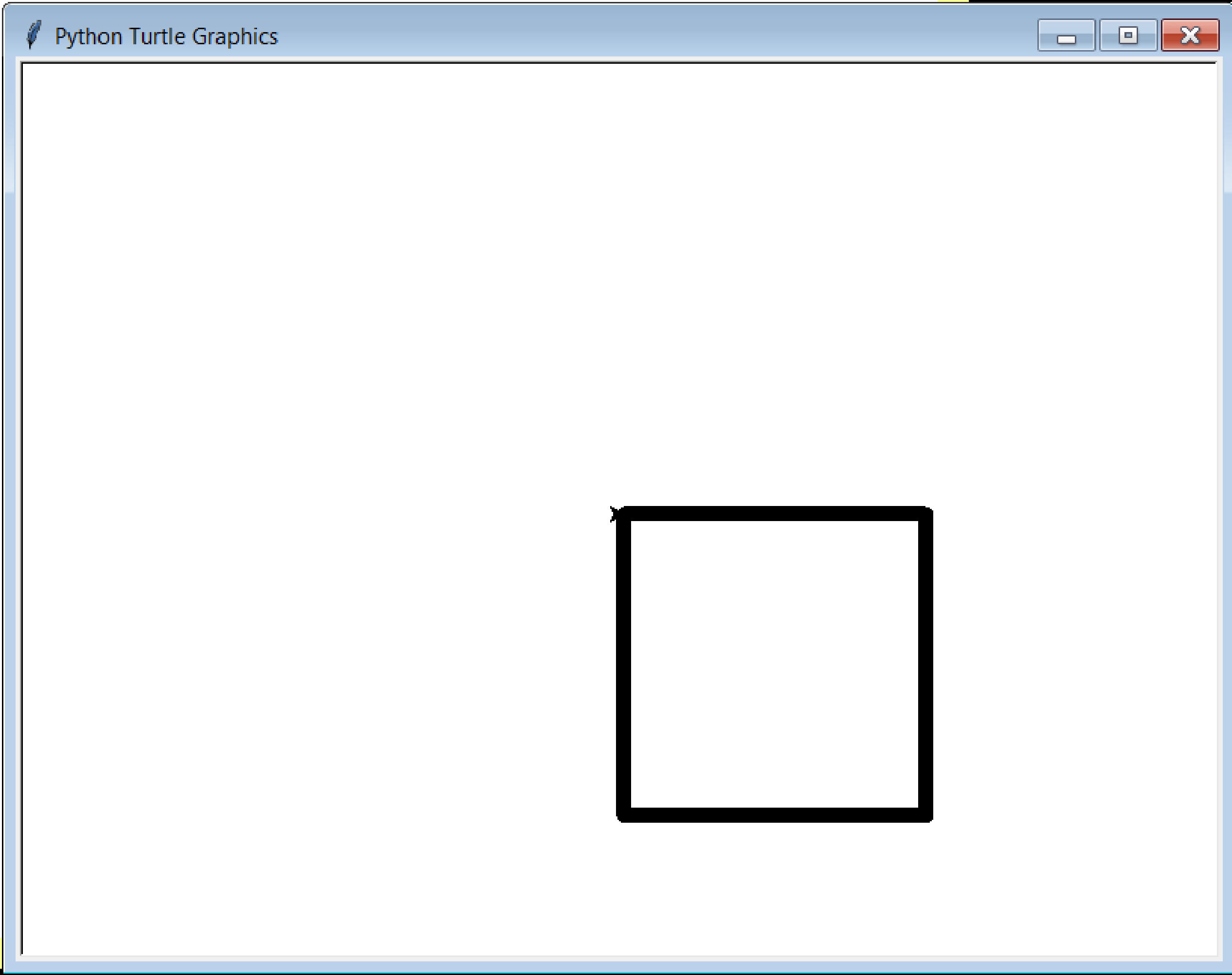


Section 5.4

Drawing Thick Or Solid Images

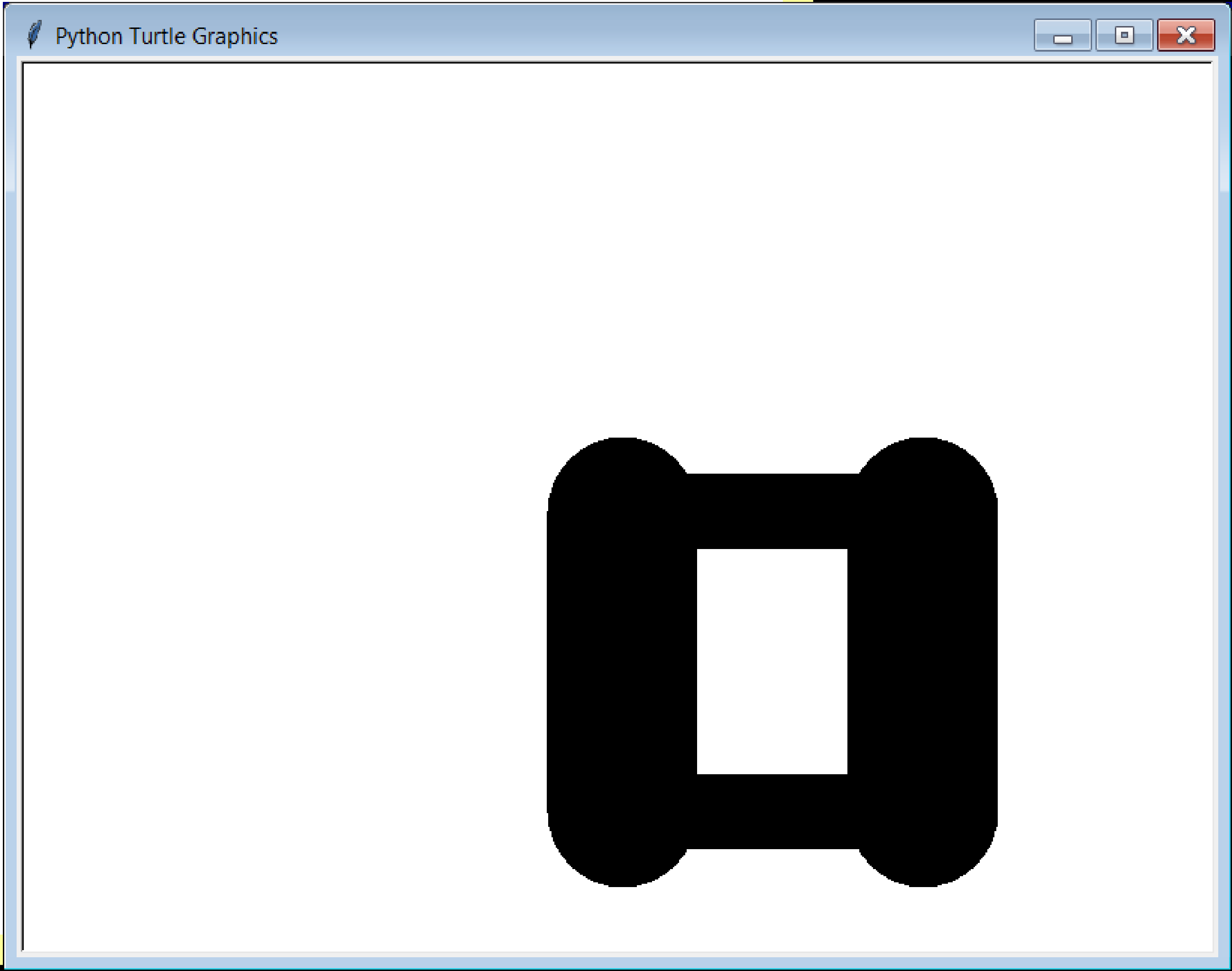
```python
# TurtleGraphics11.py
# This program demonstrates how you can change
# the thickness or <width> of the lines.


from turtle import *

setup(800,600)

width(10)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

update()
done()
```

```
# TurtleGraphics11.py
# This program demonstrates how you can change
# the thickness or <width> of the lines.


from turtle import *

setup(800,600)

width(10)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

update()
done()
```



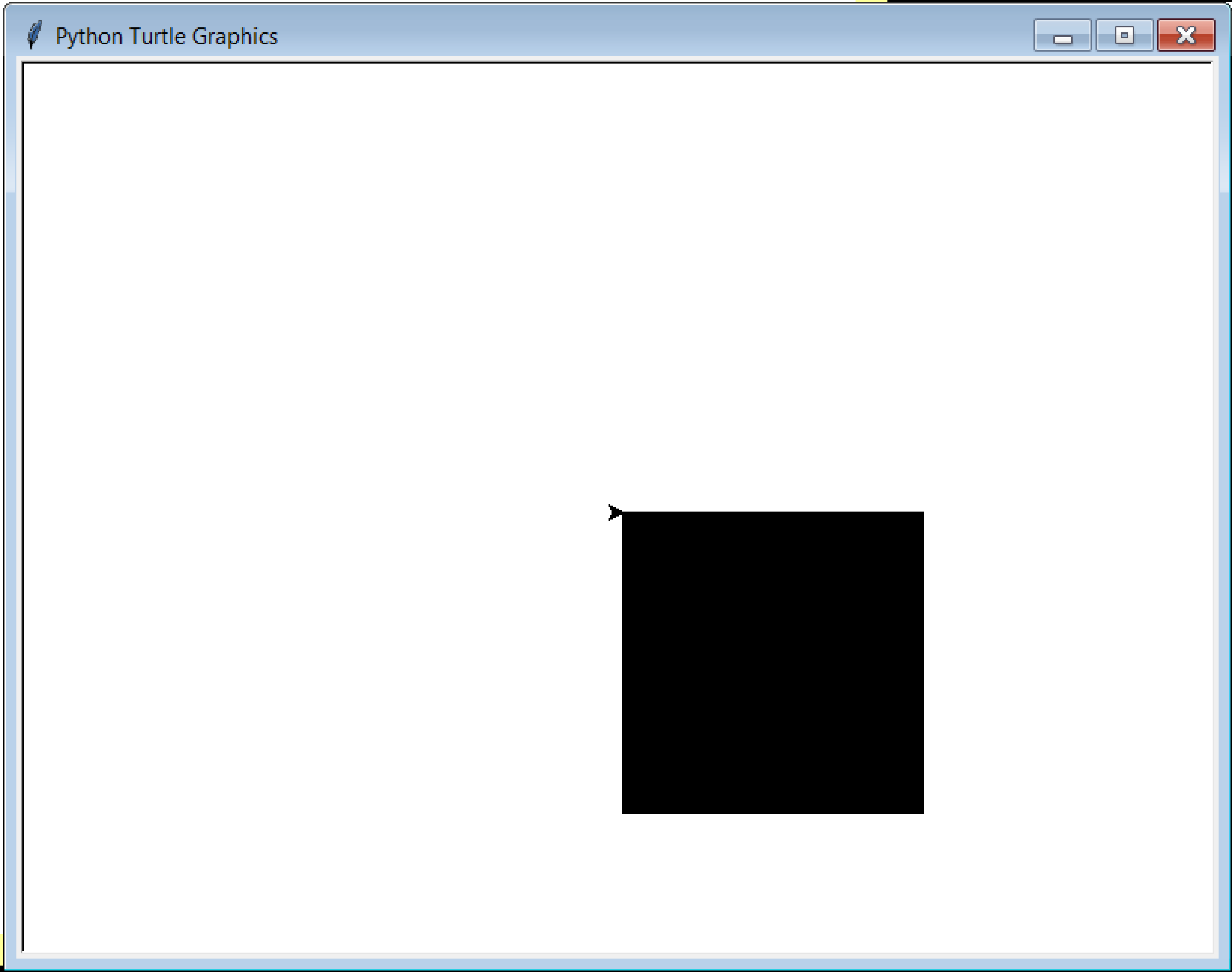
```
# TurtleGraphics12.py
# This program demonstrates that different
# lines can have different widths -- which
# can be quite thick.


from turtle import *

setup(800,600)

width(50)
forward(200)
right(90)
width(100)
forward(200)
right(90)
width(50)
forward(200)
right(90)
width(100)
forward(200)
right(90)

update()
done()
```

```python
# TurtleGraphics12.py
# This program demonstrates that different
# lines can have different widths -- which
# can be quite thick.


from turtle import *

setup(800,600)

width(50)
forward(200)
right(90)
width(100)
forward(200)
right(90)
width(50)
forward(200)
right(90)
width(100)
forward(200)
right(90)

update()
done()
```

