ISTA 130: Fall 2015
Programming Assignment 2
(100 points)
Loopy Turtles

Due: Thursday, September 17th by 11:59 pm
(submit via D2L dropbox)

Please read the instructions below carefully and follow them closely. All problems (and parts of problems) are required except as noted in the instructions below.

Important: Your filenames *must* be identical to the filenames given below. For any functions you are asked to write, you *must* use the *exact* function names given in the descriptions, and you *must* have parameters in the order shown in the description. Otherwise, our tests will fail.

Also Important: Make sure you *always* save a backup of your work somewhere safe (such as your D2L dropbox or dropbox.com).

1 Part I: Functions with Loops (35 points)

It may help you to sketch pictures out on scratch paper before writing any code.

Polygons

Open **your editor** and create a new file called "polygons.py". In the file:

1.) Copy the code from the "template.py" file and paste it into your new file.

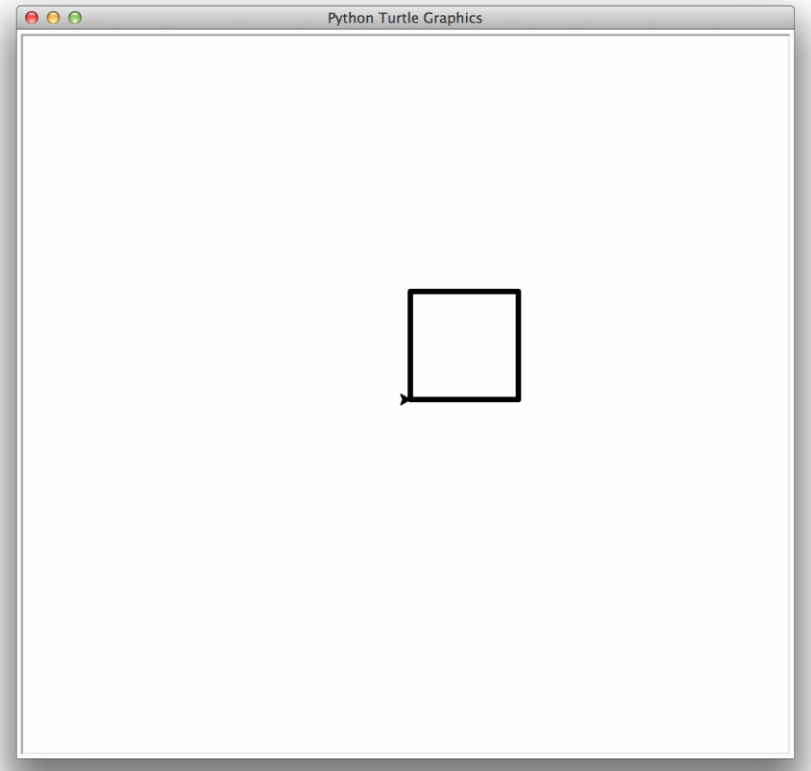
2.) Update the documentation string at the top.

3.) Write a function called `polygon` that has 3 parameters. The first is for a turtle object. The second is for a numeric value giving the number of sides the polygon will have. The third is for a numeric value giving the length of the sides of the polygon. The function draws a *regular* polygon with the given number of sides, each side of the given length. Make left turns only.

- For example, if you were to call the function like this:

```
polygon(some_turtle, 4, 100)
```

