CW2 Kyle Spicer
Module: G
Practical Name: practical-1a
Due Date: July 13th, 2022

Project Summary:

Practical 1-A’s goal was to create a class and a module that would extract bank account objects by values of various properties. The student was required to create class and a module with various functions. The Python bank account class creates objects for use in a separate module. A utilities module was required that consisted of various functions, read in an outside .txt file, and produce appropriate output after being cycled through the functions.

Challenges:

1. Creating and using a class. The instruction was great for learning Python classes and why we use them, however, when it came time for me to call my class in another module I had some trouble. After researching and several attempts, I was able to properly utilize my class.

2. Creating functions to read a file, create objects, and return proper output. I had a lot to learn with formatting and logic of unique functions. I spent a large amount of time ensuring my functions worked properly and interpreted the object correctly.

3. Properly utilizing Git. This project I focused on using Git as it was designed to. Understanding the branch structure and how to properly work on a file and push your changes will be instrumental in future projects.

Successes:

1. Formatting output. Printing the proper output went really smooth for me. Once my programs worked as they should and I properly called my functions, formatting the output was simple.

2. Pycodestyle. Running my programs through pycodestyle to ensure my formatting meets PEP8 standards went really well. Although there were many corrections, I was able to format my programs to return no errors.

Lessons Learned:

1. Creating a design plan prior to beginning the project really helped me understand what my goal was and what I was trying to achieve. Prior to this project, I would just begin writing code with no direction, however, this project I needed to sit and think about my attack. This helped me remain focused and organized as I created my programs.

2. Build a little, test a little. As I was writing code, I would constantly stop and test my work. This helped me understand what was actually happening with my code. By thoroughly testing, I was able to catch errors or mistakes at that point in time, rather than waiting until later and potentially having issues identifying the origin.