SI 506 – Programming I, Fall 2016
Final Project + Plan (4000 points total)

Important dates & deadlines:

Final Project Plan due on Canvas: Before class Monday 11/28 (400 points)

Final Project Plan discussion: Section, Week 9 (11/16-17)
Final Project Plan review/feedback: TBA, ASAP post-11/28 (and in office hours, FB group, etc)
Coding party for project work: 12/11, time TBA

Final Project due: Tuesday 12/13 at 11:59 PM (last day of classes)

Point distribution:
Final Project plan: 400 points
Final Project: 3600 points
Graded with a rubric: you'll need to fulfill all listed requirements to get all the points.

Your final project in this class is building an "API mashup" – a Python program you can run on a computer that gets data from a couple different REST APIs, performs some computation or analysis on parts of that data so that a human can learn something new or find something interesting, and results in output that is human-readable/human-usable.

We suggest you think about concepts, ideas, data you're interested in, and consider making a simple version of something that deals with one/a couple of those things. You'll see and hear about examples in section and lecture in upcoming days, but the complexity of the final product should be approximately the complexity of ~2 - 3 combined problem sets (something along the lines of the Flickr problem set integrated with data from another service, for example…), and the final submission should include a clear explanation of what you have done and why you have done it.

Below is a list of the requirements for the final project PLAN and for the final project itself. Before you start working on the project plan, **you should read through this whole document and consider what you'll need to have for the final project.**

Final Project **Plan Requirements**

For your final project *plan*, which is 400 points (of the 4000 total for your final project), you need to SUBMIT:

- a document to Canvas (**Final Project Plan** assignment > **506_project_plan.txt**) that contains answers to the following questions
- a **.py** file called **506_project_data.py** that contains the code to get API data, described below
- 1 text file (or 2) that contains cached sample data from the APIs you plan to use

Note that these don't have to be your *final answers* or your absolute final plan. You are welcome to change this plan later, without approval from staff – but it's intended to provide you with a solid set of guidelines, a solid start, and a bunch of support with which to plan out your project, so we advise you to try and stick with it.

FINAL PROJECT PLAN QUESTIONS

- **What's your overall project plan, in 1-3 sentences?** What data will you get, what will you do with it, what will you analyze, what will the output be, what can a person get out of running it?
- **Why are you choosing to do this project?** How would you or someone else run this program you're planning to write?

- **What classes are you planning to define? For each class, answer the following:**
    - What will the class be called? (e.g. Post, Photo, Song…)
    - What will one instance of that class represent – how would you describe it in English?
    - What will the class constructor require as input?
        - (A dictionary representing a song? A dictionary representing a photo? Where does that dictionary come from? Or a set of values? What kind of data will it require to create an instance?)
    - What are 3 instance variables the class might have?
    - What are a couple methods the class might have? What will they do? Why?
- *If you plan to define functions that are not methods inside class definitions,* **What functions are you planning to define**? What will they take as input? What will they return?
- **What sequence of data might you be sorting?** By what attribute will you sort that sequence? Why?
- **Which APIs are you planning to use? Provide links to the documentation for each.**
- **What authentication does each API require?** How did you get access to it? Is it working as expected? What parameters do you need to include in your request?
- **What might you need to test in this project?**
- **What are your major concerns, if any?**

- **Write some code to get data from each API and save that data in a file (cache it)**. It doesn't have to be exactly the data you will use in your program, and only need be a few lines (make a request, load the data into a Python object…) but show us (and yourself!) that it's possible. **Then, write some code to extract at least one element from the data you retrieve from each API.** (For instance, if you want to examine Facebook comments, write code to extract a Facebook comment.) Provide comments in your code that explain what it's doing – what data are you getting? Why are you using these parameters for this API? **Submit that code and the data file(s) in addition to the answers to these questions.**

- **Tell us what we need to know to run the code you've submitted.**
    - Let us know here what we need to do in order to run your code. We don't have time to figure it out on our own for everyone's!
    - What data do(es) the file(s) you've submitted contain (describe it in English in 1-2 sentences)?

