1 Administrative Topics

- Return the quiz
- Make sure everyone got grade reports from Bruce and Stephanie
- Change in office hours for Stephanie: M: 1:30-4, T: 2-5, F: 2-4

2 Project 7

Next week, we will eliminate all turtle calls from the scene code. So, if you want to get ready for next week, put as much turtle code into the interpreter as possible, e.g. make init, goto, orient, and place functions. I used my interpreter.init function to turn the tracer off, but you could use it to create a window of a specific size, or speed up the tracer. I use place to both orient my turtle and to place it in a new position.

The design of this project is that code in lsystem.py does all the L-system manipulation (reading one in from a file, adding a rules, etc.) and string-building. The code in interpreter.py handles the turtle commands (e.g. the code that determines how something is drawn). The code in scene.py handles the scene construction (where and what is drawn).

Here is a basic set of code for scene1. The idea is to use lsystems and interpreter functions as efficiently as possible. Mostly, this means that we cut out some of the file i/o.
Note that I employ two of the interpreter functions I talked about above.

One of the extensions is to add an L-system-based shape to a scene from project 2 or 3. You can also use shapes from project 2 and 3 at will.

To add leaves and berries, you need to add new symbols to the L-systems that generate those trees. Remember to put leaves at the ends of branches (which means you should place them before ] characters).

You may also want to control the color of a drawing. I did that by adding a color parameter to drawString, but you may want to add symbols to the L-systems to control some aspect of the color.

3 Classes

On Friday, we talked about classes. Classes are used to describe new object types. The new object type we designed on Friday was called Student, and it stored and reported a year of graduation and a name. It has two accessor methods, getName and getYear.

Today, we add a mutator method setYear so that students can update their graduation gate if necessary.

We also add a method (getGradeTitle in section A and getSeniority in section B) that, based on the current year, identifies a student as a first year, sophomore,
junior, or senior.

Writing the code for setYear is straight forward – it looks much like the initialization code. We simply need to update the year entry in the object’s symbol table.

Writing getSeniority requires us to determine what the current year is. In Section A, we spent some time scouring the on-line documentation and found a solution, but we are not entirely confident it is the best solution. We then wrote the code under the assumption that the month of the year doesn’t matter.

Here is the complete class definition code from section B. The .py files are on the website as well.

In the future, we need to ensure the code determining the current year is written correctly. Also, we need to take the month of the current year into account.