Computer Project # 11

Overview

This project focuses on rendering shapes using Python libraries. You will create functions in this project as well as complete a drawing function. This assignment is worth 55 points (5.5% of the course grade) and must be completed and turned in before 11:59 PM on Monday, April 25, 2016.

You will use Turtle graphics to draw a picture containing multiple shapes of multiple colors and arranged to correspond to your student ID number. We will provide half of the shapes required by the program and ask you to use your imagination to draw the remaining half. The rest of the project will require you to complete skeleton code. The final output of the program will be a shape that will correspond to your student ID number.

Background

The artist Allan McCollum conceived the Shapes Project to create a unique two-dimensional shape for every person on the planet. He has developed a scheme to generate 31 billion unique shapes – more than enough for the projected population of earth. He uses his scheme to design hand drawn shapes. In this project we will use a variation of his model to create a program that can generate unique shapes based on an MSU student ID. The To know more about the background for this project, please refer to: The Shapes Project
(http://allanmccollum.net/amcnet2/album/shapes/McCollum_Shapes.pdf).

We also introduce you to the concept of hashing as part of this project. To read more about hashing, please refer to:
http://www.i-programmer.info/babbages-bag/479-hashing.html

Originally written as a part of the logo programming language, Turtle graphics (http://en.wikipedia.org/wiki/Turtle_graphics) is one of the oldest graphics programs. It is a 2D graphics package that uses a Cartesian coordinate system and a "turtle" which you can imagine has a pen attached to its body. The turtle can move around the plane, drawing as it goes. Python has a module that implements the behavior of the original turtle graphics program and this module is simply called "turtle" (see Appendix B of the text and the comments in the sample file turtleSample.py).

Deliverables

The deliverables for this assignment are the following files:

1. proj11.py (it originally said shapes.py)

Be sure to use the specified file names and to submit them for grading via the **handin system** before the project deadline.

Specification

Download the file `shapes.py` from the project directory. This file contains all the functions required to draw a shape corresponding to your student ID number (the functions are incomplete). We will draw the shape in 4 quarters. Each quarter will be referred to as a quadrant henceforth. Figure 1 shows an example of a shape coming together from its 4 quadrants. The quadrant on the top left will be referred to as Quadrant 1, the quadrant on the bottom left will be Quadrant 2, the quadrant on the bottom right will be Quadrant 3 and the quadrant on the top right will be Quadrant 4. The final shape produced will have equal length and breadth.The `shapes.py` file mainly does the following steps:

1. It asks the user to input the length of the shape.
2. It asks the user to input their student ID.
3. It draws shapes in each of the 4 quadrants by calling their corresponding functions.
4. Performs cleanup tasks at the end

Please observe the code provided to you in `shapes.py` for each of these tasks. Please observe that every `drawQ` function requires a shape value to be calculated. This is to be done as follows:

> We assume every student ID consists of a letter followed by 8 digits. You need to ignore the letter and focus exclusively on the digit ID value. We map the 8 digits to the 4 quadrants by grouping every 2 digits into a 2 digit number. e.g. the student ID "A12345678" uses the number 12 for Quadrant 1, the number 34 for Quadrant 2, the number 56 for Quadrant 3 and the number 78 for Quadrant 4. For every quadrant, we calculate a shape value by hashing the quadrant's corresponding 2 digit number into a single digit value. This hashing is performed by adding up the digits of the quadrant's corresponding 2 digit number and dividing the sum by 10. The hashed value is the remainder of

