SI 507 Fall 2017 | Final Project

Deadline: December 14, 2017 11:59 PM

No late days for the final project — grades are due 72 hours after the final, so we've got to start grading fast!

To Submit

- **A link to a GitHub repository, which should contain the following:**
  - A **README.md** that, as described below, tells us exactly what to do to understand and run your code. We shouldn't have to ask you ANY questions after reading this. See below for more on the README expectations.

.

  - Any code files required to run your project, which contain the programming requirements listed below. **The main file we should run should be called SI507F17_finalproject.py**, but if you include other files and import them into that one, that is fine! All code files included must run — we will not grade code that does not run. (We should be able to run python SI507F17_finalproject.py — with command line arguments, or any other additional instructions you explain in your readme — and get your final result.)

.

  - A test suite file, called **SI507F17_finalproject_tests.py** — it should import any code from other files that is being tested! (We should be able to run python SI507F17_finalproject_tests.py and see the test results in the console.)

.

  - A requirements.txt file from your virtual environment.

.

  - Any .py or other file templates that we have to fill in. e.g. secret_data.py — give us a version of it without your key and secret, and tell us how to fill in our own, if we need to! (You should include the entire explanation of what data we need to put in such a file — it's not enough to say "copy what we did in class", but it is OK to c&p from instructions we gave you if they are indeed the same!)

.

- **An example of the final result of the project, submitted on Canvas, so we can see what it should look like when we've run it.** This could be a literal file that the project generates, or a screenshot of what it should look like when it's run.

    - (You could also consider including an image like this in the README.md to show users what the output should be like! But doing so is not required.)

Requirements

Below are all the requirements for the 507 F17 Final Project.

- **Create a git repository on your computer to build your project, and push it to a GitHub repository of your own.**
    - If you have others make suggestions on your project — make sure you are following the academic honesty rules, of course — they can do so via GitHub tools!

.

- **Get data from least one complex source, in a way you have learned this semester, with caching.**
    - Scraping with BeautifulSoup from any website (choose carefully: is the data you want there? is the HTML markup good/clear enough to make scraping reasonable?)
    - OR getting data from an API *that uses OAuth authentication*.
    - You may use additional data sources as well, but you must use at least one of these ways to get data. Of course, you may use both if you want, but be careful that you're scoping your project properly so you can get it done in time.
        - You may use a client library if you want to fulfill the data gathering requirement — but make sure you are confident it will work for you before you commit to it, and make *sure*, when you submit your project, you've been clear to us in the README about what we need to do to run your code, and that you've included all the libraries necessary in your requirements.txt file!
    - **You must have a caching system set up in your code, and it must be at least as complex as e.g. the one in the textbook**. It's OK to borrow heavily (but should cite in your README code you borrow) from the textbook, code from class, etc, just make sure it all works *for your code in your project* (including functions 'for caching' that don't work/don't cache anything won't earn points).

.

- **Your code should include at least 1 class definition.**
    - The class should include a __repr__ method and a __contains__ method.
    - You must create at least one instance of the class you define and actually use it in the process of your program. (Defining a class / creating an instance you never actually use will not count.)

- You can use this class definition for anything useful in your program: using it to process the data you gather and storing them in database tables, or using it to process data that results from querying your database… anything you want.
- You may want to consider e.g. the code you've seen in section — in section 10, and in others. (Of course, you should *not* copy any class definition code from class. Your class definition should be your own. If you are using similar data, you still need to make your definition very different from what we've done in class.)

.

- **Store data that you get in at least 2 different database tables**.
    - You should, in your script, set up at least 2 database tables in a database, and write code to store data in them.
    - Each database table should have a least 2 columns and end up containing at least 4 rows of data.
    - The tables you create should have at least 1 relationship existing between them, e.g. so you could make a reasonable JOIN query on the two tables.
    - Of course, you can have more than two tables if you want, and you may make the database structure more complex.

.