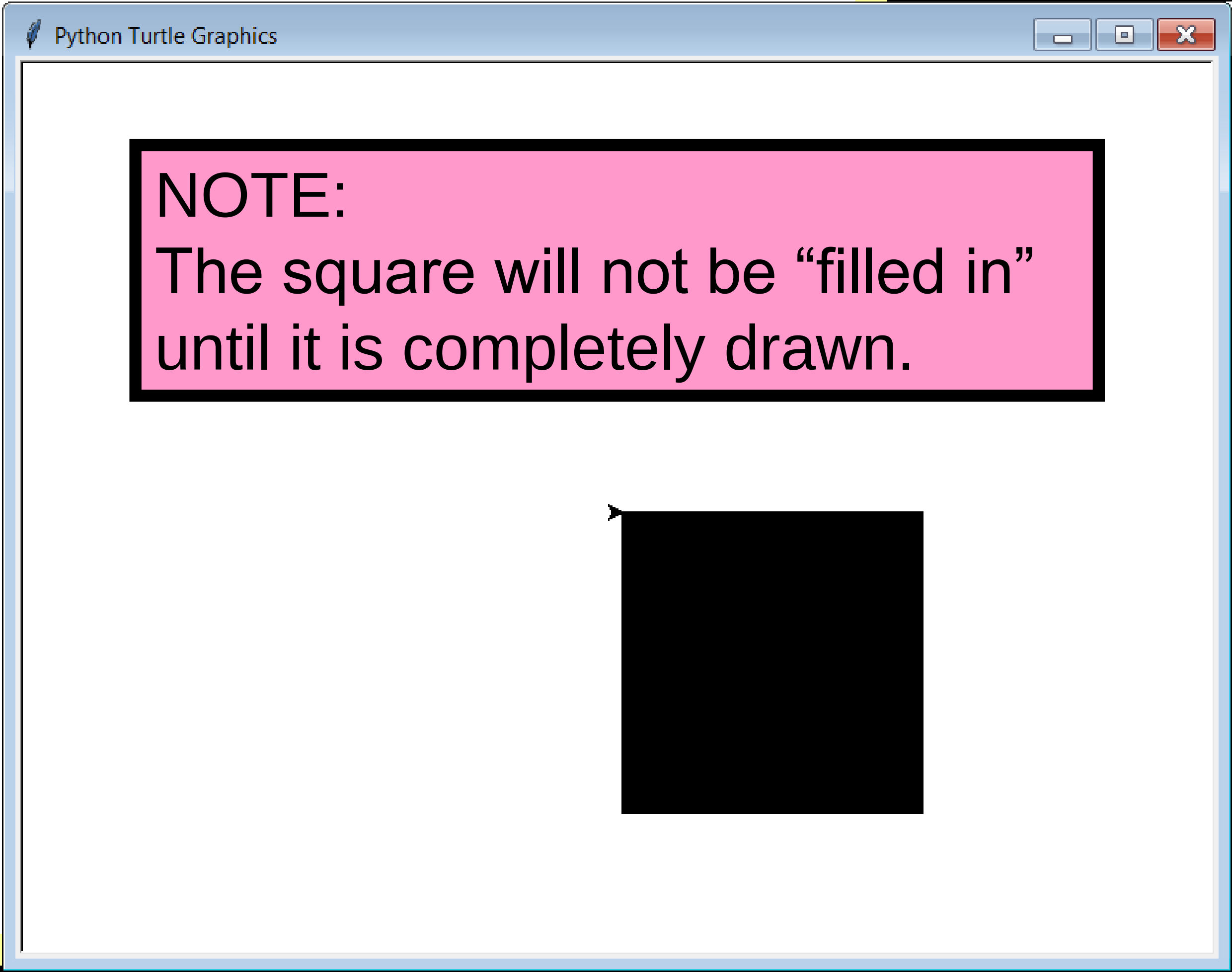


```python
# TurtleGraphics13.py
# This program demonstrates that you can
# "fill" enclosed shapes.

from turtle import *

setup(800,600)

begin_fill()
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
end_fill()

update()
done()
```

```python
# TurtleGraphics13.py
# This program demonstrates that you can
# "fill" enclosed shapes.

from turtle import *

setup(800,600)

begin_fill()
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
end_fill()

update()
done()
```



```python
# TurtleGraphics13.py
# This program demonstrates that you can
# "fill" enclosed shapes.

from turtle import *

setup(800,600)

begin_fill()
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
end_fill()

update()
done()
```



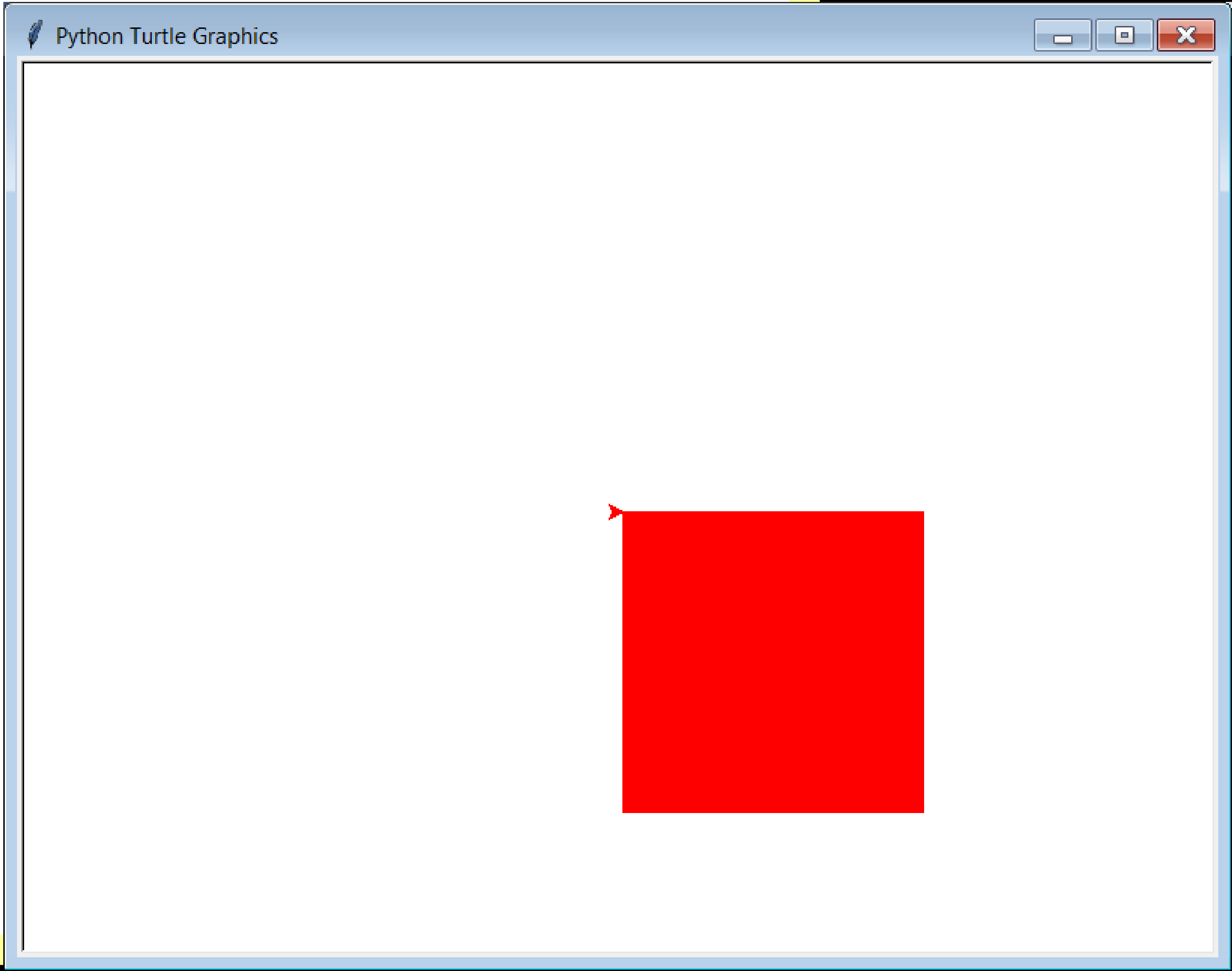
NOTE:
The square will not be “filled in” until it is completely drawn.

Section 5.5

Adding Color

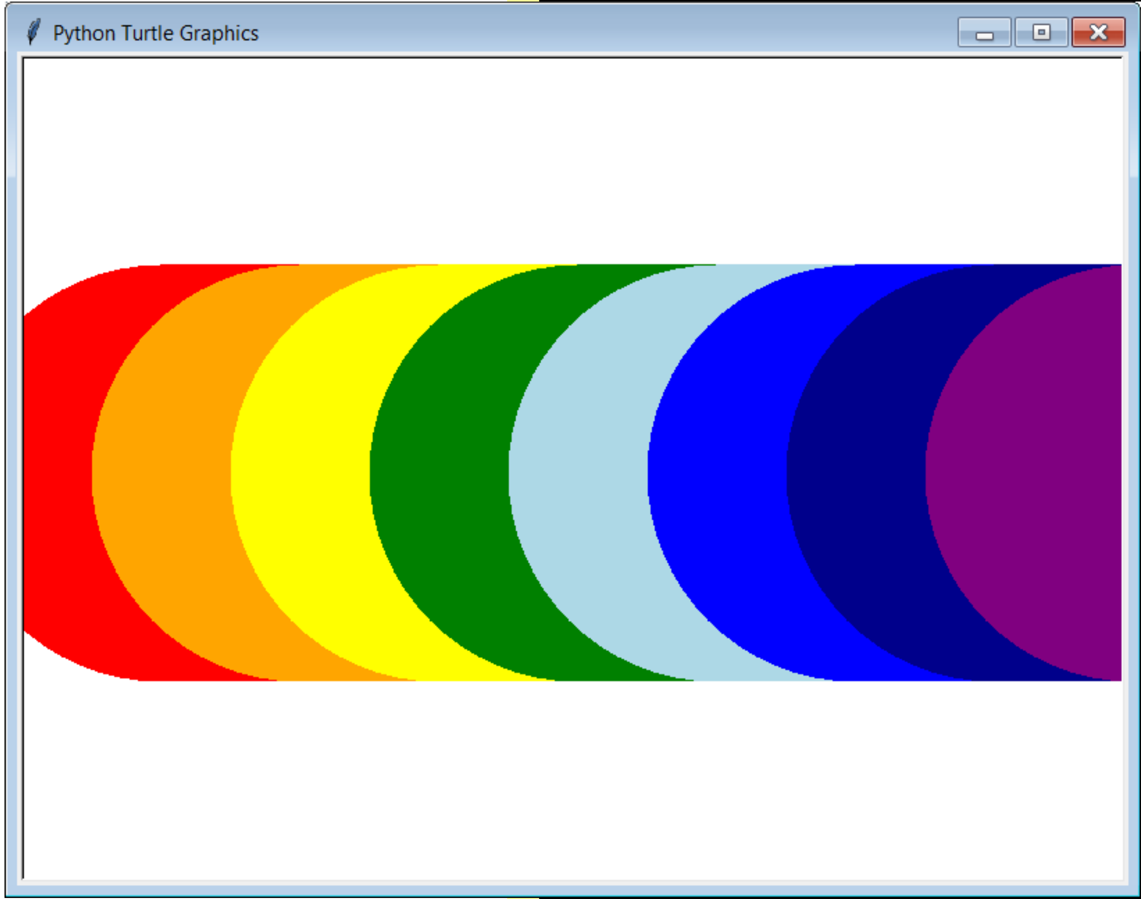
```python
# TurtleGraphics14.py
# This program demonstrates that you can
# draw in other colors besides "black".


from turtle import *

setup(800,600)

color("red")
begin_fill()
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
end_fill()

update()
done()
```

```python
# TurtleGraphics14.py
# This program demonstrates that you can
# draw in other colors besides "black".


from turtle import *

setup(800,600)

color("red")
begin_fill()
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
end_fill()

update()
done()
```