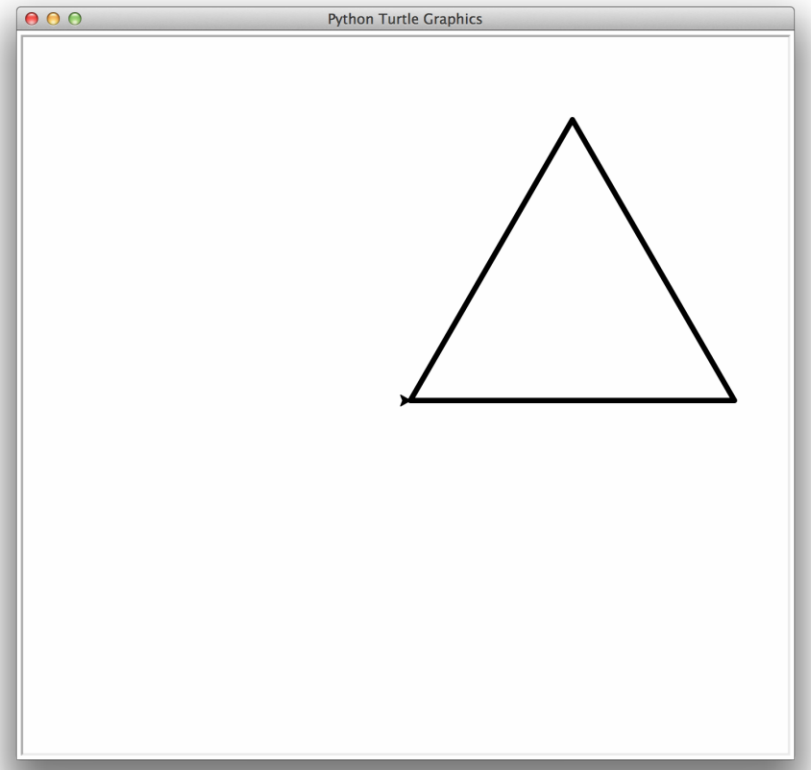
  it would draw a square with sides of length 100:



- If you were to call the function like this:

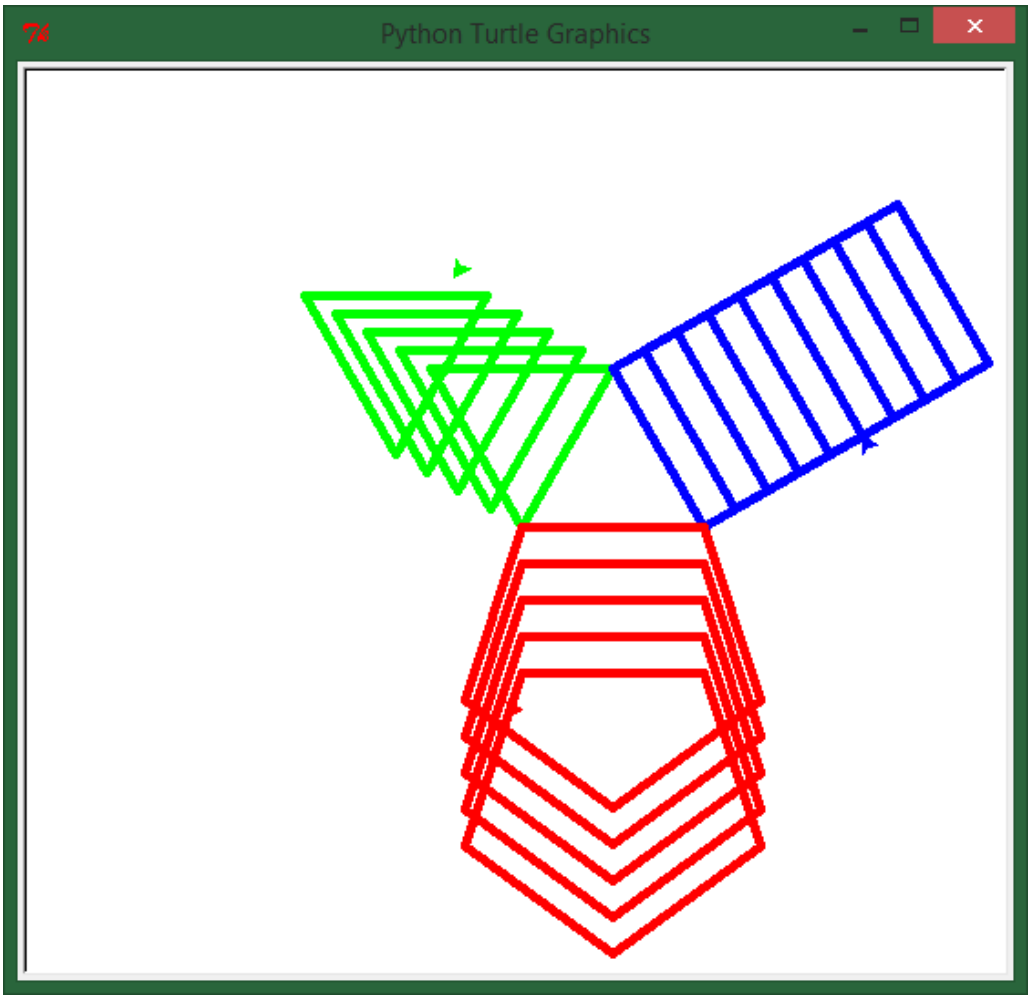
```
polygon(some_turtle, 3, 300)
```

it would draw an equilateral triangle with sides of length 300:

