CS 8 Lab: pa02

October 22, 2014

1 Overview

This week we will have fun drawing shapes with the turtle module. We will write 5 different functions all saved into one file. This pdf has suggested steps for completing the assignment.

2 Getting Set Up

Pull up the assignment description from `http://cs.ucsb.edu/~koc/cs8/hwexpa/pa02.html` and read through it.

Open up a new python file called `pa02.py` (use past handouts for help if needed). As the first three lines, type in comments containing your full name, lab section time, UCSB UMail email address, and perm number:

```
# Your name, Lab time
# Your UCSB email address
# Your perm number
```

In this programming assignment, you have to write five functions that use the turtle module. Make sure you define them exactly as described on the assignment description. For now, type all of these function definitions in your file with "filler" code. In the code below, each function tells myTurtle to move forward 50 pixels. This is called a **stub**. Stubs can be thought as "filler" code that you put in as placeholder before you type in the actual code. Note that you'll also have to import the turtle module. Later on you may also find that you need to import the math module. Your file should look something like this:

```
import turtle

#drawPentagon draws a regular pentagon with side length d
def drawPentagon(myTurtle, d):
    myTurtle.forward(50)

#drawPolygon draws a regular n-gon with side length d
def drawPolygon(myTurtle, n, d):
    myTurtle.forward(50)

#drawStar draws a regular 5-pointed star with side length d
def drawStar(myTurtle, d):
    myTurtle.forward(50)

#drawStarX draws a regular n-pointed star with side length d
def drawStarX(myTurtle, n, d):
```

```
    myTurtle.forward(50)

#drawTheodorusSpiral draws an n-corner Spiral of Theodorus with side length d
def drawTheodorusSpiral(myTurtle, n, d):
    myTurtle.forward(50)
```

Save your file.

2.1 Running on the command line

If you're on a computer without IDLE or just want to run this on the command line, open up a Terminal window and navigate into the directory where you saved your pa02.py file. Type `python3` and press enter. You are now in the Python shell/command line environment. Remember you can exit at any time by typing `quit()`. After you type `python 3`, type `import pa02` to import your file. To make a new turtle called Alice (you can name the turtle whatever you want) for testing, type:

```
Alice = pa02.turtle.Turtle()
```

This should make a new window pop up, with a triangle-like cursor in the middle. Say hello to your new friend! To test what you have in one of your functions, type something like:

```
pa02.drawPentagon(Alice, 10)
```

This should make your turtle move 50 pixels forward (if you're using the stub myTurtle.forward(50) in your functions. You can try this with any of the functions, just make sure you pass in a turtle where the function has "myTurtle" and make sure you pass in the correct amount of numbers (some functions require two numbers, some require one).

(Not required) Try doing the following when you make your turtle, and see what happens:

```
Alice = pa02.turtle.Turtle(); Alice.shape('turtle')
```

2.2 Running in IDLE

If you're on a computer with IDLE, go to `Run`→`Run Module` where you have your file. Then, over in the IDLE prompt (the window with the `>>>`), type `import pa02`. To make a new turtle called Alice (you can name the turtle whatever you want) for testing, type:

```
Alice = turtle.Turtle()
```

This should make a new window pop up, with a triangle-like cursor in the middle. Say hello to your new friend! To test what you have in one of your functions, type something like:

```
drawPentagon(Alice, 10)
```

This should make your turtle move 50 pixels forward (if you're using the stub myTurtle.forward(50) in your functions. You can try this with any of the functions, just make sure you pass in a turtle where the function has "myTurtle" and make sure you pass in the correct amount of numbers (some functions require two numbers, some require one).

(Not required) Try doing the following when you make your turtle, and see what happens:

```
Alice = turtle.Turtle()
Alice.shape('turtle')
```

3 Write your code!

Now you want to go into your file and replace the stub `myTurtle.forward(50)` with the correct code. Here are a few general tips:

- Draw out what you want your turtle to do on paper, step-by-step, maybe even multiple times.
- Use your on-paper drawing to calculate angles.
- If you catch yourself typing the same code in a given function multiple times, consider using a for/range loop.

The following subsections have notes/tips regarding specific functions.

3.1 Notes on drawStar

- When drawing the 5-pointed star, we only want the outline of the star (do not include any lines inside of the star).
- I have posted a link on my website to help you with the angle calculations: `http://www.hyperflight.com/images/pentagram_angles.gif`

3.2 Notes on drawStarX

- We assume $n \geq 5$. In other words, the least amount of points on a given star we'll test is 5.
- Given n, the number of points on a regular star polygon, there may be more than one version. To differentiate between these versions, star polygons are defined by $\{p/q\}$ where p is the number of points and q is the density of the star. In our case, $p = n$, and we're restricting the problem so that $q = 2$. What this means for you, and this project, is that the measure of the interior angles of the regular n-star polygon is:

$$\frac{180(p-2q)}{p} = \frac{180(n-2\cdot 2)}{n} = \frac{180(n-4)}{n}$$

That is, given n, you can calculate the measure of the interior angle (in degrees) by finding $\frac{180(n-4)}{n}$. Your stars should look like the ones in the first row of the Star Polygons shown here: `http://upload.wikimedia.org/wikipedia/commons/thumb/7/76/Regular_star_polygons.svg/600px-Regular_star_polygons.svg.png`
If you want to read more about this, you can do so here: `http://en.wikipedia.org/wiki/Star_polygon`

3.3 Notes on drawTheodorusSpiral

- I suggest you read a bit about them on the link in the programming assignment before you being programming it. Read through the construction section on the Wikipedia page: `http://en.wikipedia.org/wiki/Spiral_of_Theodorus`
- Please do include all lines, including the hypotenuse of each triangle.
- You'll have to use some trigonometry to calculate the angles (remember: SOHCAHTOA). I suggest you use the python math library for this. You can learn more about the math library here: `https://docs.python.org/3/library/math.html`
- Consider d like a "scaling" factor. You want the lengths of all the lines/sides to scale up by a factor of d.
- If $n = 17$, there should be 16 triangles. The 16th triangle has a hypotenuse of length $\sqrt{17}$.

- When $n \geq 17$, the triangles will start to overlap. If you can get your program to stop drawing new hypotenuses once they hit an earlier triangle, awesome. If not, don't worry. Just have all hypotenuses draw to the center of the spiral. I will not deduct points for this.
- Again, drawing stuff on a sheet of paper really helps. I highly recommend you do this for this function. I drew one triangle at a time to help me visualize it.

4 Testing and Turnin

Make sure you test all of your functions with various n and d values. Follow the procedure outlined in sections 2.1 or 2.2 above for testing.

Ready to submit? Make sure you move your file over to CSIL first (if it isn't there already). Then, in a Terminal, navigate the the directory containing your pa02.py file. To turn in, type the following command:

```
turnin pa02@cs8 pa02.py
```

and follow the on-screen directions. Remember, I will grade the last submission turned in before the deadline if you turn in multiple versions. **The deadline for this project is Thursday, October 23rd, 2014 at 11pm. We will not be accepting late submissions like we did last week, so make sure you give yourself enough time to complete and turn in your project.**