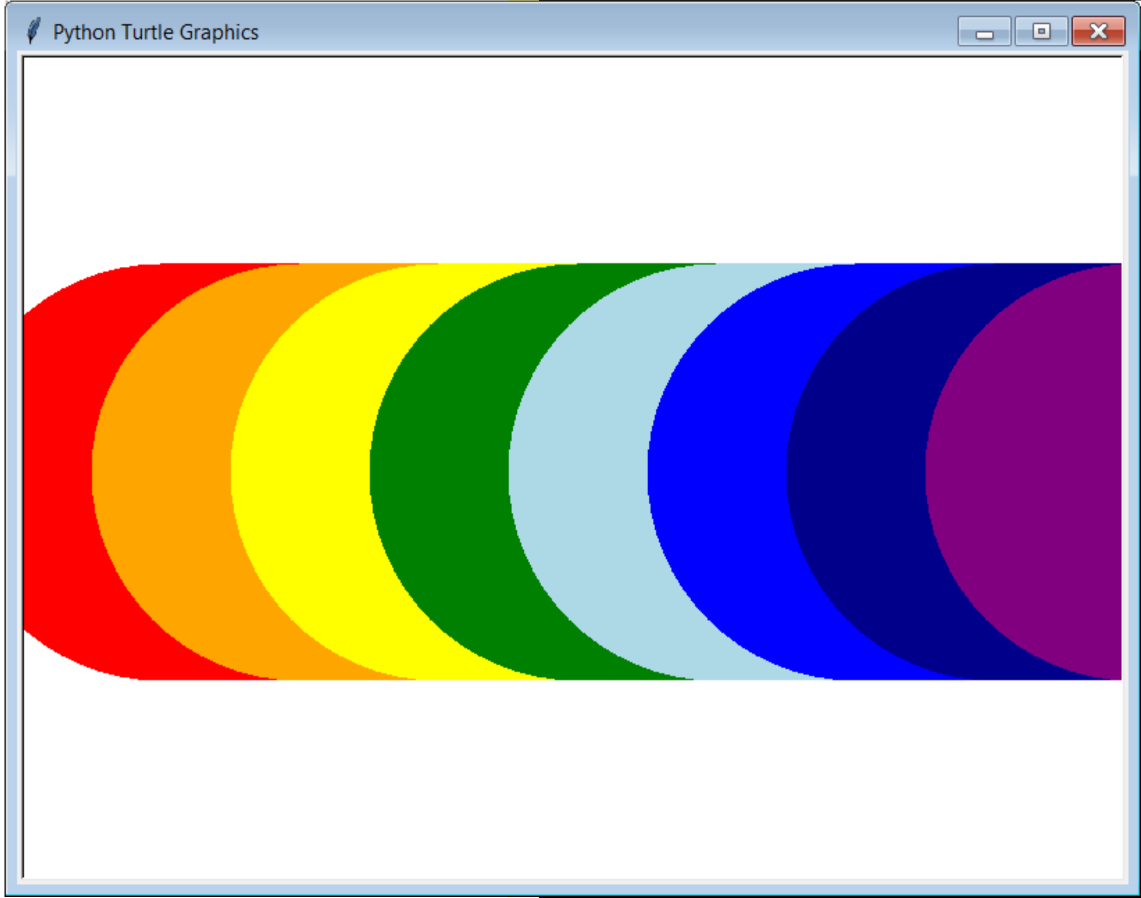


```python
# TurtleGraphics15.py
# This program demonstrates several colors.

from turtle import *

setup(800,600)

backward(300)
width(300)
color("red")
forward(100)
color("orange")
forward(100)
color("yellow")
forward(100)
color("green")
forward(100)
color("light blue")
forward(100)
color("blue")
forward(100)
color("dark blue")
forward(100)
color("purple")
forward(100)

update()
done()
```

```python
# TurtleGraphics15.py
# This program demonstrates several colors.

from turtle import *

setup(800,600)

backward(300)
width(300)
color("red")
forward(100)
color("orange")
forward(100)
color("yellow")
forward(100)
color("green")
forward(100)
color("light blue")
forward(100)
color("blue")
forward(100)
color("dark blue")
forward(100)
color("purple")
forward(100)

update()
done()
```



```python
# TurtleGraphics15.py
# This program demonstrates several colors.

from turtle import *

setup(800,600)

backward(300)
width(300)
color("red")
forward(100)
color("orange")
forward(100)
color("yellow")
forward(100)
color("green")
forward(100)
color("light blue")
forward(100)
color("blue")
forward(100)
color("dark blue")
forward(100)
color("purple")
forward(100)

update()
done()
```



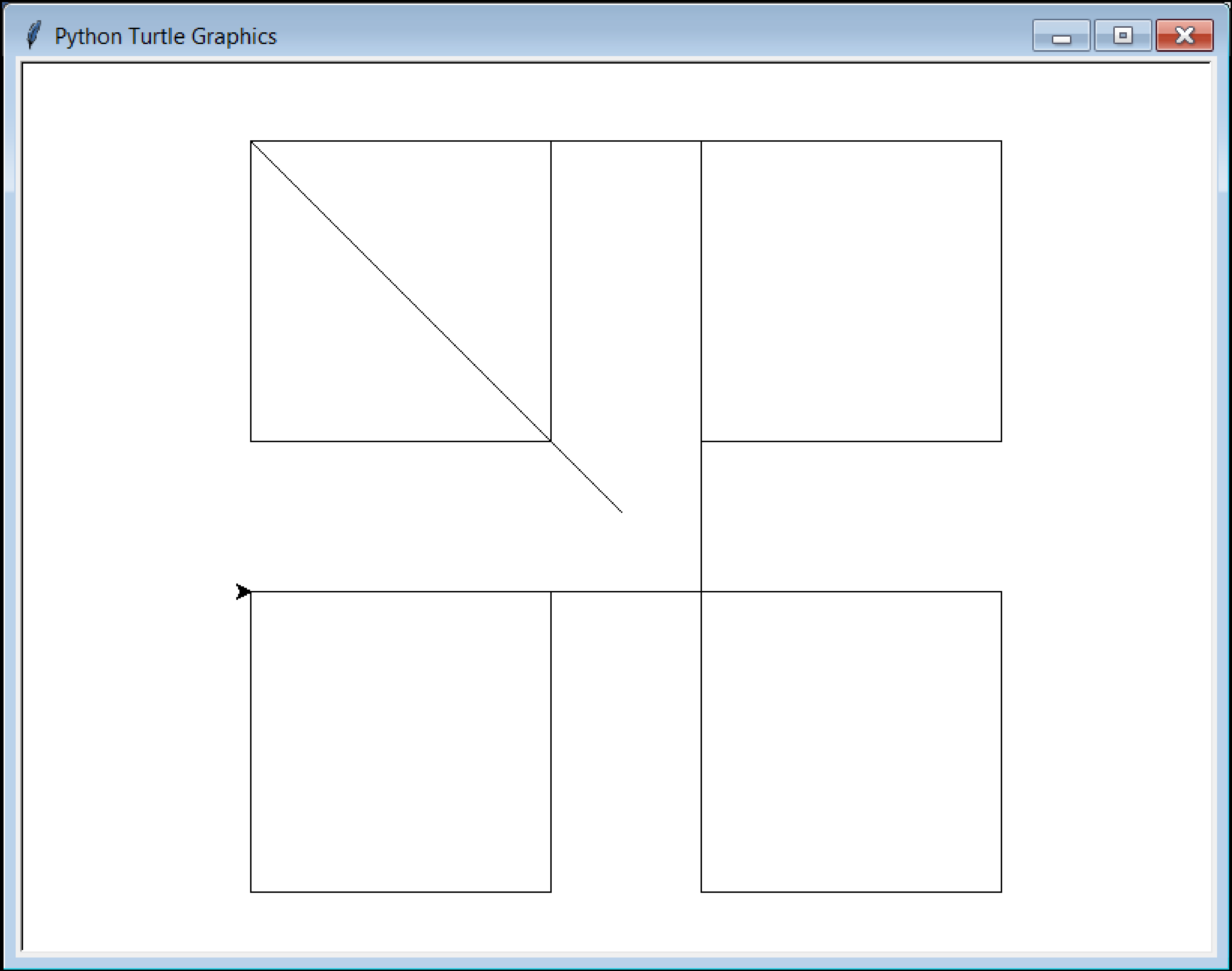
NOTE:
Python has 140 different colors.
These are the same colors used in HTML to create Webpages.