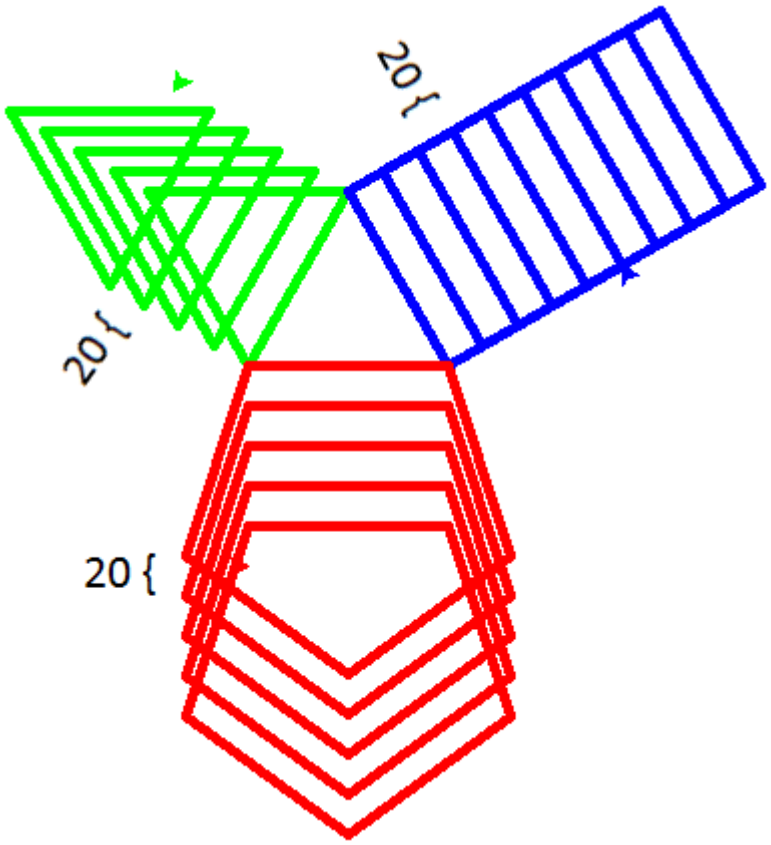


- Note: you'll need a `for` loop to do this. (See the lecture slides and code for examples)
- If you're having trouble, think about drawing a polygon from the turtle's perspective. How many turns do you take? How much do you turn in total? How much do you turn on each individual turn?
  - If you answer these questions first for a triangle, then a square, and then a pentagon, you'll see a pattern. We also derived the answer in class using geometry (see ppt).

4.) In the `main` function definition write code that will draw the following figure (set speed to 0):



- Notes:
  - You'll need to use a `for` loop to repeatedly call your `polygon` function
  - The polygons in the figure are size 100
  - The figure shows 3 sets of 5 polygons (triangles, squares and pentagons)
  - The 5 polygons in a given set are offset from each other by 20.
    - (e.g. the horizontal edge of each of the green triangles is offset by 20 from the next green triangle)
    - The following diagram may help:



- You can choose your own colors, but use three different colors, one for each set of polygons.
- If you need to scale down the sizes to make the figure fit in your drawing window that is fine (e.g. instead of size 100 use 50 and instead of shifting by 20 shift by 10).
- From the figure you can see that three different turtles were used. You don't have to use different turtles, but it will probably make things easier for you.
- It will also be easier if you start by drawing only the red pentagons. The other two sets of polygons are drawn in exactly the same way.

5.) Verify that your documentation makes sense and that you've added documentation to each of your functions.

6.) Verify that your program works

7.) Upload your file to the Program 2 dropbox folder on D2L

2 Part II: More Loops (35 points)

It may help you to sketch pictures out on scratch paper before writing any code.

Castles

1.) Download the "castles.py" program posted on D2L (the same place you found this HW file).

- Run the file to see what it does.

