CSE 231 Spring 2013

Programming Project 10

Assignment Overview

This assignment will give you the opportunity to make up some classes and use them in turtle graphics to do some drawing.

This assignment is worth 60 points (6.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, April 8th , 2013.**

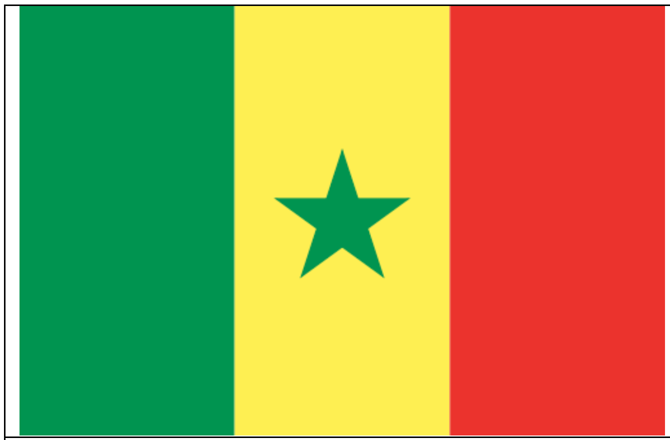
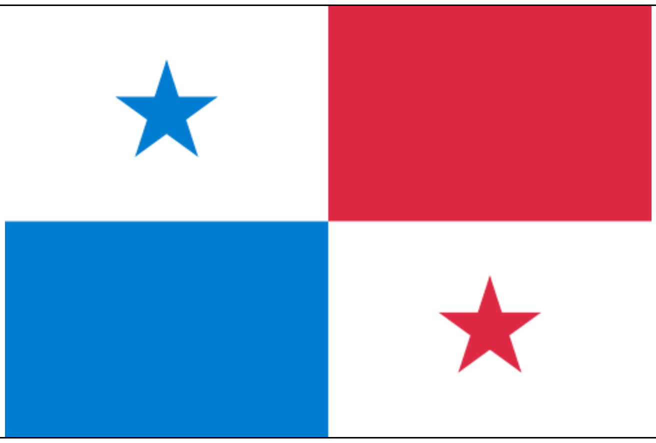
Background

We are going to make three: five-pointed stars, rectangles and flags that consist of various combinations of those stars and rectangles. We will read in, from text files, some specifications so that we may draw some flags

There are a lot of flags out there, many with similar elements. Take a look at:

http://commons.wikimedia.org/wiki/Flags_with_stars

Many of these flags can be drawn with our two constructs. Here are two we will draw:

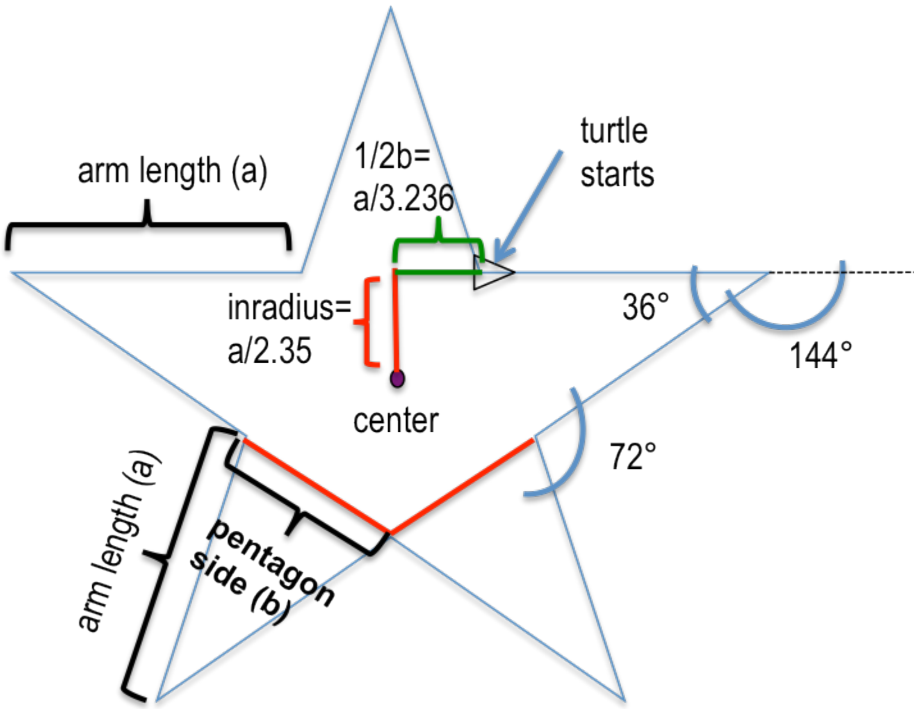
	
Flag of Senegal	Flag of Panama

Requirements

You are going to create the following classes and their methods

- `Star` class:
  - `__init__ (self, x, y, arm_length, color)`
  - `draw(self, turtle)`
  - `__str__(self)`
- `Rectangle` class:
  - `__init__ (self, x,y,width,height,color)`
  - `draw(self, turtle)`
  - `__str__(self)`
- `Flag` class:
  - `__init__(self, file_object)`
  - `draw(self, turtle)`
  - `__str__(self)`

Star Class
Here's the geometry.



$$\frac{a}{b} = \varphi = 1.618034$$

$$b = a/1.618034$$

$$\frac{b}{2} = \frac{a}{3.2361}$$

$$inradius = \frac{b}{2\tan(180/sides)} = \frac{a/1.618}{2\tan(36)} = a/2.3511$$

- `__init__ (self, x, y, arm_length, color)`
- `draw(self, turtle)`. It draws the star **centered** at the given x,y coordinates using the provided turtle.
- `__str__(self)` method returns string of the form: 'Star: x:int, y:int, arm:int, color:str' . For example:
    - `s = Star(100, 200, 50, 'green')`
    - `str(s)` → `'Star x:100, y:200, arm:50, color:green'`

Rectangle Class
Shouldn't need any geometry for rectangles.
- `__init__ (self, x,y,width,height,color)`
- `draw(self, turtle)` draws the rectangle **centered** at the given x,y coordinates using the provided turtle
- `__str__(self)` method returns string of the form: 'Rectangle: x:int, y:int, width:int, height:int, color:str'. For example
    - `r = Rectangle(100,200,300,300,'blue')`
    - `str(s)` → `'Rectangle x:100, y:200, width:300, height:300, color:blue'`

Flag Class
The flag class works based on a text file specification of a flag. The textfile has the following format:

single int, number of rectangle specifications that follow
each line is a rectangle specification of the form x,y,width,height,color using comma separated values
single int, number of stars to follow
each line is a star specification of the form x,y,arm_length,color using comma separated values

Below is the specification for the flag of Senegal (included in the project). 3 rectangles, 1 star

```
3
-200, 0, 200, 400, green
0,0,200,400, yellow
200,0,200,400, red
1
0,0,50, green
```

- `__init__(self, f_obj)`. The `f_obj` is an open text file that specifies a flag (as described)
  - it makes an instance for each rectangle line and stores it
  - it makes an instance for each star and stores it
- `draw(self, turtle)`. Draw all the rectangles and stars read in from the file specification using the turtle parameter
- `__str__(self)`: Returns a multiline string. For example:
  - `f = Flag(Senegal_file_obj)`
  - `str(f)` →
    ```
    "Rectangles
    x:-200, y:0, w:200, h:400, c:green
    x:0, y:0, w:200, h:400, c:yellow
    x:200, y:0, w:200, h:400, c:red
    Stars
    x:0, y:0, a:50, c:green"
    ```

Deliverables
You must use handin ***two things this time***:

- your `proj10.py`
- a flag specification called `myFlag.txt` Hard to draw flags (complicated flags, not like the ones I provided) will have potential to garner extra points!!!!!

Other good information

1. Two flag specification files are provided in the project directory. Take a look at them.
2. I provided a `main.py` for you to work with. You may copy it into your solution.
3. Pick a flag from the website provided above to make the specifications for your own flag
4. The `readline()` method of an opened `file_obj` is very useful here. It reads exactly one line at a time from the file. Since you know what the first line will be (an integer describing the number of rectangle lines to follow), you know what every line in the file will be and can read them appropriately using `readline()`.
5. It gets irritating to have to watch the turtle slowly draw. The setting `pen.speed('fastest')` makes it draw very quickly but you need to delay between flags to see the result. In the time module is a method `time.sleep(seconds)` where you can specify a delay in seconds. The `main.py` uses this.
6. If you get some weird errors with colors, ask yourself: did you strip the color specification when you read it in? `'green' != ' green '` (see the spaces, I tripped on that)
7.