Section 5.6

Lifting the Pen

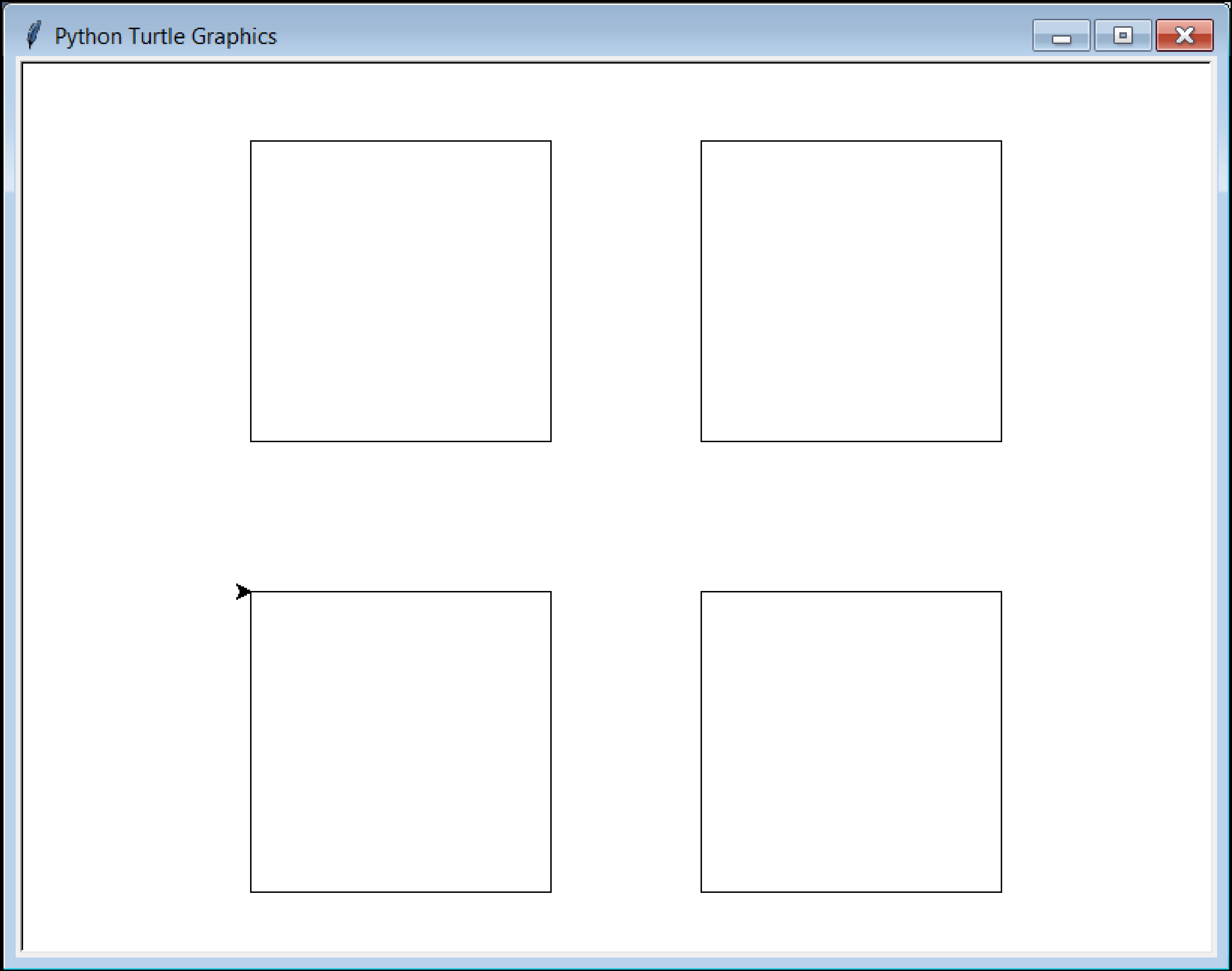
```python
# TurtleGraphics16.py
# This program tries to draw 4 separate squares.
# Before each square is drawn; the "turtle" moves
# to one of the corners of the graphics window.
# The problem is the turtle is always drawing
# a several lines are drawn that we do not want.


from turtle import *

setup(800,600)

left(135)      # face North-West
forward(350)   # Move to top-left corner
right(135)     # face East again

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

forward(300)   # Move to top-right corner

# draw square
forward(200)
right(90)
forward(200)
right(90)
```

```
# TurtleGraphics16.py
# This program tries to dr
# Before each square is dr
# to one of the corners of
# The problem is the turtl
# a several lines are draw


from turtle import *

setup(800,600)

left(135)        # face North
forward(350)     # Move to to
right(135)       # face East

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

forward(300)     # Move to to

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

right(90)        # face South
forward(300)     # Move to bottom-right corner
left(90)         # face East again

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

backward(300)  # Move to bottom-left corner

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

update()
done()
```


Python Turtle Graphics

```python
# TurtleGraphics17.py
# This program improves on the previous program
# by adding strategic <penup> and <pendown>
# commands at the appropriate places.
# Now, only the 4 squares are drawn.


from turtle import *

setup(800,600)

penup()
left(135)      # face North-West
forward(350)   # Move to top-left corner
right(135)     # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

penup()
forward(300)   # Move to top-right corner
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
```

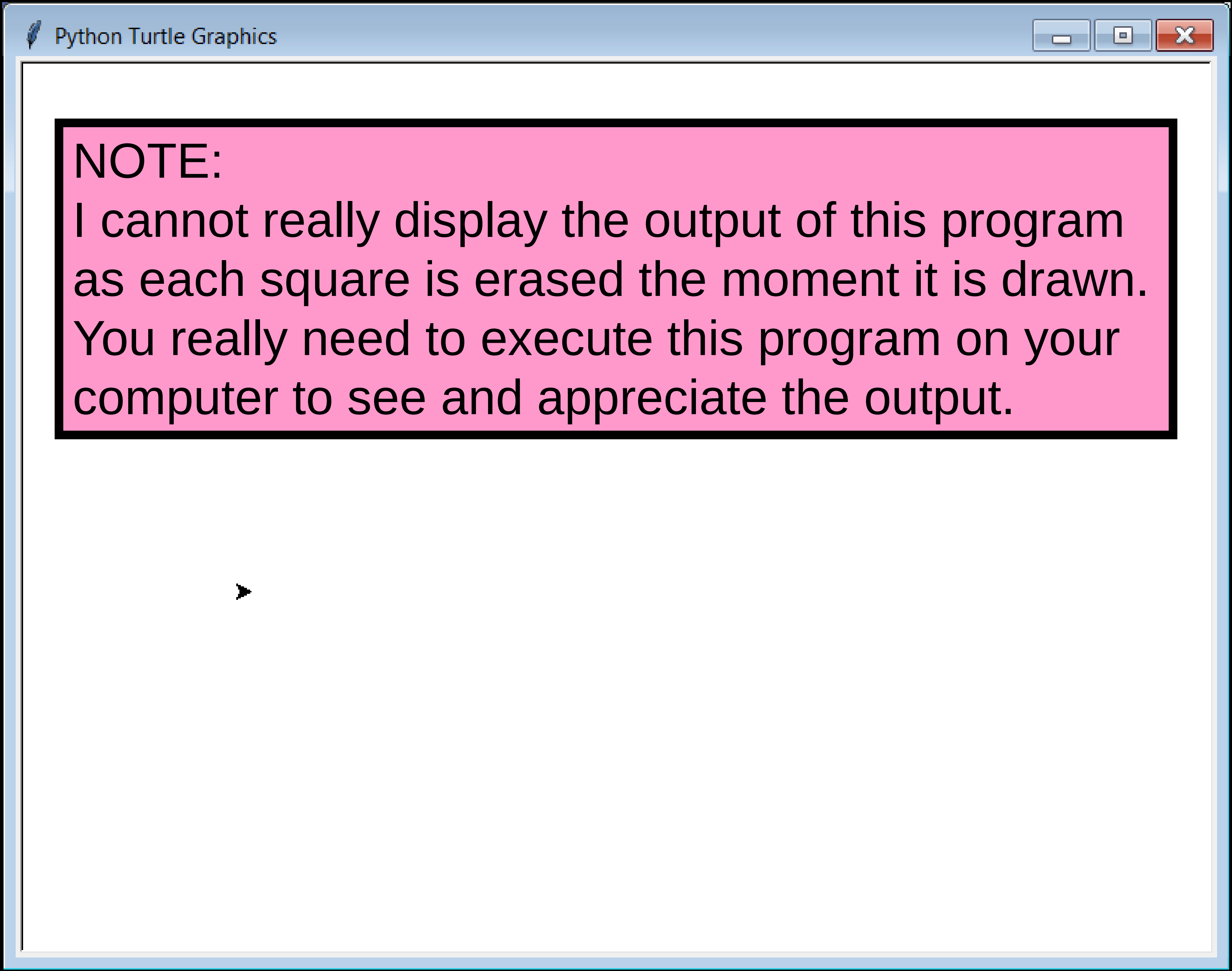
```python
# TurtleGraphics17.py
# This program improves on the previ
# by adding strategic <penup> and <p
# commands at the appropriate places
# Now, only the 4 squares are drawn.


from turtle import *

setup(800,600)

penup()
left(135)     # face North-West
forward(350)  # Move to top-left cor
right(135)    # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

penup()
forward(300)  # Move to top-right co
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

penup()
right(90)     # face South
forward(300)  # Move to bottom-right corner
left(90)      # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

penup()
backward(300) # Move to bottom-left corner
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

update()
done()