2.) Update the documentation (add your name and your section leader's)

3.) Now make the following changes to the file:

a.) Copy the `polygon` function you wrote in Part 1 above and paste it in this file.

- NOTE: Use only left turns in your polygon function so that it works well with the existing code

b.) Delete the `triangle` function from the code.

c.) Update the code to use the `polygon` function instead of the `triangle` function you deleted.

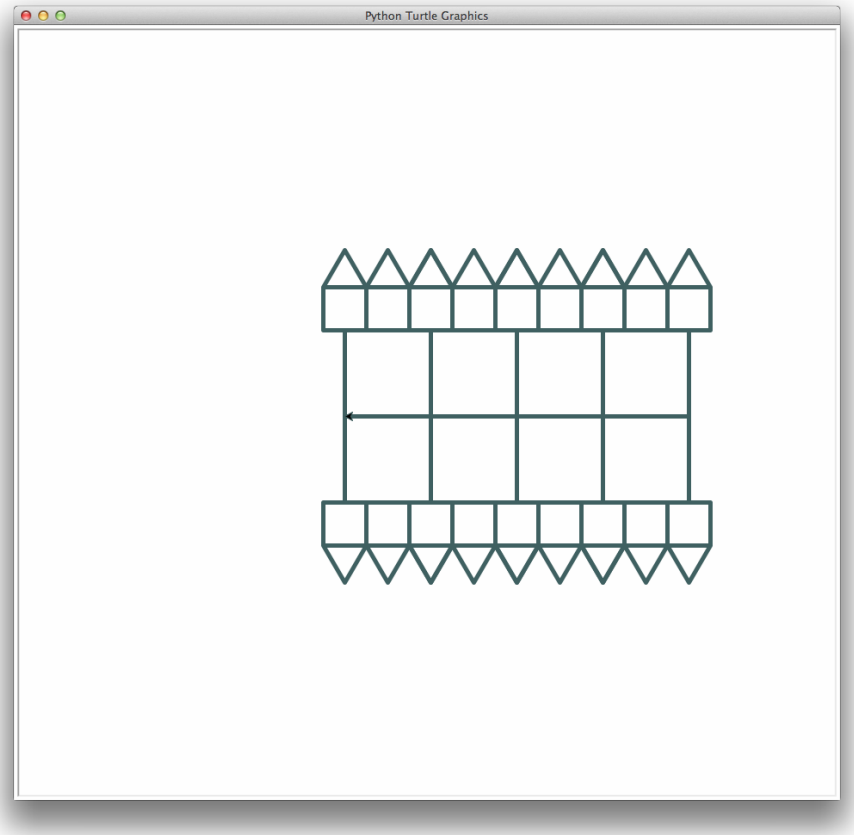
- make sure it works before you go to the next step!

d.) Delete the `square` function from the code.

e.) Update the code to use the `polygon` function to draw any squares.

- make sure it works before you go to the next step!

f.) Change the code in `main` so that it draws castles in the pattern seen in the following figure:



4.) Verify that your documentation makes sense and that you've added documentation to each of your functions.

5.) Verify that your program works

6.) Upload your file to the Program 2 dropbox folder on D2L

3 Part III: Get Weird Revisited (30 points)

It may help you to sketch pictures out on scratch paper before writing any code. Some of the wilder drawings will get shown in class.

Draw Weird Shapes

In your text editor create a **new file** called "drawing.py". In the file:

