CS 150: Intro to OOP, Fall 2008
Programming Assignment 4

Due *via email* by 11:59pm on October 22, 2008

For this assignment, you must work individually. As usual, you are welcome to discuss *general* python syntax, but your design and coding should all be done individually. Please refer to the class syllabus or consult the instructor if you have any questions.

In the first programming assignment, you created an animation which included several animals. At that time, you relied on the use of the class, Layer, to manipulate your animal as a single coherent unit, rather than as a scattered collection of individual shapes. We saw several advantages in that representation of an animal, such as being able to easily move the animal in the scene.

That use of the class Layer was a step in the right direction, but this time we would like to go a step further, having you create and document a new class which represents your specific animal.

Background discussion:

For the sake of discussion, let's suppose that my first programming assignment generated an animation which included several monkeys. Presumably my code included some 30-40 lines of code specifically for modeling a monkey. I probably instantiated various shapes to represent the arms, legs and tail of the monkey, as well as instantiating a new Layer to represent the monkey as a whole. Then the body parts were added to the layer, and the layer was added to the canvas. Suppose that I were proud of my representation of a monkey and that I wanted to add more monkeys into my scene, or better yet, I wanted to allow other people to conveniently add my monkeys into their own animations. I might accomplish this by simply making the 30-40 lines of code which I had used available to others, verbatim, for inclusion in other places. But a much better design, in the spirit of object-oriented programming, would be to define a new class, say Monkey containing the necessary code. Then, others could use this new class with minimal effort, just as they had used the more primitive shapes. For example, an artist might simply specify:

```
chimp = Monkey()
paper.add(chimp)
chimp.move(80,120)
chimp.scratch()
```

This is our goal. Of course, there is no reason to reinvent the wheel; we do not wish to create our new class entirely from scratch. We have already seen that the concept of a Layer is a great model for representing a Monkey, and so we will use inheritance to define a Monkey as a subclass of Layer. Therefore, we might start out our class definition using syntax such as:

```
from cs1graphics import *

class Monkey(Layer):
    ...
```

In this way, any object from class Monkey inherits all of the instance variables and methods associated with the class Layer, such as move(), draw(), setDepth() and so on. Of course, if we do not add any additional code to the class definition for Monkey, then our class will be identical to that of a Layer. This assignment will require you to add new instance variables and behaviors as well as overriding existing behaviors.

Requirements:

There are three main sets of requirements to keep in mind while designing your class.

1. Create a new class, representing a type of animal, and define at least three new and distinct methods which are appropriate for customizing your chosen animal.

   In this assignment description, we have been using the discussion of a class Monkey purely for example. Your project should involve the development of a new class to represent your own choice of animal, most likely an animal from your first programming assignment (though you are free to change your mind).

   Please adhere strictly to the following minimum requirements for the development of such a class:

   - You will need to write a constructor which appropriately initializes a newly created animal. Presumably, this will involve the creation and placement of several underlying shapes, which are then added to the animal using the inherited method add().

     You may decide whether or not to allow additional parameters when calling the constructor which effect the initial settings for your animal.
   - Think carefully about the geometry from the perspective of the user. That is, if they execute, chimp.moveTo(100,100), hopefully the monkey is moved visually to that location with a meaningful reference point. That is, perhaps the bottom of the foot is moved there, or the nose, but it should not be that the monkey is played halfway across the screen. In similar respect, rotate and scale are based upon the notion of the reference point, so consider how you define the reference point of one of your instance (by default, it is at coordinate (0,0) of the Layer).
   - All monkeys are not alike! The point of such a class is not simply to make precise clones of an animal, but to allow reasonable variance as well. For example, I might allow a user to change the eye color of a monkey, the tail thickness, the positioning of the arms, the direction of the face. This is where the 3 distinct methods for customizing your animal come into play.

2. Properly document all aspects of your newly defined animal class.

   For someone else to know how to use your class, you must provide sufficient documentation. For this, we would like you to use docstrings, as described in earlier work. Specifically, please ensure that:

   - You provide a docstring to begin the class definition, which gives an overview of the class as a whole.
   - Any method which you create should begin with an appropriate docstring, giving an overview of the behavior, as well as explicit descriptions of any parameters or return values.
   - The constructor documentation should give sufficient explanation of the initial geometry so that a user could properly place on of these instances into a scene.

3. Include a *separate* program, called test.py, which tests your animal and demonstrates several aspects of its use. Your animation for this assignment does not need to correspond to the precise animation you created in the original Artist assignment. However, we would like your new animation to satisfy the following requirements:

- Demonstrate the use of each new method introduced in your class.
- Display at least four distinct animals from your new class, making sure that the properties vary among those animals.

Extra Credit: I'll know it when I see it.