Final Project

Dr. Victor Norman
CS 106
Fall 2014

The final project in CS106 is an opportunity for you to showcase what you have learned in this class, and begin applying your new-found knowledge and abilities to a problem that interests you personally.

The final project may be an individual or a small group project, although I hope that in either case you consult with others in the class and with professors (if possible) for help in design, implementation, and debugging.

I encourage you to investigate and (hopefully) use existing python packages already available on the net -- packages like matplotlib, pymaps, scipy, pygame, sympy, openpyxl, vpython, etc.

I see final projects as falling into two categories: simulations and data analysis. For simulations, it is often best if you can create a class representing each of the types of "actors" in your simulation. Then, if you create multiple instances of these classes, how do they interact with each other? (Read the "hanging chain" case study in chapter 5 in our book. One could model a link in the chain as a single object, and then neighbor links would affect/interact with each other.)

Data analysis projects take data and manipulate it, either to analyze its properties, or to allow users to visualize it in new ways. To do this kind of project, you need data (that should be obvious, but I thought I'd point it out anyway). I would much prefer that you don't pick a project where you have to collect, gather, or fabricate your own data. You don't need extra work to do.

Your project must be interactive with the user. You may either ask the user to enter data via raw_input(), or you may take values on the command line (using OptionParser). You must implement a -v/verbose flag to help you debug your project as you write it.

Grading

I will grade your project as follows:
• 15%: design
• 15%: documentation (good comments, good docstrings)
• 50%: working code
• 15%: unit tests implemented and thorough. If your project does not use classes, then make sure you have a good amount of print statements (couched in if verbose: clauses) so that a user can turn on the verbose flag and see much more about what the code is doing as it runs.
• 5%: complexity/significance of the project

Note that it is better to choose a final project that is not overly complex and get it right than it is to choose a project that is too complex and not finish it. I recommend that you find a project that you can implement in stages, so that at multiple points you can have a "finished" project, and then decide if you want to or have time to proceed to the next stage. If you choose this route, it would be best to document these stages in your design document.

Timeline

There will be three deliverables for this project, due according to the following schedule:

1. Project Design: You must submit to me a document outlining what your project will do. See below for more details.
2. Project Showcase: after the written portion of our final exam, those who have presentable work will present their work to the class. You should be able to run your code at this time. If you have concrete results to show, please do so. You may want to have a web page or PowerPoint slides to describe what your project does.
3. Project Submission: You submit your final project code by midnight the day of the final (Thurs., Dec. 11, 9 am). If you use data for your project, you must submit the data files as well.

I will combine the points from these deliverables to compute your score for the final project/presentation.

Project Design

This document must include:

• A high-level description of the project,
• The main algorithm of your project, as pseudo-code. E.g., for a hangman game, you might write:

1. Print introductory message
2. Get words from file into a list of words.
3. Main loop:
   3.1. Ask user for how many letters in their word
3.2. Initialize variables to hold the number of guesses made already and letters guessed already.
3.3. while there are still guesses available:
  3.3.1. get guess from the user, repeatedly until they give a letter they haven’t guessed yet.
   etc...

- A description of each class you intend to implement, including the class variables and methods. You should note which class variables are mutable and which are immutable.
- What you expect the input and output to look like: if it is text, then a quick sample of what it will look like. If it is graphical, a description of it.
- The ways a user can alter the run of the program by changing input.

Note that the more work you do on your design, the less time you’ll have to spend writing the code (because you’ll implement more of it correctly the first time). It is a proven fact. **Due Nov. 26, midnight.** Note: don’t leave this until Nov. 25, please. The file should be a Word document, PDF document, or text document. It must not be python code.

**Ways students lose points**

- Start way too late...
- poor documentation: no comments.
- poor software organization: repeated code that could easily be in functions. Mega-giant main loops, e.g. Or, functions defined within a mega-giant main loop.
- The main loop should ideally be quite small – perhaps 20 to 40 lines -- with calls out to functions or class methods to do the work
- magic numbers in their code... Repeated checks against a value that should be put as a constant.
- Methods in classes that use raw_input(): makes it completely non-portable.

**Ideas**

See our website, under Resources, for ideas.