Final Project Requirements

SUBMIT:

- README.txt file that follows the README_TEMPLATE.txt template
- Python program file to run
- Any additional files necessary to run your program
- Sample output from running your program (an example text file or spreadsheet your program generates, a couple demonstrative screenshots, a screencast of it running… whatever makes sense)

Mechanics of your program (2200 points)

- *You must get data from at least two REST APIs.*
    - At least one of those APIs must *not* be one of:
        - The FAA
        - iTunes search
        - Flickr
    - Though you may use any of those (or all of them, if you really come up with a reason to…), you will also need another that is not among that list in order to completely fulfill this requirement.
- *You must cache the data you get from the REST APIs.*
    - We should be able to run your program successfully without access to the Internet, depending on cached data files.
    - You may include an option to run the program with live data, but you must also include sample data in some form.
- *You must define at least 2 Python classes.*
    - Each class must have at least 3 instance variables that are dependent upon input to the class constructor (so if you define a class `Student` that always has `self.university == "Michigan"` in its constructor, that would not count as one of the 3, though it is certainly allowed)
    - Each class must have an `__init__` method and 2 other methods. (The two methods may include `__str__` if it makes sense for your program!)
- *You must include at least one sort of a sequence of data that requires use of a key parameter in the sorted() function (or the .sort list method)*
    - For example, sorting a list of integers like `sorted(lst_ints,key=lambda x:x)` does not count, because it is functionally the same as `sorted(lst_ints)`
- *You must include at least one advanced accumulation*
    - An invocation of map, filter (or reduce) OR
    - A list comprehension
- *You must define a total of at least 6 functions.*
    - This 6 may include methods in your classes ***besides the __init__ constructor methods in class definitions*****, which do not count for this requirement**
    - Each must include some form of computation/change, even if it is simple (e.g.
    - `def get_value(self):`
    - `    return self.name`
    - does not count.
- *You must include at least 8 Unit Tests that test whether your program behaves correctly.*
    - We suggest that you be careful to organize these clearly
    - We also suggest that after some initial poking around at code, you write your unit tests first! Pretend like you're writing yourself a complex problem set and then solving it.

Using your code (1000 points)

- **Your code must run.** Note:
    - Just like problem sets, it should run without errors! Try/except blocks and conditional statements can be very helpful here.
    - If you give instructions that restrict user input (for example, "enter a string to search for"), we will abide by those instructions when we grade! But if your program is supposed to gather info from a Facebook feed and analyze it along with some other data, it should not break if it runs on your FB feed, my FB feed, your friend's FB feed, or the group feed, etc.
    - Your tests should depend upon cached data, and you should include the cached data file(s) necessary in your final project submission.

- All of your code must run **non-trivially**, which means that it must have a reason to be included in your final program. Requirements include accumulation and sorting, for instance, so you should make your project plan expecting those things to matter for the analysis you do. If you have a sequence of data sorted that makes no difference to your analysis or your output, then it was trivial *to the project*, even if the work to complete it was challenging! If you have a test that could never fail, that is a trivial test because it does not really tell you anything about whether your program is working!

- You must make use of all the mechanics elements in your program.
    - You must create and use at least 1 instance of each class you define, non-trivially
    - You must invoke (or potentially invoke, if your program is interactive) each of the methods of your classes, non-trivially
    - You must invoke (or potentially invoke, if your program is interactive) each of the functions you define, non-trivially
    - Your non-trivial sorting must be involved in your analysis (there should be a meaningful reason you are sorting data in some way)
    - All of your tests must run

Explaining your code (400 points)

- You must provide a README along with your code that follows our provided template, including listing all of the installation requirements and all of the included files.
- Your README should include a clear explanation of the motivation for and the result of your program.
- You must include in your submission *every file necessary to run your program*.
- Running your program must generate human-readable output. This could be
    - A .CSV file/spreadsheet
    - A text file
    - Easily-readable interactive command line output
    - Something else? (Check with staff.)

~Rules~

You certainly can:

- Use any other Python libraries or concepts we have not learned about, if you would like
- Use files that you depend upon and open aside from cached data, e.g. a list of words or a book text from Project Gutenberg, data you download in a .CSV format, etc.
- Include code that you wrote for other assignments/problem sets during this course and improve upon it/add more to it
- Include code that we provided to you on Canvas
- Talk to others about your project

- Openly license your project and post it on GitHub or any other website if you choose to…

You may not:

- Use code from any previous problem set to count for project requirements. For example, if you use the Post() class for Facebook posts in your project, you can certainly include it, but none of the methods defined on the problem set will count for project requirements. (If you add several more complex methods to the Post class, those would count for requirements, but e.g. `negative()` and `positive()` and `emo_score()` would not.)
- Work on the *same* project with someone else. You're more than welcome to help and support each other, but these are individual projects to turn in.
- Use external sources without crediting them. If you use code from an open source project or code that a friend wrote, (a) it will not count for requirements but you may include it with their permission/license, and (b) you must credit them in your README (see README template: there is a place where you can state any help/contributions).