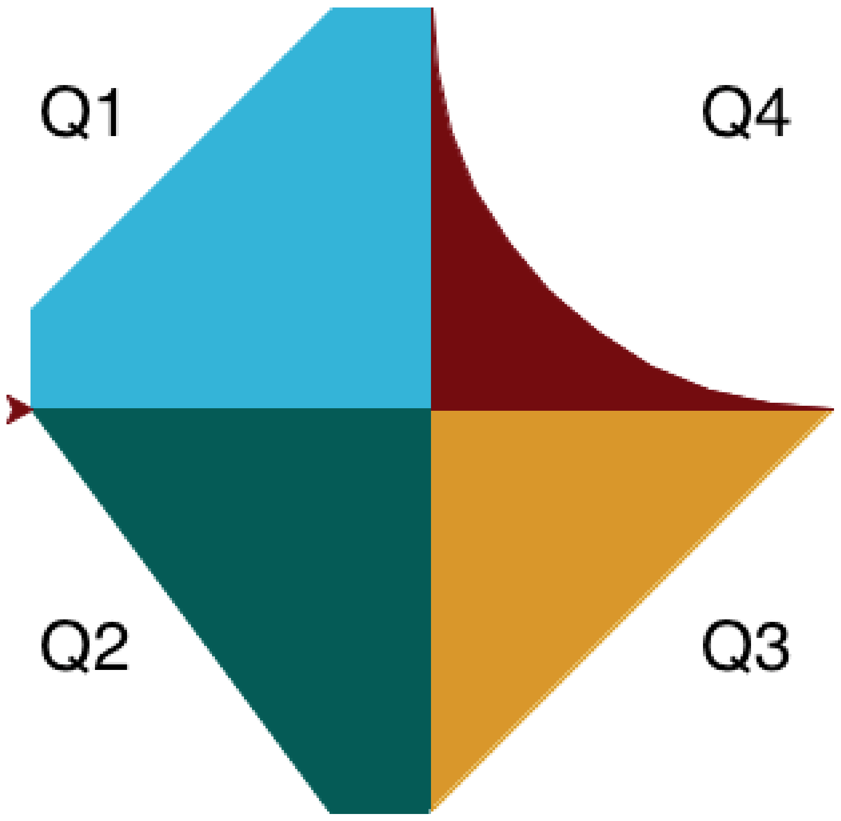

Figure 1: A shape corresponding to student ID A12345678

this division. e.g. if Quadrant 1's corresponding 2 digit number is 12, we can derive its shape value as 1+2 which equals 3 and on dividing 3 by 10, we get a remainder of 3. Similarly, the hash value for the 2 digit number 98 would be 7. This calculation of the final hash value is performed in each `drawQ` function and the derived hash value is passed as the shape to the `drawShape` function.

Please note that you are to modify code only by replacing the comments given in the code. We have indicated sections in `shape.py` *where we expect code to be added in by you.*

The functions in `shape.py` to be filled in by you are as follows:

1. `def drawShape(shape, side_length):` This function draws the shape as indicated in the `shape` argument assuming a side length as indicated in the `side_length` argument. One key assumption made by this function is that drawing is to be always started by moving forward by `side_length` pixels, turning left by 90 degrees and moving forward once again by `side_length` pixels. Since the `shape` parameter can have a value ranging from 0 to 9, we need to draw a different shape for each of these 10 values. We have drawn 5 of the shapes for you and expect you to draw the remaining 5. The only requirement for each of the 5 shapes is that they should be shaped differently and they cannot be same as the 5 shapes already provided in `drawShape`.

2. `def getHash(number):` This function has to be passed a 2 digit number as an argument and it returns the corresponding hash value for the argument. The calculation of the hash value is performed as described earlier.

3. `def drawQ1(identifier, side_length):` This function calls the `getHash` function to find the shape value corresponding to Quadrant 1 by using the passed argument value in `identifier`, positions `turtle` so that the pen is at the bottom left corner of the shape and then calls the `drawShape` function to draw the corresponding shape. Please note that the `identifier` argument would contain the entire value of the identifier. e.g. if the user gave their student ID as "A12345678", then this is the value that would be contained in the `identifier` argument. It is fine to strip off the leading letter "A" before passing the ID as an argument so that it comes into the function as the string '12345678' in this example.

4. The `drawQ2`, `drawQ3` and `drawQ4` functions perform operations similar to `drawQ1` for their corresponding quadrants. However, the way

to position the turtle pen correctly will have to be done differently for different quadrants since the way in which `drawShape` starts drawing cannot be modified by you.

Notes

1. To facilitate automatic testing of your code you must prompt first for the side and then for the ID.
2. Error checking. You need to error check the input so erroneous input will be caught and the program will reprompt for input. The user must successfully enter a side length before you prompt for the student ID. An error in the student ID will cause a reprompt only for the student ID, not prompt for the side. You need to determine what errors need to be checked.
3. Be creative on the shapes you provide – yours should not be like others.
4. The primary constraint on shapes is that every shape will have the same side length on each axis so, when combined, the shapes look nice.
5. IMPORTANT: There is a bug with turtle and Spyder so that every other time you draw with turtle it generates an error. You do not need to restart – simply run the program again.