“Selected Programming Language: Introduction to Python”

CS 427

Faculty of Science, Department of Mathematics

Ref:

Richard L. Halterman, LEARNING TO PROGRAM WITH PYTHON, 2011

Richard L. Halterman, Fundamentals of Python Programming, 2017

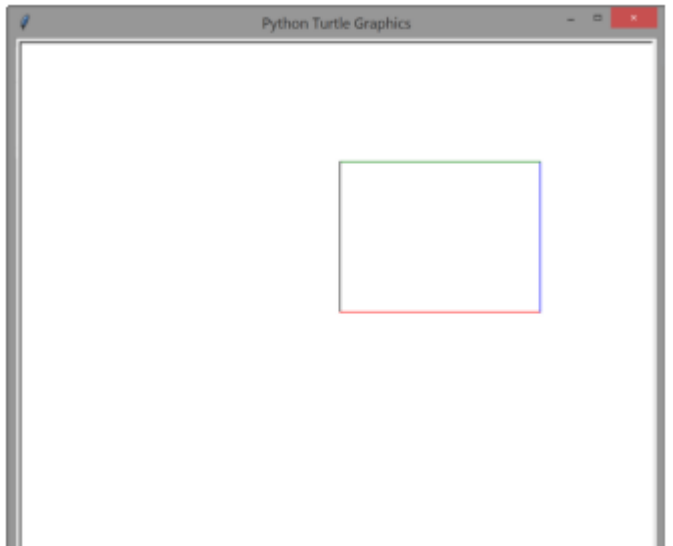
http://python.cs.southern.edu/pythonbook/pythonbook.pdf

Chapter 6*

Turtle Graphics

Turtle Graphics

- Turtle graphics on a computer display mimics these actions of placing, moving, and turning a pen on a sheet of paper.
- It is called Turtle graphics because originally the pen was represented as a turtle moving within the display window.
- Python includes a Turtle graphics library that is relatively easy to use.



Turtle Graphics

boxturtle.py - E:/courses/Python course/codes/boxturtle.py (3.7.2)

File Edit Format Run Options Window Help

```
#Draws a rectangular box in the window
import turtle
turtle.pencolor('red')       # Set pen color to red
turtle.forward(200)          # Move pen forward 200 unit
turtle.left(90)              # Turn pen by 90 degrees
turtle.pencolor('blue')      # Set pen color to blue
turtle.forward(150)          # Move pen forward 150 unit
turtle.left(90)              # Turn pen by 90 degrees
turtle.pencolor('green')     # Set pen color to green
turtle.forward(200)          # Move pen forward 200 unit
turtle.left(90)              # Turn pen by 90 degrees
turtle.pencolor('black')     # Set pen color to black
turtle.forward(150)          # Move pen forward 150 unit
turtle.hideturtle()          # Make the pen invisible
turtle.exitonclick()         # Wait for user input
```

Turtle Graphics

- Few of the functions provided by the turtle module:

• setheading: sets the pen to face in a particular direction

• pencolor: sets pen’s current drawing color.

• forward: moves the pen forward a given distance at its current heading

• left: turns the pen to the left by an angle specified in degrees.

• right: turns the pen to the right by an angle specified in degrees.

• setposition: moves the pen to a given (x; y) coordinate within the graphics window. Draws a line from the previous position unless the pen is up.

Turtle Graphics

- title: sets the text to appear in the window's title bar
- penup: disables drawing until the pen is put back down (allows turtle movements without drawing)
- pendown: enables the pen to draw when moved
- hideturtle: makes the pen object (turtle) itself invisible; this does not affect its ability to draw.
- tracer: turns off tracing (drawing animation) when its argument is zero—this speeds up rendering.

Turtle Graphics

- update: renders all pending drawing actions (necessary when tracing is disabled).
- done: ends the drawing activity and waits on the user to close the window.
- exitonclick: directs the program to terminate when the user clicks the mouse over the window.
- mainloop: used in place of done to enable the graphics framework to handle events such as user mouse clicks and keystrokes.

Turtle Graphics

- If you are annoyed that the drawing is too slow, you can speed up the rendering process in several ways. You can add the following statement before moving the pen:

```
turtle.speed(0)     #  Fastest turtle actions
```

- The speed function accepts an integer in the range 0. . . 10. The value 1 represents the slowest speed, and the turtle's speed increases as the arguments approach 10. Counterintuitively, 0 represents the fastest turtle speed.

Turtle Graphics

- The permissible strings correspond to the following numeric values:

• "fastest" is equivalent to 0

• "fast" is equivalent to 10

• "normal" is equivalent to 6

• "slow" is equivalent to 3

• "slowest" is equivalent to 1

Turtle Graphics

- The delay function provides another way to affect the time it takes to render an image.
- Rather than controlling the overall speed of the turtle's individual movements and/or turns, the delay function specifies the time delay in milliseconds between drawing incremental updates of the image to the screen.

```
turtle.delay(500)
```

- The numeric values are in milliseconds.

Refactoring

- In a code, two sections exhibit the same similarities, this code represents duplication of coding effort, and this is known as *code duplication*.
- Code duplication is undesirable for several reasons:
  - The extra code to write requires more work on the part of the programmer.
  - Code duplication results in code that is more difficult to maintain.
- Functions are ideal for consolidating duplicate code. Through a process known as *code refactoring*.
- A programmer can place the common code within a function definition and tune the minor variance in behavior of the duplicated code via parameters.
- Any correction or enhancement made to the code within the function automatically will be available to all the callers of that code. This removes the possibility of introducing an inconsistency into the application.

Refactoring to Eliminate Code Duplication

Listing 7.23: noduplication.py

```
import turtle


def draw_lines(color, y):
    """ Draws 10 horizontal lines of a given color stacked
        on top of each other with the lowest line appearing
        at position y on the y axis. """
    turtle.color(color)
    for x in range(10):
        turtle.penup()
        turtle.setposition(-200, y)
        turtle.pendown()
        turtle.forward(400)
        y += 10
```

```
# Turn off animation
turtle.tracer(0)
draw_lines("red", -200)
draw_lines("blue", -100)
draw_lines("green",0)
draw_lines("black", 100)

turtle.update()     # Ensure all of image is drawn
turtle.done()
```

Thank you