ItoP: Assignment 4

Introduction

This assignment differs from previous assignments in that it will be open ended and that **you may work in a group of up to three people**. Furthermore, the graded deliverable will be a report instead of code (though you must still attach your code).

Directions

The directions are simple: put together a Python project that exemplifies the material you've learned throughout the semester or any other areas of Python that you've explored. The project may be anything of your choosing, as long as it utilizes Python substantially. I **highly recommend** you email me your idea for the project before starting on it to make sure it is an acceptable use of Python. If you turn in an assignment that I deem insubstantial, and we did not have a prior discussion about the idea, you will receive a 0.

I don't mind if your project is just the result of a tutorial, such as this Flask mega-tutorial. If you follow a tutorial, you must cite it in your report. The tutorial must still qualify as "a substantial use of Python", so you should ask me about it first.

I have listed a few ideas in the Project Ideas section.

Report

The graded deliverable for this assignment is a report covering, at the least, the following topics:

- The first and last name and EID of all group members
- The goal of your project
- A demo of your project including screenshots, screen recordings, log files, graphs, and/or any other relevant content
- Directions on how to run your project, including all files, libraries, and other setup necessary
- A comprehensive list of references covering any material you learned from and any existing code you used

There's no page limit requirement or recommendation for the report. Just be sure it covers the topics above. **Keep these two guidelines in mind when writing the report:**

- 100% of your grade is coming from the report. If you spend 30 hours on your project and then submit a poorly written report, your grade will suffer.
- From my perspective, the most important part of the report is explaining what you learned about Python by working through your project (Note: this isn't explicitly listed in the topics above because it should be evident throughout the report).

Instructions continued on the next page

Submission

You should submit a zip file to Canvas called `as4.zip`. The zip file should contain a PDF called `README.pdf` and a directory called `src`. `README.pdf` will be your report in PDF format. All of your source code for the project should be put in the `src` directory. If you do your project in a Jupyter Notebook, you may optionally submit a single file in the zip directory called `README.html` (by exporting your notebook to HTML format). You must still meet all the requirements for the report and project if you submit a single `README.html` file.

Project Ideas

The following are possible ideas for the final project. These are just suggestions to indicate the level of Python I expect your projects to exhibit. Feel free to follow one of these ideas exactly, modify one of them, or come up with something completely independent for your project. The difficulty of each idea is indicated by the number of ☠ icons (max of 5), relative to the material we have covered in lecture.

Disclaimer: I haven't necessarily done these exact projects on my own, but I have either 1) done similar projects in Python or 2) done similar projects in another language and have a sense of the API design/complexity in the analogous Python libraries. Please take the difficulty ratings with a grain of salt.

Flask website

We covered a very basic Flask webpage in discussion section, but most substantial webpages will at least have a database attached to the backend. You may build any webpage that sounds interesting to you, but it must have a database and some dynamic content (i.e. the HTML page is constructed with some interaction with the Python backend). If you want to add extra flair, you can also learn CSS to make your webpage look more modern. Bootstrap 4 is a drop-in set of styles that can make styling very easy without having to deal with CSS yourself. It will be helpful to install `pip` as your Python package manager when building a webpage.

Some concrete examples:

- Flask mega-tutorial through at least chapter 7, inclusive. (☠☠)
- A small social media site in which you can log in to an account, add friends by their username, and share text posts that are visible on the home page of all your friends. Similarly, all of your friends' posts will show up on your home page. (☠☠)

Graphing calculator

Build a graphing calculator that asks the user for an arbitrary expression for $f(x)$ and plots a line graph (☠, if you use the right Python features). You can also expand the calculator to support expressions for 3D graphs (☠), polar graphs (☠), parametric graphs in 2D (☠), parametric graphs in 3D (☠), and/or other types of graphs.

Instructions continued on the next page

Chat service

Python has support for a network stack through sockets. Build a chat service in which one computer acts as a server that can relay messages between other client computers. A single server hosts a single chat room in which all participants can send public messages to one another (☠☠☠). The chat service can be upgraded to include other features such as multiple chat rooms (☠), direct messages (☠), or sending memes (☠☠).

Data analysis notebook

Create a Jupyter Notebook that performs data analysis on some data set. Since data analysis is more about the data than the actual Python code, it may be difficult to judge whether the project uses Python substantially or not. Instead, the requirement will be to discover some insight into a data set using Python libraries (☠☠☠). You must provide well documented evidence of your insight and most likely several graphs. A simple insight was how we discovered that, based on IGN reviews, the PS4 has higher quality games than the Xbox One. I made the claim half-seriously, but, for example, you may be able to discover much more concrete evidence in the data set to back the claim. Kaggle is a great resource for finding data science problems of all difficulties and possibly even earning a monetary reward.

Other ideas

- Build a game using `pyglet` or `pygame`.
- Do something clever with computer vision using Python bindings for OpenCV.
- Train and use a classifier with `scikit-learn` or some other machine learning library.
- Build a useful Python application with a desktop GUI made with PyGUI, Python bindings for GTK+ 3, or some other windowing toolkit.