CSE 231 Spring 2009

Programming Project #10

Assignment Overview
We are again going to use some turtle graphics to draw some pictures, but we will use classes to created the basic shapes and assemble them. The project is worth 50 pts (5% of the grade) and is due before midnight, Monday April 13th .

The Problem
You are going to create at least 5 classes. Using instances of those 5 classes you are going to draw a face (a picture of a face). You will write a function that assembles the face.

Program Specifications:

1. 5 classes (at least). One can be a circle class (which is easy) and one can be a square/rectangle (also easy), but you must come up with 3 "interesting" classes as well (not simple turtle commands, but some construction of multiple commands to draw the class instance)
2. Each class will have an `__init__` , `__str__` and a `draw` method:
   a. the init method will take a Boolean fill argument (fill or not fill with color) and a color argument (the color of the figure)
   b. the draw method will take x and y coordinates arguments (where the figure is drawn)
   c. str is the conversion to a string. It returns a string and is used for printing in python.
   d. Other arguments may be required for your classes, this will be reflected in both your docstrings and the face function.
3. a function `face()` with takes no arguments but draws the required face.
4. All classes and methods **require** a docstring for a general description of the object/method.

Deliverables
Turn in proj10.py

1. Please be sure to use the specified file name, i.e. "proj10.py"
2. Save a copy of your file in your CS account disk space (H drive on CS computers).
3. Electronically submit your program.

Assignment Notes:
What counts as special?
The idea is to make classes for facial features (eyes, nose, smile, hair, eyebrows, whatever) that have to be drawn and are not just a circle or a square/rectangle. An ear, for example, is a kind of half-arc (but not exactly if you were going for accuracy), eyebrows would be two half-arcs, eyes might have an iris, a pupil. You get the idea. Not just a face consisting of simple, ugly geometic primitives, but something more interesting.

Some classes, once made, might need extra arguments. An ear for example, would face left or right depending on where it was added. You could make a tooth class that took different sizes and fit together in a mouth.

Using turtle graphics:
In order to use turtle graphics in python you must first import the turtle module. You can then use the help function in idle to find out what methods this module includes and what they do. Just type "import turtle" in the idle command window, hit enter, and then type help(turtle) and scroll up through the list and information.

Also, turtle graphics changed in Python 2.6.1, please make sure you only use the commands from 2.5.x. See http://www.python.org/doc/2.5.2/lib/module-turtle.html for details.

`turtle.up(),turtle.down():` Set the pen state to be up (not drawing) or down (drawing)

`turtle.right(degrees), turtle.left(degrees):` Turn the direction that the turtle is facing. The amount of turn is indicated in degrees.

`turtle.forward(distance), turtle.backward(distance):` Move the turtle forward or backward the amount of distance indicated. Depends on the direction the turtle is facing. Draws a line if the pen is down, not if the pen is up.

`turtle.goto(x,y):` The goto method moves the turtle to a specified point, drawing a line along the way if the pen is down, and not drawing if the pen is up. Note: The turtle always starts at the point (0,0). The goto method moves to the indicated x,y coordinates.

`turtle.color(r,g,b):` The color method sets the color that the pen will hold for all drawing until the color is changed. It takes three arguments, each a floating point number between 0.0-1.0. The first is the amount of red, the second, the amount of green and the third the amount of blue.

`turtle.cirle(radius):` draw a circle of the indicated radius. Note where the turtle starts its drawing. It starts at the bottom of the circle.

`turtle.write(string):` Write a string starting at the present turtle point.

`turtle.fill(flag):` Use the command turtle.fill(1) before you start drawing a figure. Draw the figure, then execute the command turtle.fill(0). The figure drawn between the two fill commands will be filled with the present color setting.