Final Project - SI 206

SI 206 Winter 2018

Milestones

There are 3 milestones that need to be turned in.

- Project Proposal, due March 25
- Project Checkpoint, due April 11
- Final Project Demo and Repository Link Submission, due April 20

Project Overview

The goal of the final project is for you to showcase what you've learned in 507 regarding:

- Accessing data via web APIs, including those that require authentication
- Accessing data via scraping
- Accessing data efficiently and responsibly using caching
- Using a database to store and access relational data
- Use basic python data structures and operations to analyze and process data in "interesting" ways
- Use unit tests to verify that data access, storage, and processing works as designed
- Use a presentation tool or framework to present data to a user
- Support basic interactivity by allowing a user to choose among different data presentation options

Here are a couple of examples that would be reasonable final projects:

- A program that lets a user choose a city and see the average ratings for different restaurant types (e.g., bar, breakfast, Indian, Mediterranean) from Google, Yelp, and OpenTable as plotly bar or scatter charts.
- A program that aggregates crime data from https://spotcrime.com/mi/ann+arbor/daily and allows a user to select one or more crime type to see a graph of crime frequency by month, either for a single year comparing across several years. Data is displayed using HTML tables within a Flask App.

Project Components

There are several components that your project must contain. Each of these are detailed in this section.

Data Sources

You must select data sources that, combined together, give you a “challenge score” of at least 8. Additionally, you must use *either* a Web API that requires authorization *or* a website where you crawl and scrape multiple pages as *one* of your data sources (these options are marked with ✣ below). Here’s how the scoring works:

Data Source	Example	Challenge Score***
Web API you’ve used before	Twitter, iTunes, NY Times, Facebook*	2
Web API you haven’t used before that requires no authorization	Wikipedia, Google Books	3
Web API you haven’t used before that requires API key or HTTP Basic authorization ✣	Yelp Fusion, Open Movie Database	4
Web API you haven’t used before that requires OAuth ✣	Open Table, Reddit, many more	6
Scraping a page/site you’ve worked with before**	nps.gov, si.umich.edu	1
Scraping a new single page**	So many!	4
Crawling [and scraping] multiple pages in a site you haven’t used before ✣	So many!	8
CSV or JSON file you haven’t used before with > 1000 records	Dataset from data.gov	2
Multiple related CSV or JSON files with at least one file containing > 1000 records	Python Questions from Stack Overflow	4

*: Many of you worked with Facebook a lot in 106/506. Also, we are providing you with an OAuth2 example that uses Facebook, in this course, so we are considering Facebook to be “low challenge” for the purposes of this assignment.

**: If you choose “scraping a new single page” you can only use this option for *one* of your project sources (i.e., you can’t scrape 2 pages you haven’t scraped before and count it as 8 challenge points).

***: The challenge scores listed here are a guideline, but specific sources may be determined to be more or less challenging depending on the details of the source and how you're planning to use it.

✣: You *must* use at least one of these options as one of your data sources.

From each source, also need to capture at least 100 records (for CSV/JSON sources you need to capture at least 1000), and each record must have at least 5 "fields" associated with it.

If you have a source you'd like to use that you don't think fits neatly into one of these categories, consult with your GSI.

Data Access and Storage

You will need to create a database to store your data. Your database must have at least two tables, and there must be at least one relation (primary key - foreign key) between the two tables. Your data processing code (see below) must draw data from the database (i.e., not from the API/web page/CSV or from the cache).

If you are working with APIs or web pages you must *also* cache the raw results (JSON or HTML) you fetch from the source. Your code that writes data into your database must go through the cache when building the database.

As part of grading, we may use your code to rebuild the database (so this should be an option supported by your code) or ask you demonstrate this capability.

Data Processing

This is largely up to you, but you need to do whatever is necessary to support the data presentation(s) your program provides. This will probably involve things like creating dictionaries to collect sums or averages within a category (e.g., instances of crime by type, review scores by restaurant type).

Unit Testing

You must write unit tests to show that the data access, storage, and processing components of your project are working correctly. You must create at least 3 test cases and use at least 15 assertions or calls to 'fail( )'. Your tests should show that you are able to access data from all of your sources, that your database is correctly constructed and can satisfy queries that are necessary for your program, and that your data processing produces the results and data structures you need for presentation.

Data Presentation

Use a tool or framework to present data to users on demand. The data should be presented in some way *other* than `print( )` statements that output to the terminal. Your program must be able to produce at least 4 different graphs/displays/presentations. These can be different groupings of data, different graph types, or can differ in other ways (if you're not sure if they're "different" enough, check with your GSI).

The two options we have explored in class that you are most likely to want to use include:

1. Provide an interactive command line prompt for user to choose data/visualization options. Display selected graphs using plotly.

2. Create a Flask App that uses HTML links/form elements to prompt for the user to choose data/visualization options. Display selected data using HTML tables (or other elements, as long as the output looks good).

3. If you're feeling ambitious, you can figure out how to use plotly with Flask. It doesn't look too hard, actually: https://stackoverflow.com/questions/36288134/plotly-offline-with-flask. YMMV.

If you wish to use a different data presentation approach, you should check with your GSI.

What to Submit

There are three milestones to the final project, each with their own submission date and grading.

Proposal

Due March 25

Submit a half page to one page single spaced proposal describing your project plan. The due date is on Canvas. Your proposal must include:

1. The data sources you intend to use, along with your self-assessment of the "challenge score" represented by your data source selection.
2. The presentation options you plan to support (what information will be displayed).
3. The presentation tool(s) you plan to use.

Your GSI will review your proposal and potentially provide feedback (if the proposal is fine, you may not get much feedback!). In some cases, the GSI (in consultation with the teaching team) may decide that the proposal requires changes and ask you to submit a revised version. In such cases, you will be provided with instructions about what to revise and will be given a due date for the revised submission.