```


Python Turtle Graphics

Section 5.7

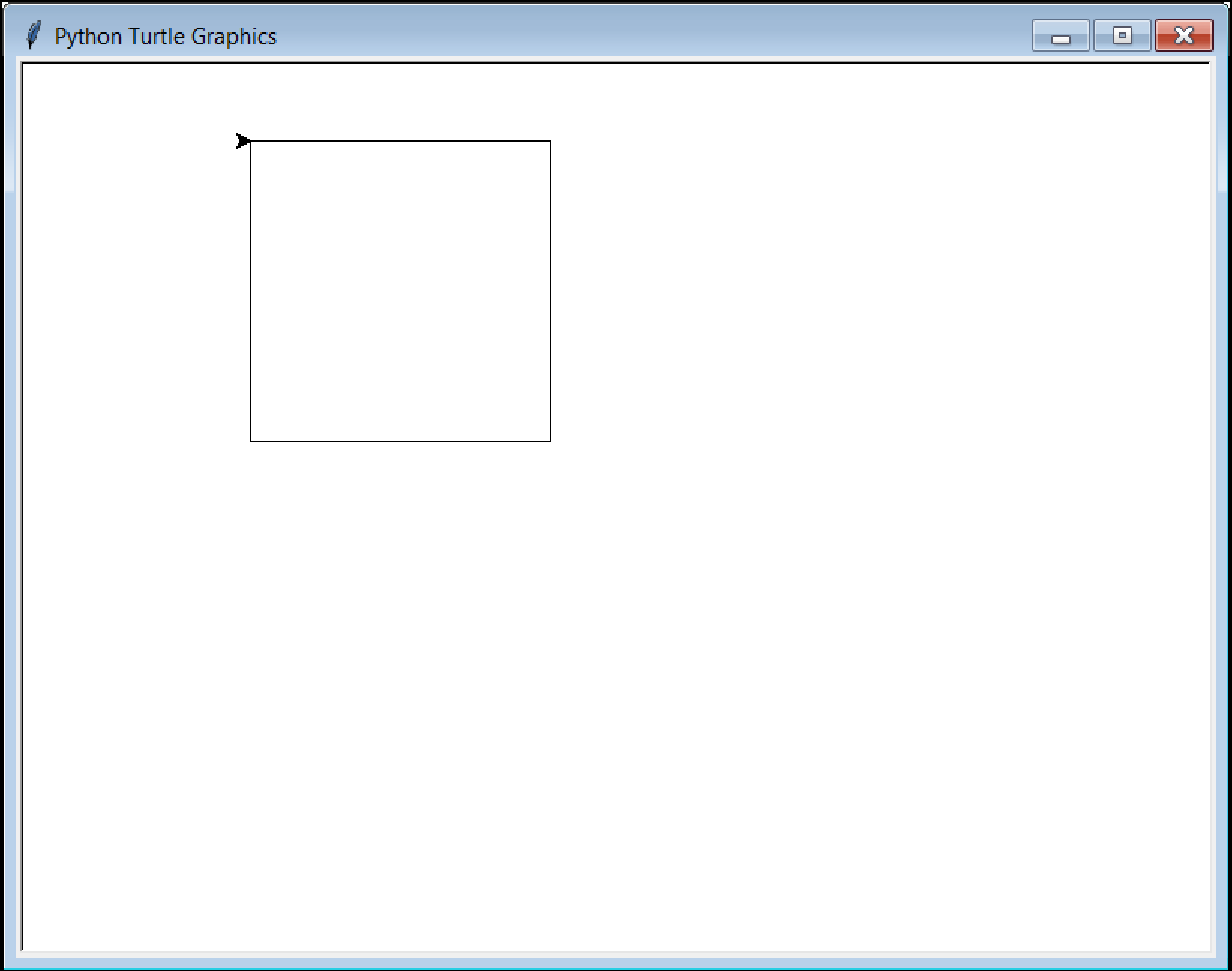
Clearing the Window

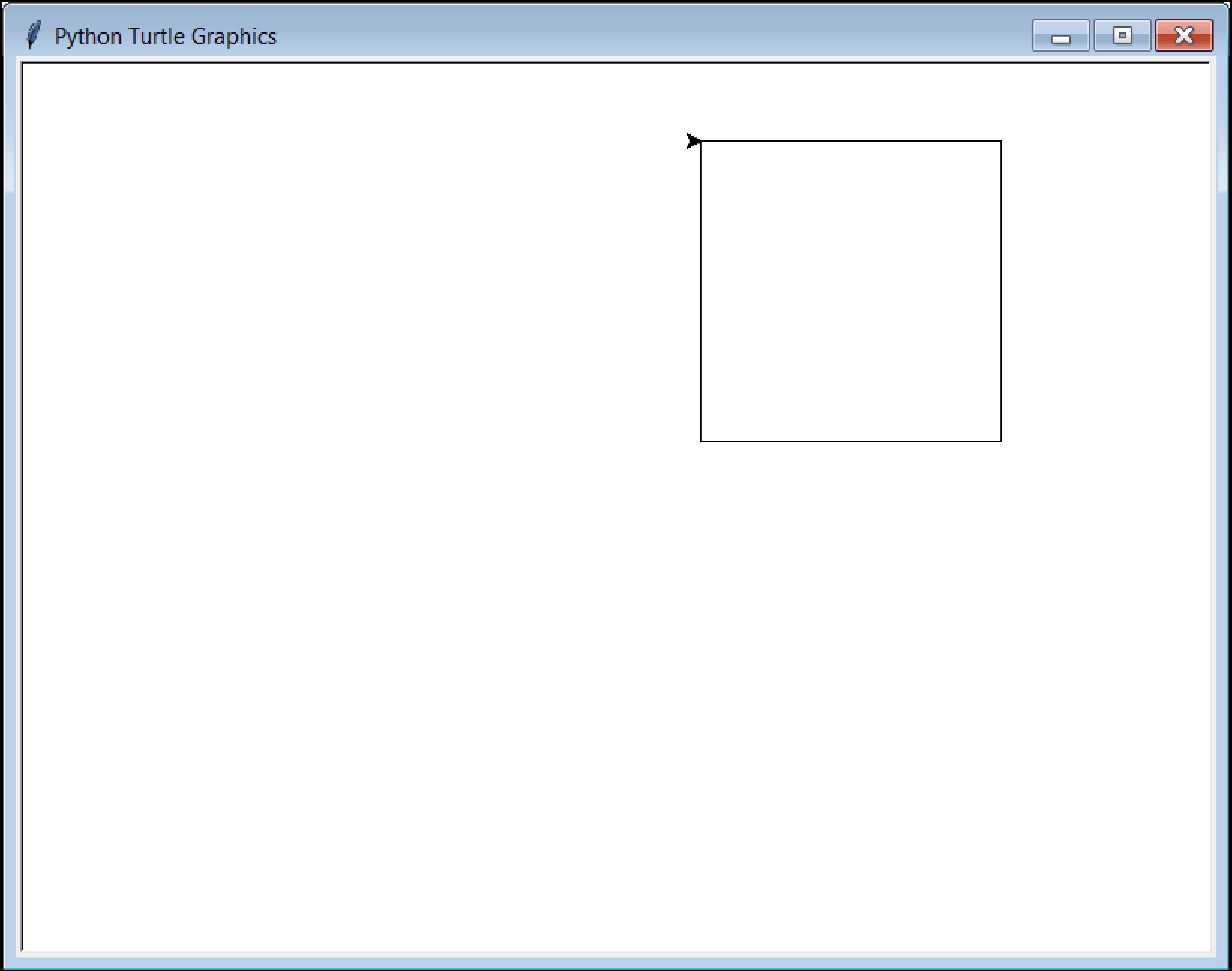
```python
# TurtleGraphics18.py
# This program puts a <clear> command after each
# square is drawn.  Now we only see one square
# (briefly) at a time.

from turtle import *

setup(800,600)

penup()
left(135)       # face North-West
forward(350)    # Move to top-left corner
right(135)      # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

clear()

penup()
forward(300)    # Move to top-right corner
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

clear()

penup()
right(90)        # face South
forward(300)     # Move to bottom-right corner
left(90)         # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

clear()

penup()
backward(300)  # Move to bottom-left corner
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

clear()

update()
done()
```

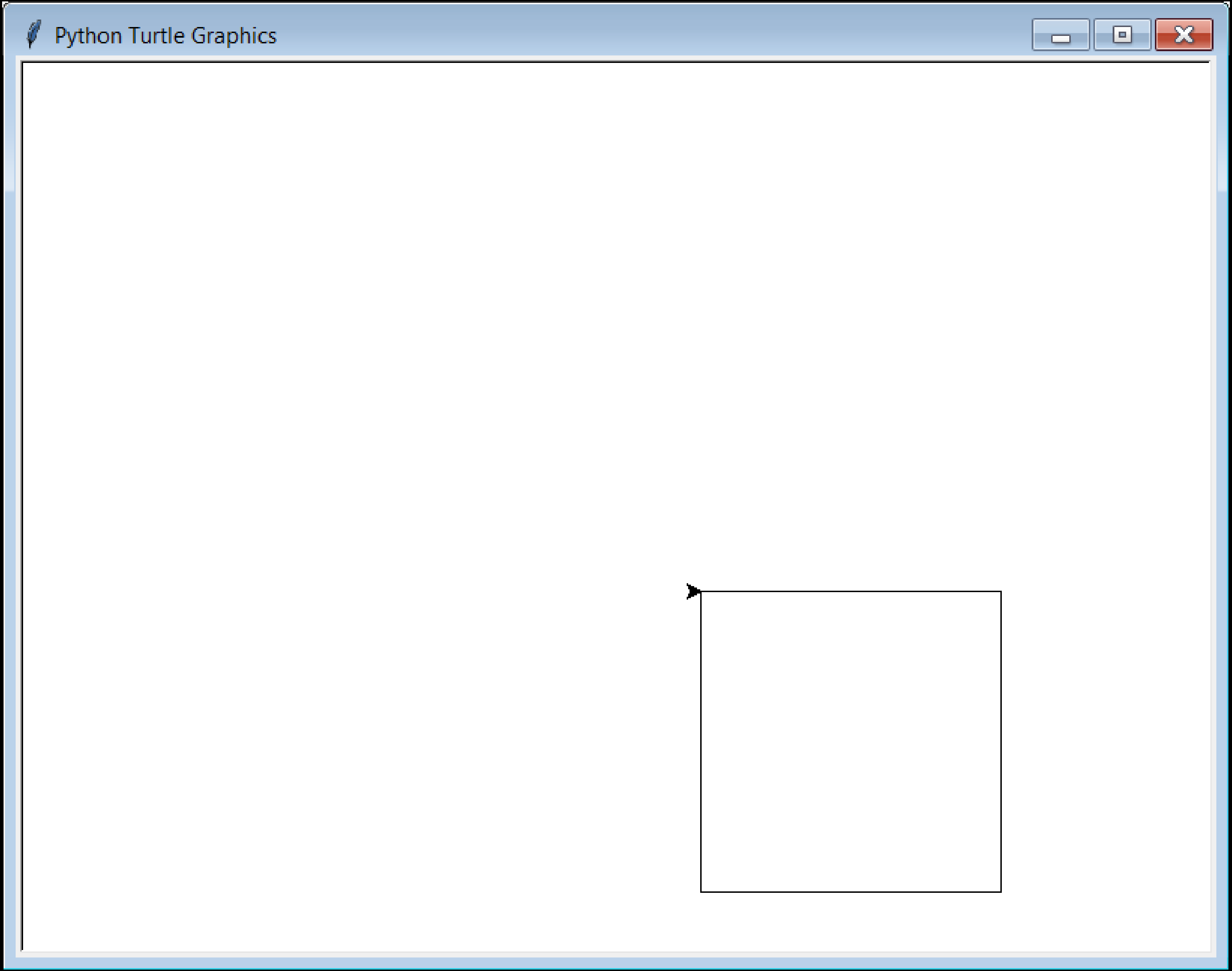


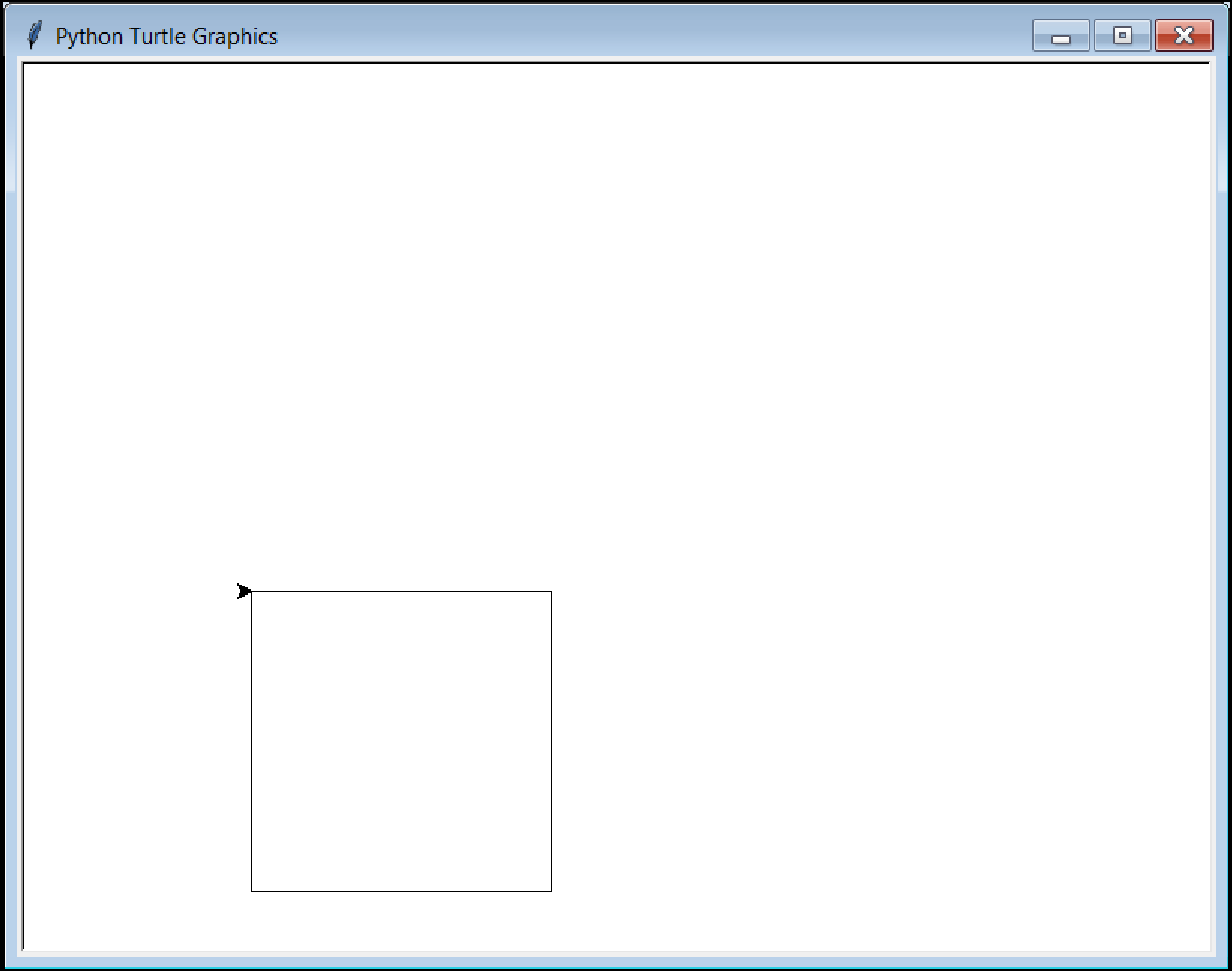
NOTE:
I cannot really display the output of this program as each square is erased the moment it is drawn. You really need to execute this program on your computer to see and appreciate the output.

