2019

Activity - Python FUNCTIONS - Drawing with Turtle

Robert J. Domanski
City University of New York - Graduate Center, rdomanski@hotmail.com

Python Activity: FUNCTIONS
"Drawing with the Turtle Library"

Learning Objectives

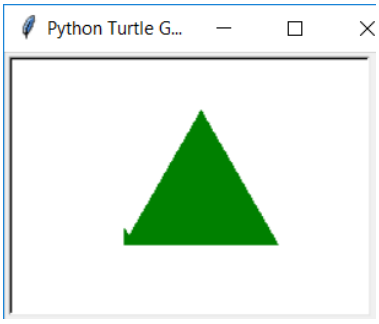
Students will be able to:

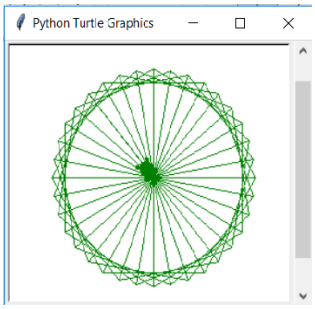
- Implement the logic of pre-defined functions and parameters
- Understand the role of programming libraries
- Write code using the Python syntax for functions
- Demonstrate debugging skills

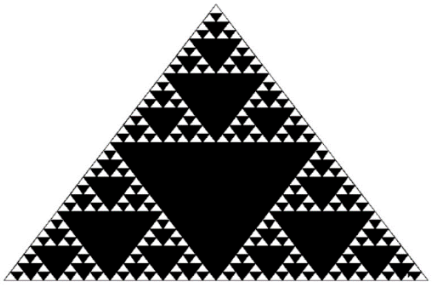
In programming, libraries are collections of pre-written functions. You can use (or "call") functions that act like commands to make your program do something specific.

One such Python library is named "Turtle". The Turtle library enables you, as the programmer, to make drawings and computational artwork that are displayed in a graphical window. These can be fairly simple or far more complex. They can even be animated.

For instance, we can draw a simple triangle, use loops to draw a spirograph of many triangles, or use something called recursion to draw self-similar patterns of triangles ("Sierpinski triangles"):







Let's start simple. To draw a green triangle, the code would look like this in Python:

```
from turtle import *

color('green')
begin_fill()

forward(100)
```

```
left(120)
forward(100)
left(120)
forward(100)

end_fill()
done()
```

Let's explain this code. To access all of the functions within the Turtle library, we need to include the first line at the top of our Python code: **from turtle import ***

We use the **forward** function to draw a line forward 100 pixels. All functions need parentheses; some functions leave their parentheses blank (like the **done** function at the bottom of the code) while others require data inside of them (called the parameters, or arguments). In this case, the **forward** function requires a parameter indicating the distance in pixels that it should move, and the **left** function requires a parameter indicating how many degrees to turn left.

Here are some other common functions within the Turtle library:

```
forward(100)        # move that distance
backward(100)       # move that distance
right(90)           # turn by degrees
left(90)            # turn by degrees
color('blue')       # change the line color
begin_fill( )       # start filling the shape with a color
end_fill( )         # stop filling the shape with color
goto(x,y)           # teleport to different coordinates
penup( )            # lift the pen up (stop drawing)
pendown( )          # put the pen down (start drawing)
shape('turtle')     # change the look of the pen
```

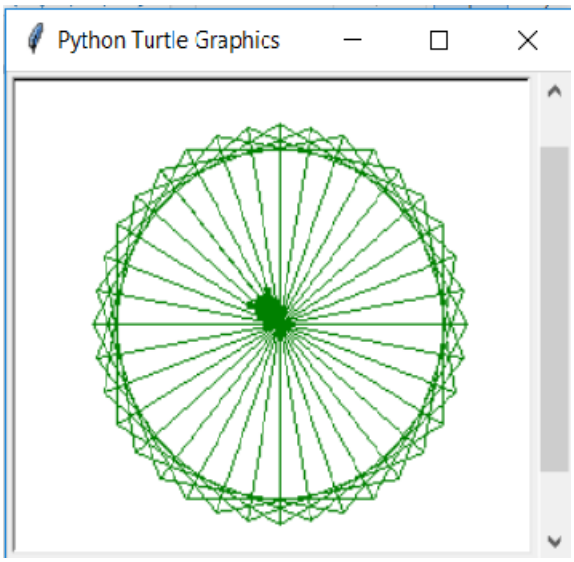
1. Type the code above, save the file with a .py extension, and run the program in the Terminal. If you receive any error messages, determine what the problems are likely to be and debug your code. Once you can run the program with no error messages, try it out. Does it work?

2. ASSIGNMENT: Draw a house. At the very least, this consists of drawing a square then drawing a triangle on top of it. Feel free to be creative and add colors, windows, doors, or anything else you want.

3. ASSIGNMENT: Create your own Turtle Art! You have free reign to be creative and draw anything that your imagination desires. The caveat: you must use as many of the Turtle functions listed above as possible.

4. CHALLENGE ASSIGNMENT: Let the user choose the shape they would like to draw. Ask the user how many sides they would like their shape to have, then draw a polygon with that many sides. All of the sides should be of equal length, and you will need to adjust your angles accordingly.

5. CHALLENGE ASSIGNMENT: Create a spirograph! Using loops, start by drawing a single shape (ex. – a square) then rotate by a set number of degrees and draw it again, repeating this until the pattern is complete. Here is an example of spirographed triangles:



6. CHALLENGE ASSIGNMENT: Create a spiral pattern! Using loops and beginning at the center of the pattern, create a spiral image like the following: