Homework 4 – Robot Personality
Due: Tuesday, October 21st, by start of class

Scored out of 100 points
Files to submit: draw.py

Notes:
• Don't forget the required collaboration statement as detailed on the course syllabus as a comment at the top of your program.
• Be sure to appropriately comment your code.
• If you find a major error in this assignment, please let Prof. Summet know as soon as possible.

Part 1: Robot Personality

So you’re sitting there, looking at your Scribbler robot, and you find it to be, well, impersonal. You've seen much more interesting robots in movies! So let's draw a robot with some personality.

Hey! That looks much friendlier. And you can do it all with python and Myro! Myro contains a multitude of commands for programming your robot and making graphics. See the documentation here: http://wiki.roboteducation.org/Calico_Graphics Also, Chapter 8, pp 175-177 has a good reference.

You'll be using the graphics aspect of Myro, so you won't need your robot. BUT! You will need to add the import statement:

from Graphics import *

in addition to the usual “from Myro import *”. The link given above and Chapter 8 in your textbook will be rather helpful. Additionally, there is a quick “getting started” page at the end of this assignment.

Your mission is to draw a robot and add some decoration to it. See exercise #2 at the end of Chapter 8 for a good starting idea on how to (easily) draw a robot portrait in Python using Myro.
You need to create a program that initializes a graphics window, and draws a robot, and adds some personalization to it. The portrait should contain:

- At least 2 different colors
- A (somewhat) recognizable picture of a robot
- Two other creative aspects to your portrait. Examples might be:
  - A hat for your robot.
  - Eyes (or other facial features) on your robot.
  - Clothes for your robot
  - Your robot's name tattooed across his chest
- Text with your team name (If you don't currently have one, make one up.)

Try to keep things in a general alignment here (Sorry to all you budding Picassos out there), but the shapes can be whatever you want. Be creative!

**Part 2: Adding movement**

Ok, now you have a really nifty robot. Now you're going to animate parts of it. Animations are really just a sequence of images with parts drawn in slightly different places. Myro provides a “move” function which will allow you to move graphic shapes to a different (x,y) location. Animate your robot in at least two ways. For example, your robot could

- wave
- lift its hat
- nod its head
- other creative movements

You will be able to easily accomplish this using loops and/or variables which adjust a part's (x,y) location. You may want to utilize the wait(seconds) function to slow down the loop's execution so that the movements are noticeable to the human eye. The exact amount of time to wait between redrawing shapes will depend on how quickly you want your movements to occur and how far you want to move the shapes in a single step. Your movements must be long enough to be noticeable. In other words, it is not sufficient to move an element of your robot once. Additionally, your movements should be relatively smooth.

**Turning it in:**
Submit your file (draw.py) to Blackboard for HW4. Remember that only one member of your team needs to submit your work, but all team members' names should be included in the file. Don't forget your collaboration statement.

You can earn up to 6 points bonus [discretion of Dr. Summet] for extra creativity/general awesomeness.
Myro Graphics Primer

More specifics can be found on pgs. 175-177 of Ch. 8 in your text.

Remember to add the import statement:
```
from Graphics import *
```

To create a canvas to draw on, you can use the commands:
```
myCanvas = Window("title to appear on canvas", width, height)
```
This creates a canvas and assigns that canvas object to the variable myCanvas. You can then use that canvas variable and draw other objects on it.

To draw shapes, you have a couple of different options: Circles, Rectangles, and Polygons. Each of these rely on making Points. These points are objects in and of themselves. You can draw a Point on your canvas, but it will be so small you can't see it. You can make points by using the command: Point(x,y). You can use these points as anchor places for your shapes. Code to create two shapes is given below.

```
c = Circle(Point(150,50), 30)
r = Rectangle(Point(110,130), Point(150, 230))
```
You make a circle by calling the Circle method with two arguments: a Point and an integer. This creates a circle with the center located at the point (150,50). The circle has a radius of 30.

You make a circle by calling the Rectangle method with two arguments: two Points. This creates a circle with opposite corners at Point1 and at Point2

The coordinate (0,0) is the upper left hand corner of the window.

You can also make Lines, Ovals, Polygons, and Text. See pg. 176 of Chapter 8 in your text for the appropriate syntax.

For any of these shapes, you can do things like set their color. For example:
```
c.setFill(Color('red'))
```
would set color the circle you made above red.

In order to display any of these shapes, you must draw them on your canvas. To do this you would do something like:
```
c.draw(myCanvas)
```
This would draw the circle above on the canvas you created in the first step.

IMPORTANT: YOU MUST DRAW OBJECTS ON THE CANVAS FOR THEM TO BE VISIBLE! Creating the shapes alone will not cause them to be visible.

You can move objects by doing something like:
```
c.move(100, 100)
```
This would move your circle 100 pixels to the right and 100 pixels down from its current x,y location.
In other words, the two arguments are the deltas for x and y.