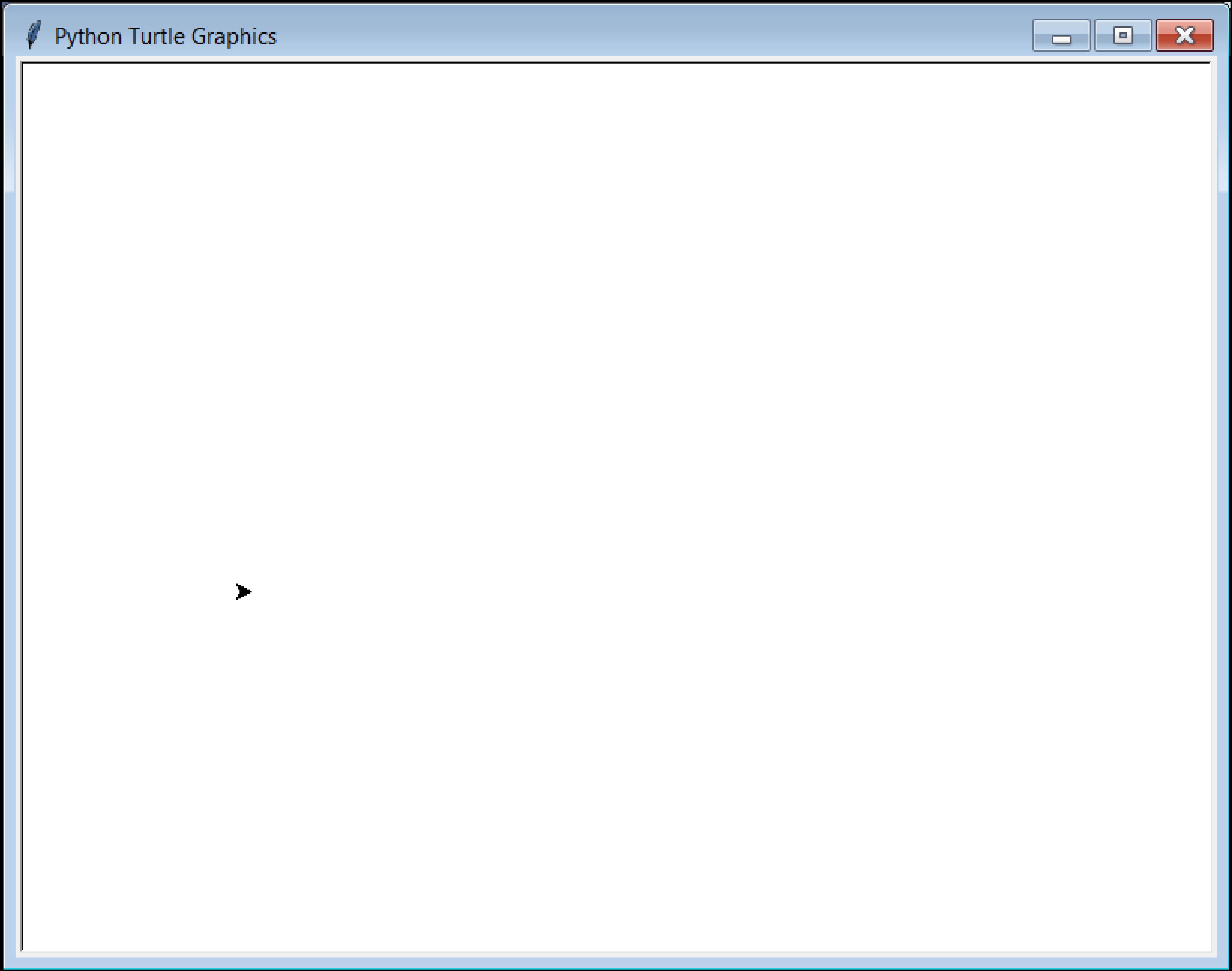
```python
# TurtleGraphics19.py
# This program imports the <sleep> command from
# the <time> library which allows you to make
# the turtle pause or "sleep" for a specified
# number of seconds.  Now each square stays on
# the screen for a full second before it is erased


from turtle import *
from time import sleep

setup(800,600)

penup()
left(135)       # face North-West
forward(350)    # Move to top-left corner
right(135)      # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

sleep(1)
clear()

penup()
forward(300)    # Move to top-right corner
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

sleep(1)
clear()

penup()
right(90)       # face South
forward(300)    # Move to bottom-right corner
left(90)        # face East again
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

sleep(1)
clear()

penup()
backward(300)   # Move to bottom-left corner
pendown()

# draw square
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)
forward(200)
right(90)

sleep(1)
clear()

update()
done()
```