1. Add the required documentation and add a `main` function.

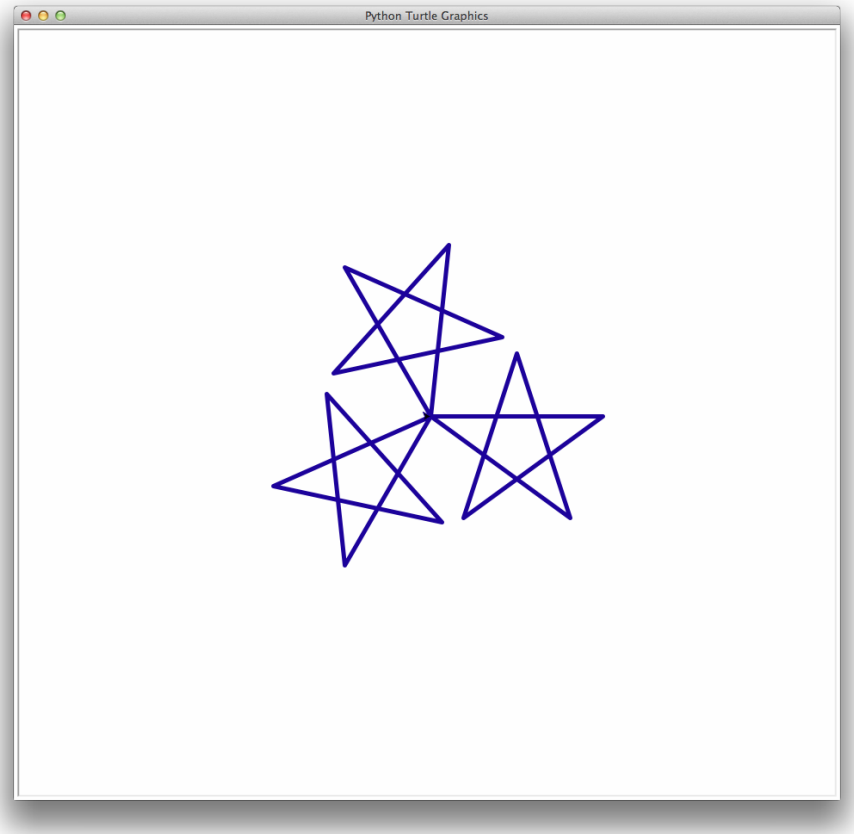
- After this homework we'll stop reminding you to add your documentation. We'll just assume you know to *always* add the required documentation.
- We'll also assume you know how to add the `main` function. If you forget, just look at "template.py".

2. Write a function called `shape` that draws a shape of your choice. The function should take 2 parameters. The first is for a turtle. The second is for a numeric value giving the size of shape to draw.

   - The shape *must* have some part that can be repeated inside of a loop. The function *must* use a `for` loop to draw the repeated part
     - It is acceptable to draw the same shape you drew in Part III of the first homework if it can be drawn using a `for` loop (i.e. there is something repeated in it)
     - Otherwise it must be a new shape. Don't use one of the shapes we've already written functions for in lecture, lab, or earlier in this assignment (or any simple variant of those shapes like another regular polygon). Be creative, draw something wacky and fun!

3. Write a function called `rotated_shapes` that takes four parameters. The first parameter is for a turtle. The second parameter is the size of shape to draw. The third parameter is the number of shapes to draw. The fourth parameter is the angle between each shape. The function draws the requested number of shapes, each of the requested size, and each rotated from the previous shape by the requested angle.

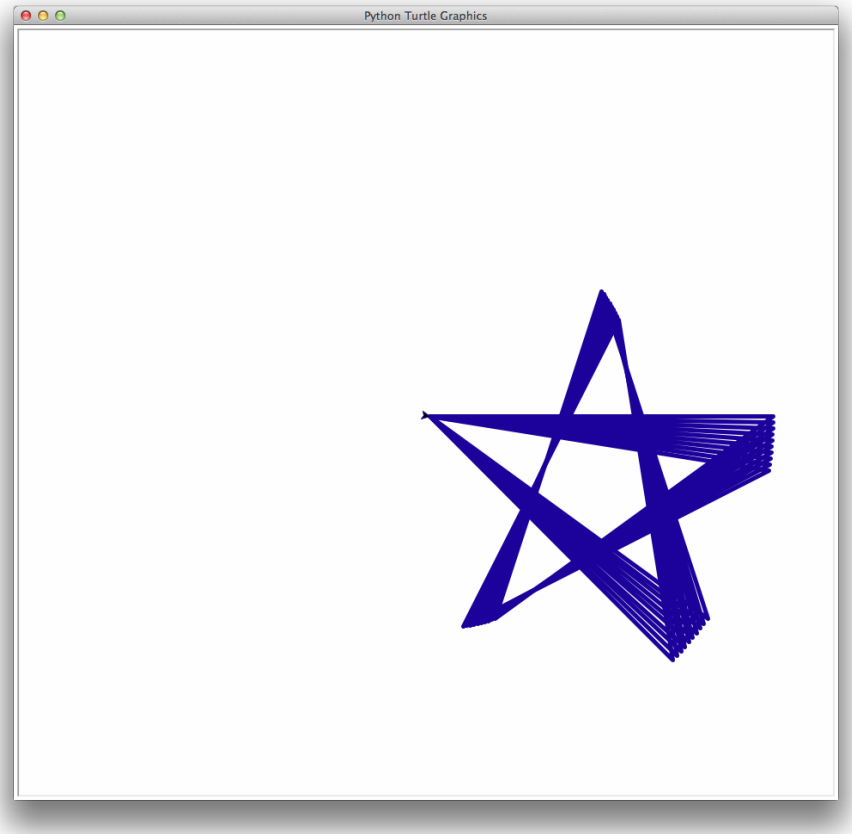
   - For example, using the star shape from hw1:
     - Calling the `rotated_shapes` function like this:
       `rotated_shapes(some_turtle, 200, 3, 120)`
       Produces 3 stars, each of size 200, and each rotated by 120 degrees from the previous star. See the following figure:



