Author: William Casey Bellew
GitHub username: psykibear
Date: 3/13/2023

- What stood out to you as the most important things you learned about computer programming this quarter? Why do you think they're important?
    - One of the most important things I've learned is the structure of coding and making the coding self-explanatory. After working in the engineering field for the past 10 years I know how difficult it can be coming in after someone not knowing where they left off or why they build a process a certain way. If you build your code and document appropriately it makes it to where any user or future developer can look at the code and understand it without having to run it line by line or sequence by sequence to fully see the process developed. This is pivotal in the software development field because you are one of many looking at and working on the same setup of code whether it is backend, frontend, or full stack.
- What topics do you still find confusing? What can you do to get a better grasp on them?
    - Some of the differences in data sets is still a bit confusing on my end. I have begun further reviewing other YouTube video series to strengthen my knowledge while seeing more exampled. I have also enrolled in an EdX course for Python and purchased the additional reading material book to further give me an understanding of these data set types and why they function in certain ways or be used in a certain way.
- What did you learn this quarter about how taking online courses works for you?
    - Time Management and Community Resources.
    - I've had issues with making sure that I've dedicated enough time to the assignments while having the proper amount of time to truly work through the examples given in the notes/modules. I quickly learned early in the semester so that I could increase my chances for an A in the course.
    - Community resources assisted me in this course by discussing some of the theorems with classmates via discord and through the forums through OSU. Software engineering and computer science is a widely discussed career path throughout all levels of the knowledge basis. Learning to use these resources accordingly can make the difference between a good and bad developer.
- How did you use the Exploration and the videos provided? What techniques helped you learn better and when they didn't help you, what did you do?
    - The exploration portions of the modules gave me good opportunity to understand the changes in code that has already been written while stepping through the lines of code to understand it's intent. I appreciated the "nudges" in the directions of how the code is supposed to perform based on the suggested changes in the code to specific lines and seeing how they change/react.
    - I did not get as much out of the videos as I did the demonstration/example codes. The ability to review the videos repeatedly did help when I ran into issues with my own code for assignments when they did not pass the auto grader. Though, fully admitting, there seemed to be plenty of discrepancies between how my code was written and how the auto grader "passed or failed" my code. If not named appropriately or built in a slightly different method it would not pass the auto grader, but other times it would pass the auto grade while receiving significant point deduction for things that did not function as they were supposed to when analyzed by the staff.
    - I continued to use online resources such as YouTube to further understand where I was stuck on ideas discussed in the modules. The additional reading material was extremely helpful as well so that my mind could be put into a different "mode" of thinking to look at the topics more

generally than specifically. There are times where I have the habit of going to specific or "into the weeds" rather than understanding the general concept.

- How can you generalize your experience in this course in a way that will help you in future courses?
    - Practice, Practice, and more Practice!
    - The best thing that I took away from this course is coming up with additional ways to practice the methods being taught either through the examples/exploratory sections or through examples found in the community resources. Software development, like other engineering disciplines, comes better with experience and practice. The more you put into your thought process and documentation of thoughts and ideas the more I was able to take away from the lessons/modules. I developed additional notes for myself by answering the questions I had after reviewing the modules for myself before reaching out to the TA's or on the forums.
- How has this course helped crystallize your ideas about what a career that involves computer programming might be like?
    - In my mind, this course helped further build on the idea that a career in computer programming is further built upon the community and the resources you have available. This availability can be based off your company resources along with your networking and self-starting mindset of programming.
    - The modules sparked several discussions and debates with my peers on how they could be used in "real like" situations. This includes some of the situations that we have reached out and found through our own communities in our hobbies. I play an online game that continuously writes to a log file on how well you are performing. I use a website that takes that log file to further understand how well you are doing and what could potentially be done to improve it. Until further investigating it based on some of the teachings in the course, I did not fully understand how it was pulling the data through the logs, organizing it, and analyzing it to meet the end user's need. This took me down the path of "what would bring me a challenge or joy in a career of software development or computer programming?" This has led me into a thought of doing back end development and data science/analytic work would possibly be something I'd enjoy as a career path.