Python Turtle Graphics

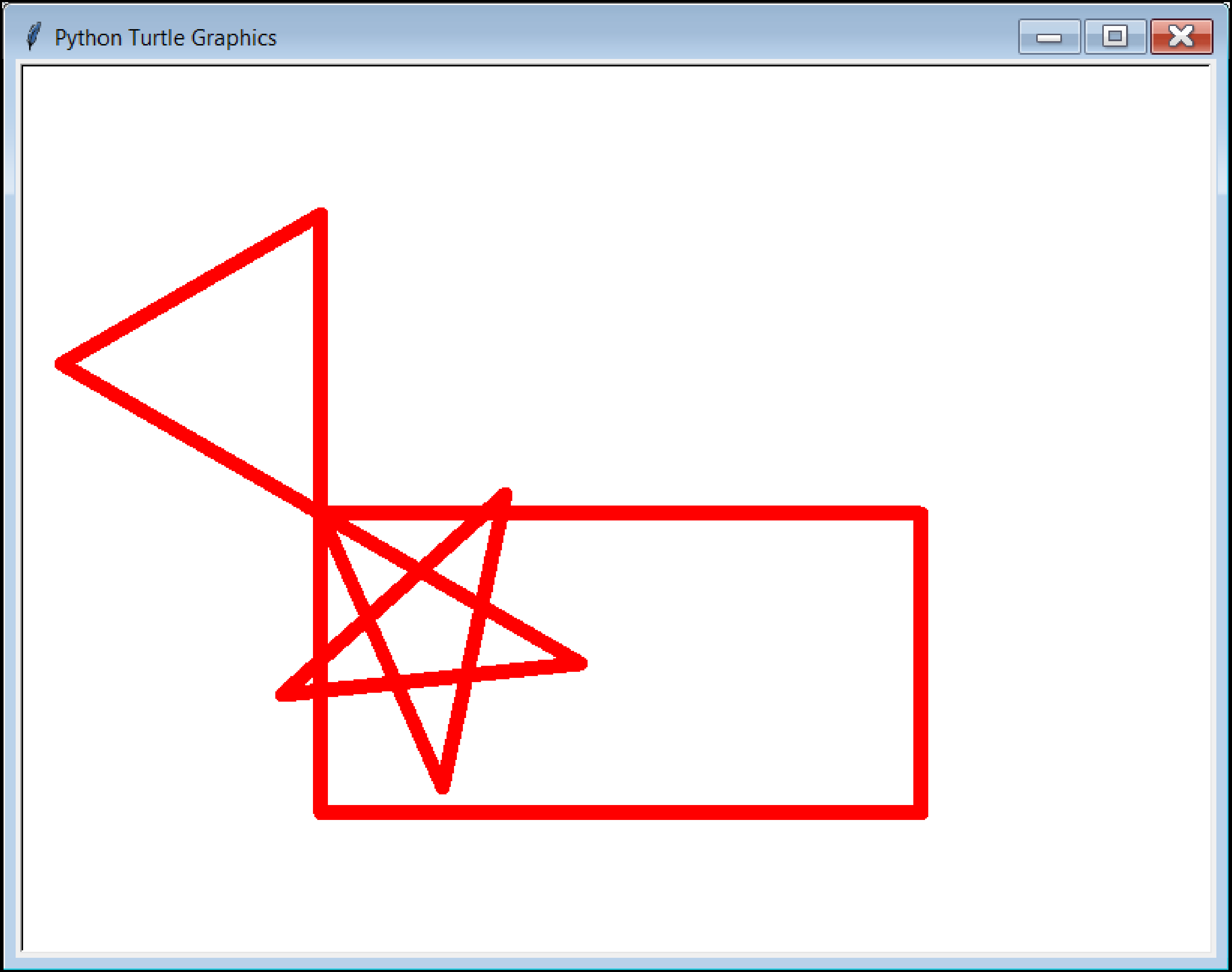

Python Turtle Graphics


Python Turtle Graphics

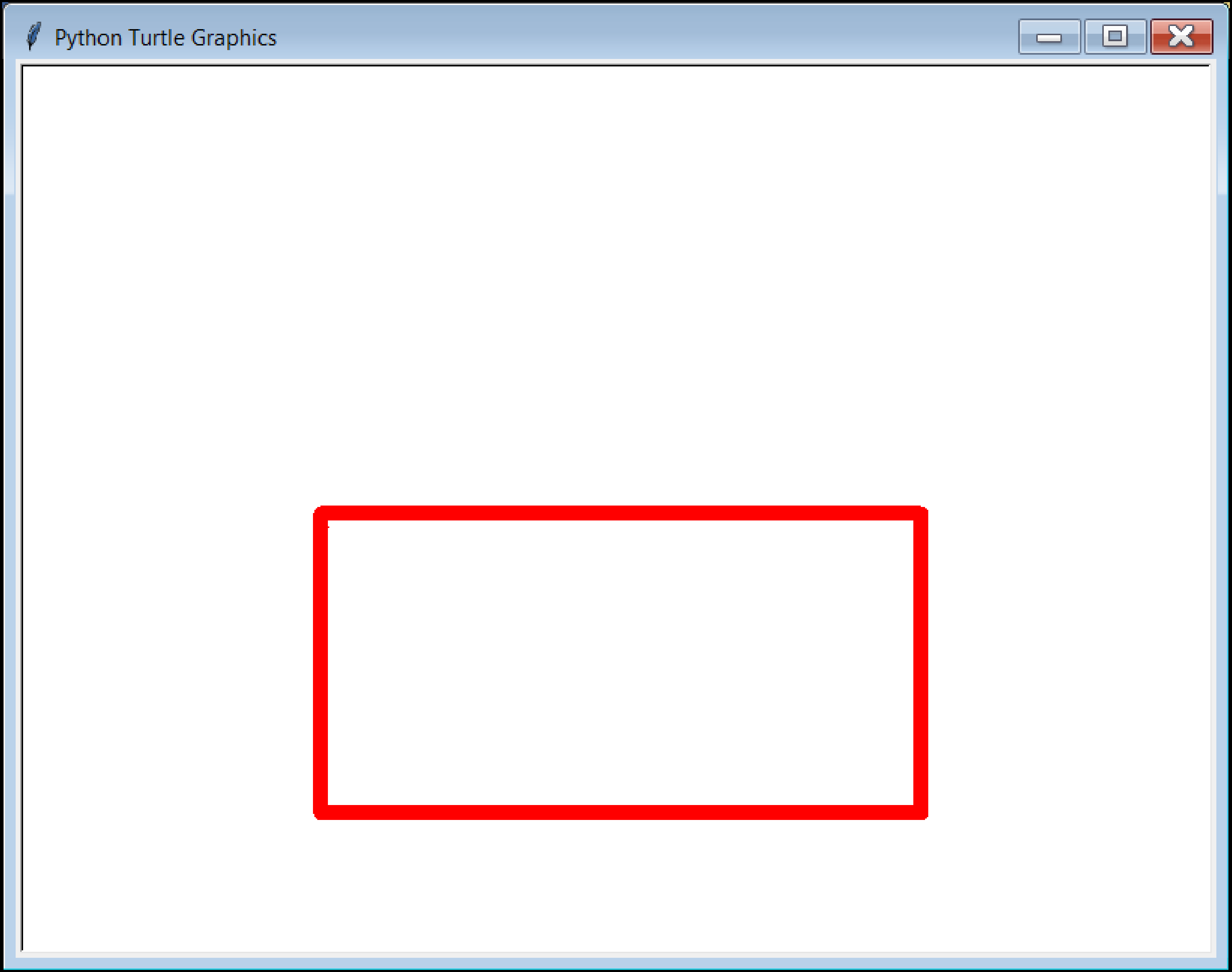

Python Turtle Graphics

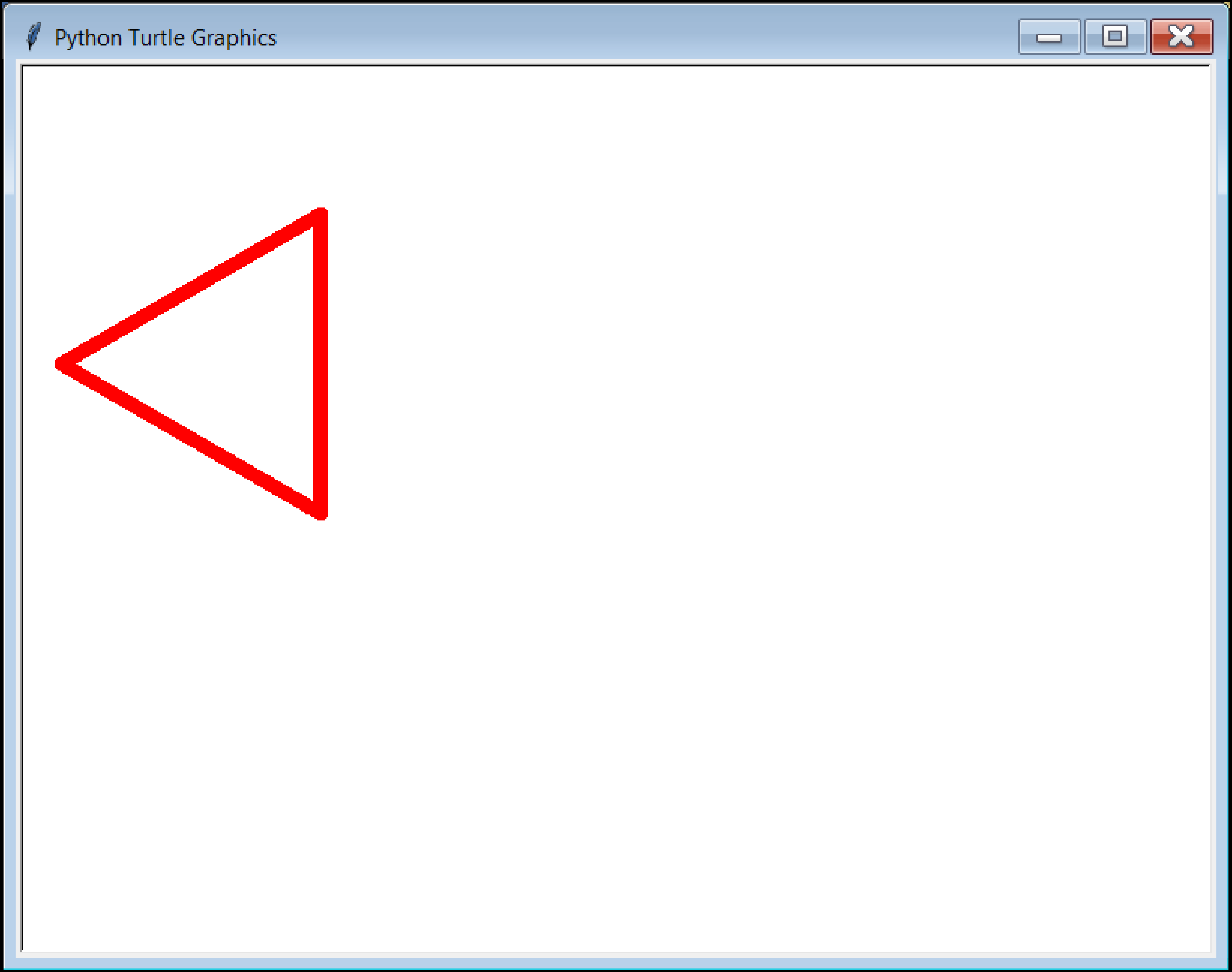

Python Turtle Graphics

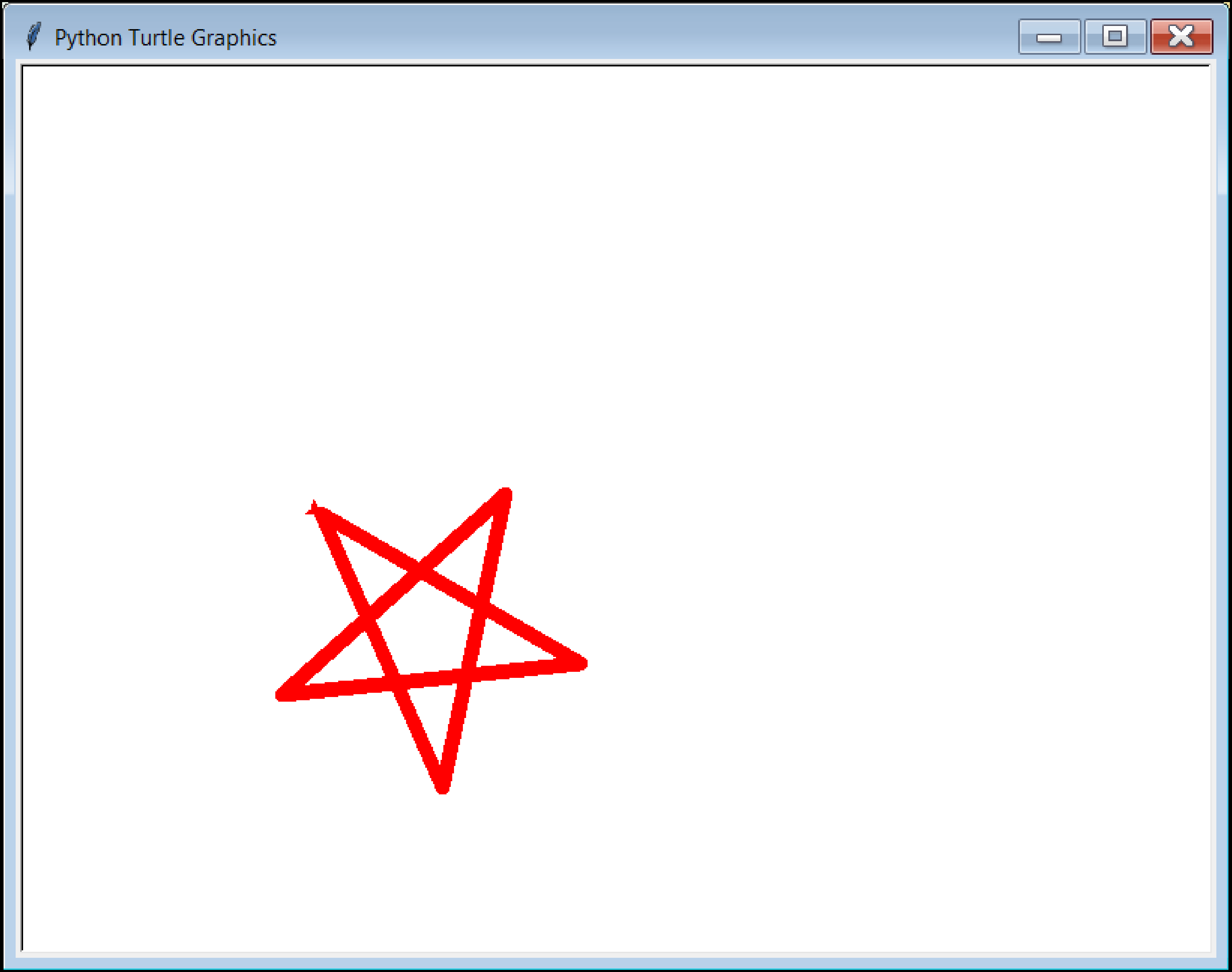
```python
# TurtleGraphics20.py
# This program draws a rectangle, a triangle,
# and a star.   Note how the way one shape
# finishes affects the location and rotation
# of the next shape.

from turtle import *

setup(800,600)
width(10)
color("red")

# rectangle
backward(200)
forward(400)
right(90)
forward(200)
right(90)
forward(400)
right(90)
forward(200)

#triangle
forward(200)
left(120)
forward(200)
left(120)
forward(200)

# star
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)

update()
done()
```