- NOTE: since your shape will be different your shapes will probably end up in different locations than those in the figure! Don't worry about that.
    - Calling the `rotated_shapes` function like this:

```
rotated_shapes(some_turtle, 400, 10, 1)
```

Produces 10 stars, each of size 400, and each rotated by 1 degree from the previous. See the following figure:

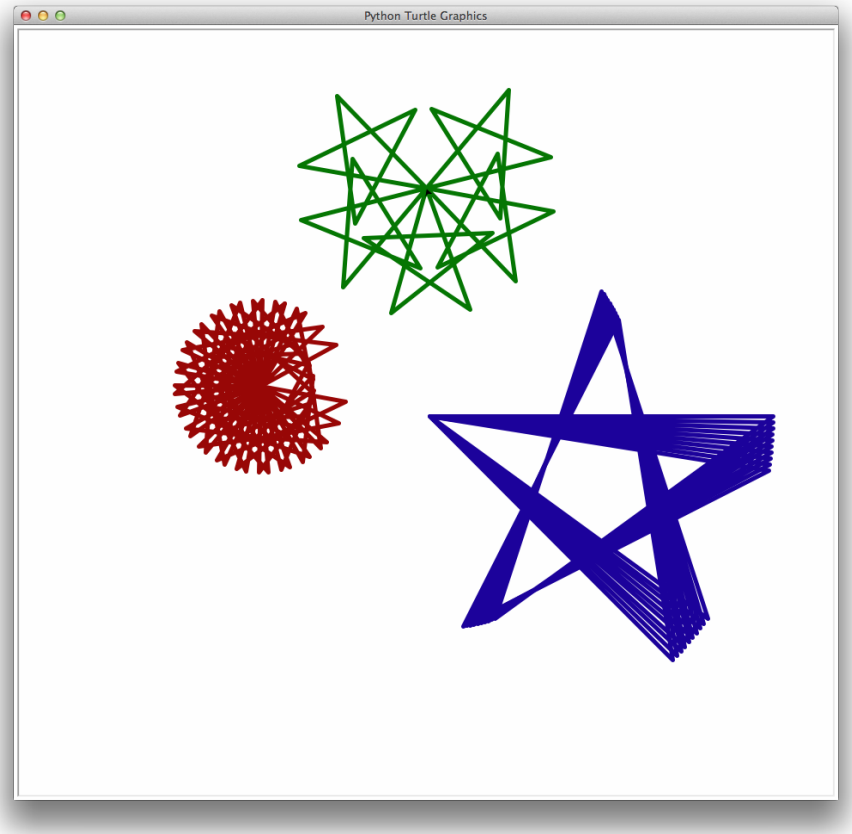


4. In the `main` function definition write code that will:

   a.) Set the turtle speed to 0

   b.) Use your `rotated_shapes` function to draw rotated shapes in at least three different locations

   - Use different arguments each time you call the `rotated_shapes` function
   - Change the `pencolor` before each call (or use a different turtle with a different color)
   - For example:



c.) Note:

- this will take some trial and error since you probably won't know where your turtle will end up facing after each call to `rotated_shapes`

5. Verify that your documentation makes sense and that you've added documentation to each of your functions.
6. Verify that your program works
7. Upload your file to the Program 2 dropbox folder on D2L