- **Include a full test suite for your project.**
    - **You must have at least 15 test methods and at least 2 subclasses of unittest.TestCase, but even more important is that you have *good test coverage* of the project.** We should be able to e.g. get instructions from you and your test suite, write code ourselves, and basically use the test suite and your README to judge whether we have accurately re-created your project (i.e. once all the tests pass, we should be pretty close to having done exactly what you did in Python).
    - You should use at least one setUp method (e.g. if you need to create instances to test, open files to test…), and a tearDown method if it is useful. Given the requirements of the project, I'm sure all of you will need to test instances and open files!

.

- **Include a clear README.md document**
    - The README should explain/include:
        - Everything a user needs to know about how to use the code included (e.g.
            - what precisely to type at a command prompt to run the code,
            - what name of database they must create for the code to work,
            - that they should pip install everything to requirements.txt,
            - what version of Python the project expects (e.g. Python 3),
            - if they need to fill in their own **key** and **secret** in a sample secret_data.py file in order to run the code, and what URLs they

should go to to find the right key and secret…any information they might need)

        - Basically what happens when the code runs (e.g. what classes are defined or what a couple major functions' input / output will be, what database tables are created and used…)
        - What the user should expect after the code runs (what should they see, what do any numbers they see represent, what will get created at the end — a file? a chart? — and how they can view it, etc)
        - Links to any resources used (e.g. client libraries, API documentation) and citations of any code borrowed from elsewhere
        - Also include in the readme a list of everyone with whom you worked on any of the project or talked to about it in depth. Doing that is OK: sourcing help from others is great! But **no two projects should be alike — this should be your *own* work.**
    - The README should be clearly organized and written in an outline format (*not* a paragraph format like an essay)

.

- **Create a Python Virtual Environment and corresponding requirements.txt file for the project**
    - You should include the requirements.txt in your GitHub repository, and its existence and reasonableness is how we will count the virtual environment
    - You should *not* commit your virtualenv files to GitHub. Remember: You can use a .gitignore file to easily keep from doing so!

.

- **Your project should result in some visual representation of your data that is clear.** How you do so is up to you, but here are some suggestions, and caveats:
        - **Note:** You do not have to come to any statistical conclusions or anything — this is not a statistics course! "This is kind of interesting to me, see it here" is a fine and expected result. We'll continue discussing possibilities and examples in class.

.

    - **It must be viewable by other people, and it must present information in a clear way** (e.g. so if I looked at only the result, I would have a pretty good idea about what your project was about, data- and results-wise, even if I didn't know anything about Python or code)

.

    - You could learn and use the library **Plotly**, which has a pretty clear Python API that comes highly recommended from others! It creates nice charts and graphs. (If you choose it, you should make sure it will work for what you want to do.) https://plot.ly/python/

  - You could, if you know or want to learn some HTML, **use Python to create an HTML document** that can be viewed in any web browser.

  - You could design and create **a very nicely organized, clear Excel sheet/workbook** that your code produces! (Careful: this should be well-designed and clear if you go this route, so I should be able to read it and really understand it, and it should have complexity. *We're looking for something of a higher level of complexity than a SI 506 project.*)

- **Some more challenging/time consuming options:**
  - You could use **Flask**, if you want to continue building on your knowledge (but be *really* careful that you feel ready to do this; web programming is a very complex topic we'll barely touch here 🙂)
  - You could learn and use a more complex object-oriented visual framework like **PyGame**
  - You could use **Matplotlib** or another library you find and come up with an image/PDF result (warning: this can be difficult/frustrating to learn sometimes)
  - Something else!

Note

  - You may build on work from a previous project, but work that you have done and submitted for a previous project, especially not work that came from our instructions, will not count to fulfill these requirements.
    - e.g. the work you did in Project 3 could be included, but scraping data from NPS.gov will not count for your data requirement, and the NationalSite class definition will not count for the class definition requirement. You'd have to add more!
    - As always, the project you submit should be *your own work*. You should type 100% of the requirement-fulfilling code you turn in, and your project should be somewhat unlike anyone else's, because it's *yours*. If you find you're seeking too much help or giving too much help, please (a) pause and back off for a bit and (b) talk to an instructor! However — also as always — seeking help and question answers within these boundaries, and answering

questions (without providing full code for a whole section of a project, for instance) on Piazza and in office hours sessions, from one another, is encouraged!