Python Turtle Graphics

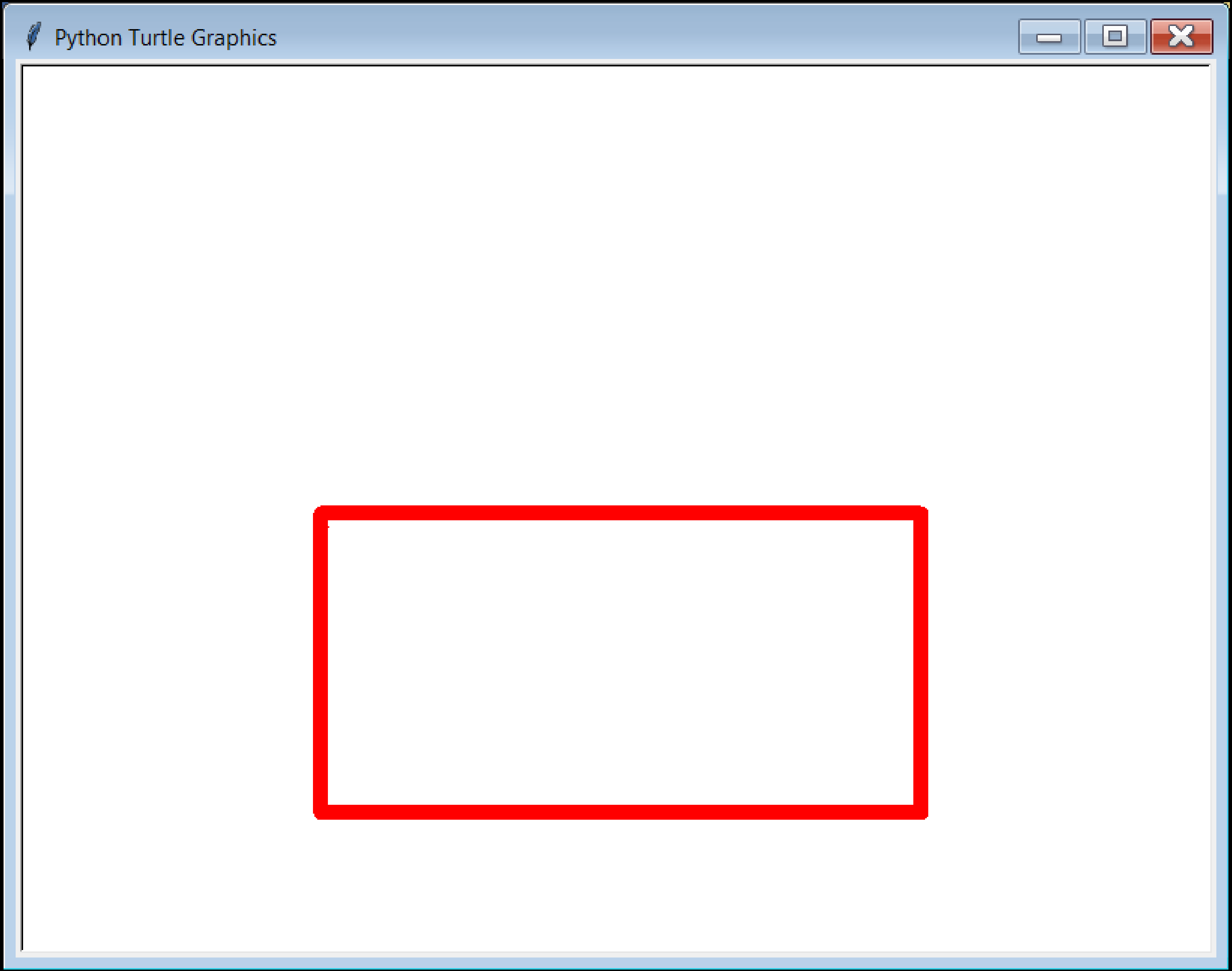
```
# TurtleGraphics21.py
# This program draws the same shapes from the
# previous program, but uses a <clear> command
# after each shape is drawn.  Note how clearing
# the window this way has no effect on the
# shape's location, orientation, color or width.

from turtle import *
from time import sleep

setup(800,600)
width(10)
color("red")

# rectangle
backward(200)
forward(400)
right(90)
forward(200)
right(90)
forward(400)
right(90)
forward(200)

sleep(1)
clear()

#triangle
forward(200)
left(120)
forward(200)
left(120)
forward(200)

sleep(1)
clear()

# star
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)

sleep(1)
clear()

update()
done()
```

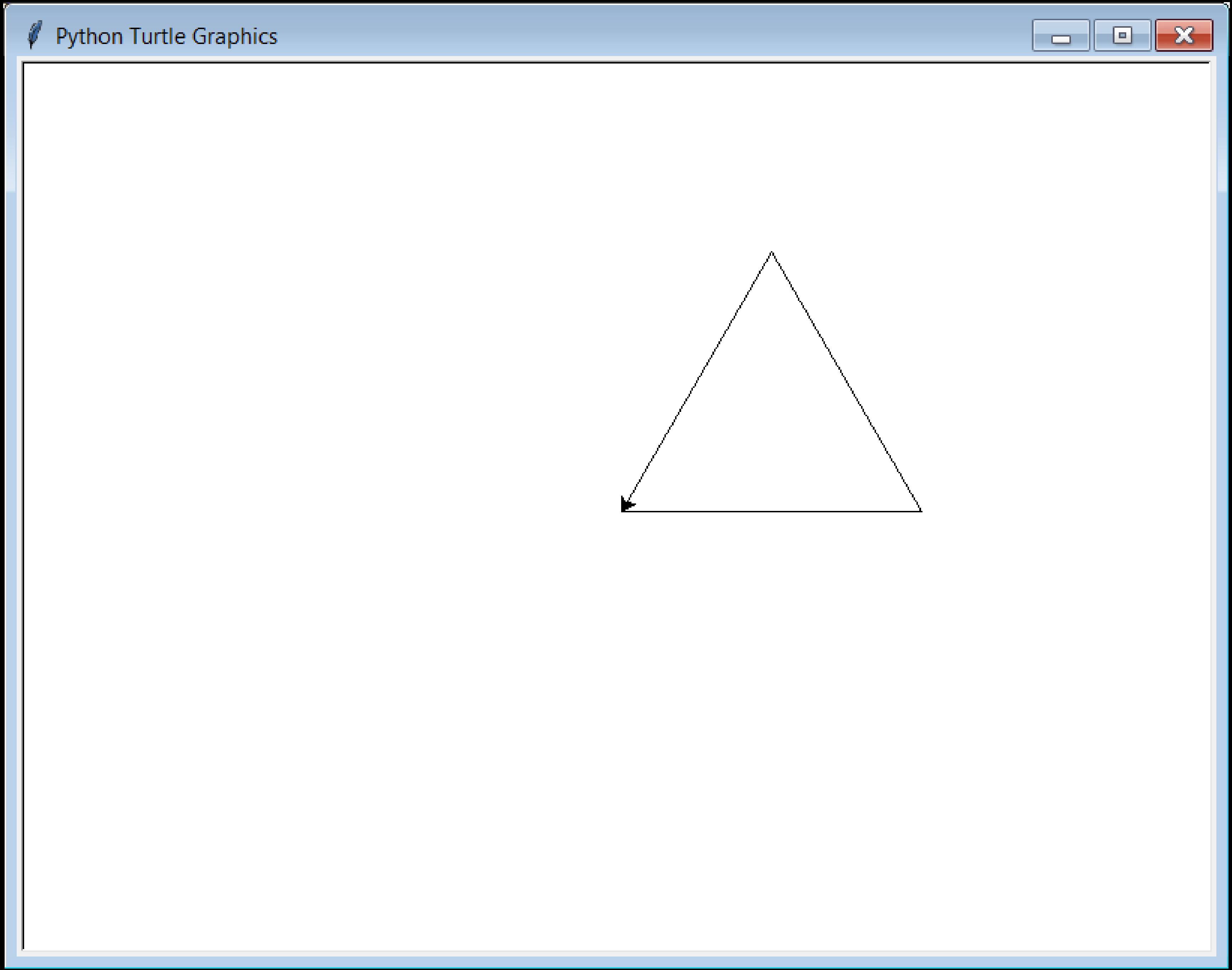

Python Turtle Graphics

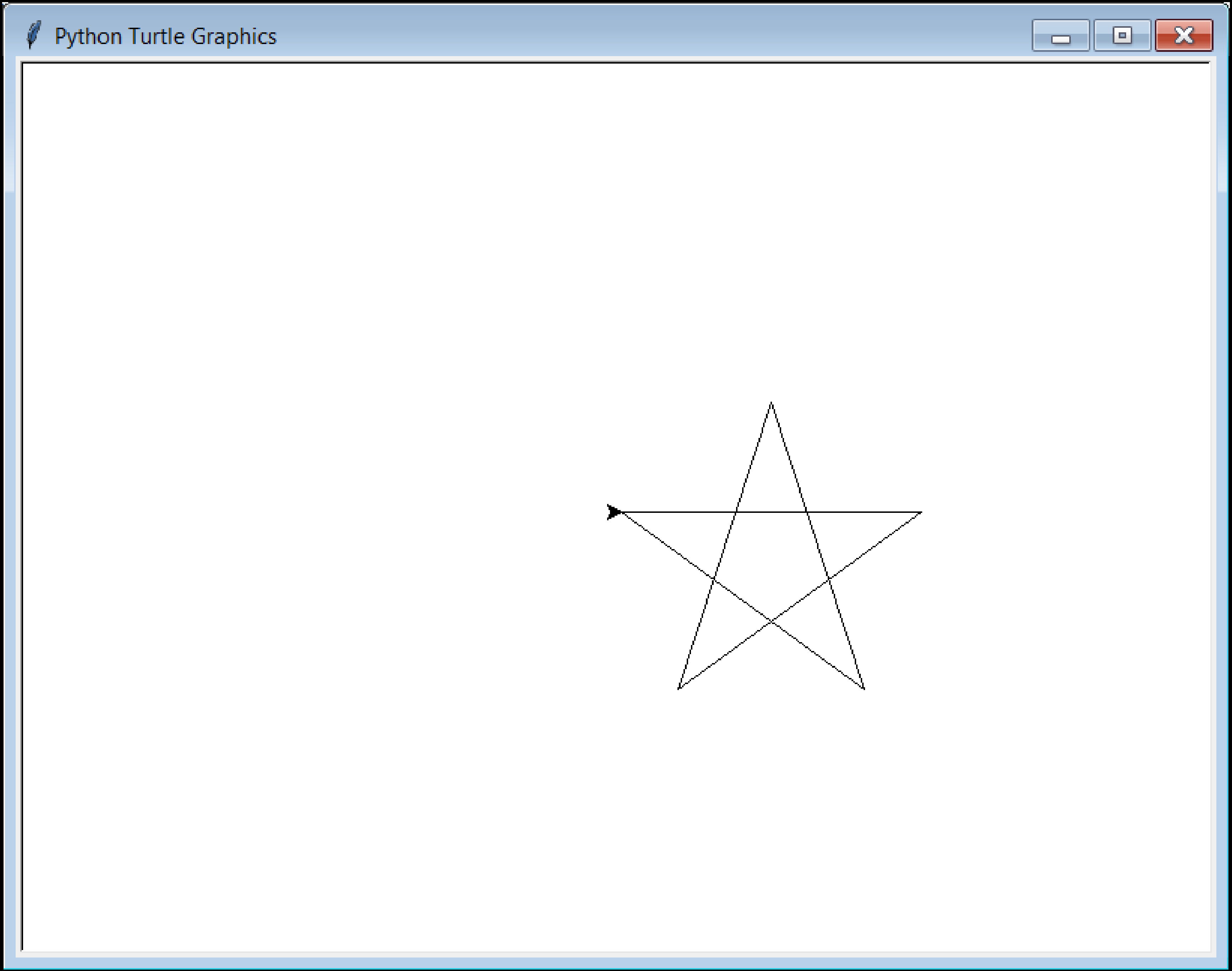

Python Turtle Graphics


Python Turtle Graphics

```python
# TurtleGraphics22.py
# This program replaces the <clear> commands
# of the previous program with <reset>.
# The <reset> command does more than clear the
# screen.  It also returns the turtle back to
# the center of the screen and makes it face
# right, sets the <width> value back to 1,
# and sets the <color> value back to "black".

from turtle import *
from time import sleep

setup(800,600)
width(10)
color("red")

# rectangle
backward(200)
forward(400)
right(90)
forward(200)
right(90)
forward(400)
right(90)
forward(200)

sleep(1)
reset()

#triangle
forward(200)
left(120)
forward(200)
left(120)
forward(200)

sleep(1)
reset()

# star
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)
forward(200)
right(144)

sleep(1)
reset()

update()
done()
```


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics

clear vs. reset

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

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.