After submitting your proposal, any significant changes (e.g., to data sources or presentation plans) will require submission of a Final Project Proposal Revision via Canvas. Follow instructions there for how to notify your GSI that you have submitted a revision. It is strongly recommended that you discuss any proposed changes with your GSI before submitting a revision.

If you change your proposal plan without submitting a revision and obtaining authorization, you risk losing lots of points on your final project grade.

Proposal Rubric (40 points)

	Poor		OK		Good	
Data sources	Sources are not identified, or are described very poorly.	0	Sources are identified, but some information is missing	4-8	Sources are clearly identified, with URLs linking to a description of the source	10
Data source challenge score	Data source challenge score is not provided	0	Data source challenge score is provided by has errors or does not meet criteria (total >=8)	4-8	Data source challenge score is provided, correct, and meets criteria	10
Presentation options identified	Not identified	0	Presentation identified, but are not clear, are not sufficiently different, or do not meet criteria (options >= 4)	4-8	Presentation options are identified, different, and meet criteria	10
Presentation tools identified	Not identified	0	Tools are identified, but there are some issues with clarity.	4-8	Tools are identified and are appropriate	10

Data Collection Checkpoint

Due April 11

Submit a screenshot showing evidence that you have been able to collect data. At a minimum, you can show a portion of your cache file. For a better grade on this milestone, show a screenshot of your data in the DB Browser.

The due date for this checkpoint is in Canvas. This due date also marks the last chance you have to make significant changes to your proposal.

Checkpoint Rubric (40 points)

	Poor		OK		Good	
Evidence of data collection	No evidence provided	0	Evidence of cached data is provided	3	Evidence of cached data *and* data added to database	5

This score assigned *per data source* and is multiplied by the challenge points for that data source (maximum points is 40).

Final Project Submission and Demo

Due April 20
Demo April 19, 20

Via Canvas, You must submit a link to a GitHub repository containing your final submission. You will need to create this repository from scratch--we will not be using GitHub Classroom for the Final Project. Your GitHub repo must contain a README.md file that gives an overview of your project, including:

- Data sources used, including instructions for a user to access the data sources (e.g., API keys or client secrets needed, along with a pointer to instructions on how to obtain these and instructions for how to incorporate them into your program (e.g., secrets.py file format))
- Any other information needed to run the program (e.g., pointer to getting started info for plotly)
- Brief description of how your code is structured, including the names of significant data processing functions (just the 2-3 most important functions--not a complete list) and class definitions. If there are large data structures (e.g., lists, dictionaries) that you create to organize your data for presentation, briefly describe them.
- Brief user guide, including how to run the program and how to choose presentation options.

Your GitHub repo must *also* contain a requirements.txt file that can be used by the teaching team to set up a virtual environment in which to run your project.

Do *not* check in any private or secret information (e.g., API keys, passwords).

Demo Sessions

You will sign up to give a short (< 5-minute) demo to your GSI, following a script that we will provide as the deadline approaches. We are planning to hold demo sessions on Thursday, April 19, 2018 8:30-10am and Friday, April 20 between about 1-5pm. The precise times and location may change slightly and will be announced later. Note that the project is due at 11:59pm on April 20, but you will need to have finished at least enough of your project to complete the demo a bit earlier.

If you are unable to attend a demo session during the scheduled times, please contact the teaching team as soon as possible to make alternative arrangements.

Note that your project needs to *also* be able to run on a grader's computer, potentially with some setup work (e.g., specifying API keys/client secrets in a particular file/format).

Final Project Rubric (220 points)

	Poor		OK		Good	
Data sources: challenge and access	Challenge level is substantially below expectation and/or external data is not accessed by the program.	0	Challenge level falls short of criteria and/or program access some but not all required information. Program may not be able to access all relevant information from one or more source.	12-24	Challenge level meets criteria. Program successfully access at least 100 records from all data sources (all records if CSV or JSON file)	30
Data storage: caching	No caching is used.	0	Caching works for some but not all sources.	8-16	Program uses caching correctly for all web-based sources.	20
Data storage: database	No database is used.	0	Program uses only one table, or fails to model at least one relationship between tables.	10-20	Program writes to at least 2 database tables and uses relations appropriately	25
Data processing	No data processing, only raw data is displayed.	0	Program produces data structures and results needed for presentation, but poor choices are used for data structures and processing.	8-16	Program creates data structures and data processing results needed for presentation. At least one class is defined. Lists and dictionaries are used where appropriate.	20
Unit testing	No tests provided, or all tests failed.	0	Some tests are defined, but the number, coverage,	10-20	The number, coverage, and quality of tests is	25

			and/or quality do not meet stated expectations.		adequate to show that data access, storage, and processing work well.	
Presentation: interaction	No interactive capability shown, or no presentation options offered.	0	Presentation options and diversity are below expectations, and/or interactive options and input are presented poorly and/or input error handling is poor.	12-24	The number and diversity of presentation options meet project criteria. Interactive input is presented clearly and errors are handled gracefully.	30
Overall quality	No presentation is shown or project lacks any coherence.	0	Data presentation lacks clarity or professionalism, and/or the project lacks coherence or insight.	8-16	Data presentation is clear and reasonably attractive. The project is coherent and produces interesting insights.	20
Code quality: modularity, comments, readability	Can't happen?	0	Code is messy, difficult to read, or employs Python features poorly.	8-16	Code is well written. Python features such as classes and functions are used appropriately. Code is readable (line length, variable and function names, small functional blocks where possible).	20
Demo presentation	No demo.	0	Demo reflects poor preparation and/or some elements are missing.	10-20	Student is well prepared for demo and shows all required elements clearly and efficiently.	30
Total						220