Week 10 Practice Problems

These are pretty long problems, requiring the most code you've written for the practice problems so far. So I've only given you three. Start early!

#1. In the Real Estate Design Problem from Week 9, we represented each house as a dictionary with entries for the different pieces of data about a house. It is actually more natural to represent each house as an object (of class House) and have each of the pieces of data represented by attributes (i.e., a street number attribute, a street attribute, etc.).

Redo the Real Estate Design Problem in an object-oriented way. Start with the code you already have and

- create a `House` object
- write an `__init__` function that accepts one entry from the MLS_list and fills in the attributes appropriately
- edit the functions that we wrote to deal with this new way of representing the houses. For example, rather than setting up a dictionary (indexed by street name) whose values are the lists of dictionaries, you should set-up a dictionary whose values are lists of `House` objects.
- Bonus question: research the `__str__` function and then write one for the House object to (partially) replace the `print_house()` function we wrote.

#2. One method of estimating **π** is to use Monte Carlo simulation as follows.

Place a circle with radius, r, inside a square with sides of length $2r$. The area of the circle is πr^2 and the area of the square is $(2r)^2$. Therefore, the ratio of the area of the circle to that of the square is **π** /4. If you randomly choose n points inside the square, approximately $n\pi/4$ of them will be inside the circle.

Write a program that estimates **π** by randomly picking points inside the square and testing if they are inside the circle. Let m be the number of points inside the circle. **π** can then be estimated as $4m/n$.

To do this, you should create an object-oriented program with three classes:

- Point
- Square (implemented using Point)
- Circle (also implemented using Point – how?)

The first two classes are discussed in lecture but you will have to figure out how to design a Circle.

To implement the estimate, Square should have a method called `generate_random_point` that returns a random Point inside the Square. Similarly Circle should have a method called `check_inside` which takes a Point and returns True/False depending on whether the point is inside the Circle or not.

#3. Extend your code for Q2 by using a Turtle class to illustrate the algorithm. Add a `draw` method to each of the classes in Q2. The method should take a Turtle as an argument and use the Turtle to draw itself. Your program should first draw the square and the circle and then every point as it is generated. (Hint: see the dot() method of the Turtle class).

Hint #1: You've seen a lot of turtle code. Go back to it and see if it can help you. In particular, the Wheel of Fortune code from Week 6 Lecture 2 might be handy to give you some ideas.

Hint #2: You will need to do some work to figure out the parameters and meaning of the Turtle methods. In particular the `circle` method is a bit tricky.

Hint #3: The animation with the turtle slows things down a lot – even at the turtle's top speed. You may want to ask the user if they want the animation turned on or not and do the right thing depending on their answer.

Bonus #1: Color the points inside the circle differently from the points outside the circle.

Bonus #2: Every time you generate a point, update the estimate of **π** and have the Turtle display it.