**CS 4500, Spring 2018**

*HW4: Visualization a 5 X 5 redistricting scheme*

If you have questions about HW4, ask the instructor. You may look for general programming advice from websites, BUT YOU MUST LIST ALL THE URLs THAT YOU VISIT FOR THAT ADVICE. Being a quality professional includes always giving credit where credit is due.

The program you are will submit will be run using the Python 3.6.4 interpreter. You can download Python, and get much information about it, at <https://www.python.org/>. They have loads of documentation and aids at <https://www.python.org/doc/> .

If you use any of the code from any of these sources, or from any OTHER source, you MUST include a citation to the URL in your opening comments of your program, and you must indicate in an internal comment where you used the code. This may be unusual for you, but it is required in CS 4500, for reasons that should become clearer later in the semester. For now, just do it.

The structure of all software you write for this course should follow the following “recipe.”

1. Opening comment: This should contain your name, the date, the title of our course, version of your software, description of the purpose of the software, any external files involved with the software, and information about any sources you used while writing the software. (The sources could include websites, as described above, people who helped you, or documents important to the development.)
2. Internal comments: include paragraphing comments for major chunks of your code. Include extra comments for anything a reader might find to be “tricky” or non-intuitive. For any functions, describe any global variables (including files) used in the function. Describe in comments what the function is supposed to do, what any parameters are for, and what is returned (including nothing if that’s appropriate).
3. Use good habits for writing easy-to-read Python code. Be careful with variable names, white space, and indentation. (Indentation is especially important in Python.) Using functions to decompose your problem solving.

A, B, and C will be graded for all of your assignments. You are likely to be annoyed when you lose points for this. Please remember, I warned you. You may be surprised how much I value good documentation and structure. Let me explain.

Most programmers who do this work for a living eventually get things to run; that’s not really separates good software professionals from mediocre software professionals. The best software professionals are efficient problem solvers, good technical writers, good teammates, and decent people with integrity. Good documentation is, in my humble opinion, the mark of a quality software professional. I will insist on good documentation for everything you submit for CS 4500. Although my opinion may be humble, it will be part (though not all) of the grading process in this course.

**Grading Rubric:**

This assignment is worth 50 points. Here are the questions I will ask while assigning you a grade:

1. Was your program turned in before the deadline?
2. Did your program follow the specification EXACTLY?
3. Does your program work as specified on the tests to which I subject it?
4. Did you include sufficient and high quality documentation in your source code?
5. Are all sources used in developing the code explicitly cited in the program?

For HW #4, you will be working on redistricting schemes having to do with the following set of 25 voters:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P | G | G | G | G |
| G | P | P | P | G |
| G | P | G | G | G |
| G | G | G | P | P |
| P | G | P | G | P |

There are 15 green voters, and 10 purple voters, a 60-40 split. We assume that in each election, there is one green candidate and one purple candidate in each district, and that green voters always vote for the green candidate, and purple voters always vote for the purple candidate. The 25 voters are to be split into 5 districts with 5 voters in each district.

When you break a set (in our case, 25 voters) into subsets (in our case, 5 districts) such that each item in the original set appears once and only once in one of the districts, it is called a “partition” of the original set. We will restrict our attention to districts that have an equal number of voters in each (in our case, 5 voters in each district), and all 5 districts must be contiguous. A district is contiguous when all 5 cells in a district touch at least one other cell in that district, where two cells can touch at their sides or corner to corner. (In other words, each interior cell has 8 touchable neighbors.)

In HW2, you randomly generated districts. In HW3, you added the constraint of contiguous districts. For HW4, you will take contiguous districts that you generate and display them in some visually helpful way on the screen, and in a textual way in a textfile. The idea is that a person running your HW4 program will be better informed about how districts look (in a two dimensional space), and how each of the contiguous redistricting schemes affects the results of the election. At the end of the textfile, statistics about the redistrictings depicted during the execution should summarize the results.