2-2 Milestone 1: Code Review

Bryan Burnett

SNHU CS-499 Computer Science Capstone

Prof. Brooke Goggin, M.S., M.S., M.S., Ed.D (ABD) Computer Science Adjunct Faculty

May 9, 2023

**Code Analysis:**

Here is the code analysis for my ATMfinal.py program that I will be working on during this course. I will start by describing the existing functionality of the program.

**Existing functionality:**

First, the account balance is defined as a float for the functions to utilized as a starting point. Next, the functions are defined as ‘printbalance’, which displays the current balance. The ‘deposit’ function allows the user to add additional funds to the account balance. The ‘withdraw’ function allows the user to subtract funds from the account balance if the withdraw amount does not exceed the account balance. These functions are placed in a list of options for the user to choose from called ‘userchoice’. These options are ‘D’ for deposit, ‘W’ for withdraw, ‘B’ for balance, and ‘Q’ to quit the program. ‘Q’ for quit the program is an undefined function that simply prints a message to the screen and then the program terminates. If none of these options are chosen, a message declaring ‘Not a valid choice.’, is displayed on the screen, and the program terminates. The program will terminate after whatever function is called and performed. There is currently no ability to perform multiple functions.

**Structure**

So, the structure needs work.The current configuration of the code does not completely implement the design. There is no login process, and only three options for the user to choose the program to do. There are no security measures in place.

The code does currently meet some of the Pep-8 standards for Python coding practices. Utilizing explicit code, one statement per line, correct function arguments, and correct return values. The code is consistent in style and format. The structure is only partially there, as it is an incomplete program covering minimal functionality. Because of the limited functionality of the code, there is no unreachable code or unneeded procedures. There are no leftover stubs or test routines. Nor any blocks of repeated code. There are no magic numbers or string constants, everything is named appropriately and does a specific function. Because of this, each function exist independently of each other until the loop calls them.

The current configuration does do some of the things a real-world ATM program should do, however an entire library will need to be built to support the multiple accounts, and calls to the library will need to replace existing functions. Storage use is efficient and will remain so, because Python has a built-in memory management system, so that does not and will not need to be programmed in.

**Documentation**

The code is clearly documented, but minimally. Each comment officially naming the function that follows it by what the function does. (For example, #deposit function tells me the function is the deposit function of the code.). The style is easily maintainable by simply updating the comment to suit the functionality.This is becausethe comments throughout the code are consistent and simple.

**Variables**

The variables are properly defined with consistent, and clear names. There are only three functions at present though. Consistent, clear, meaningful names will be utilized in the upcoming design.Each of the assigned variables currently has proper type consistency and will in the upcoming enhanced version. The variables are all used in the current code. There are redundancies and limitations in the code however, and they will be cleaned up in the revision.

**Arithmetic Operations**

Currently, thecode does not compare floating-point numbers for equality, it only compares numbers so that they do not exceed the balance for the withdraw function purposes. There is no need to prevent rounding errors currently. There is nothing in place in the code to round numbers or any call to round numbers. This will not change in the enhanced version.

The code does actively avoid the subtracting of numbers that exceed the balance it can subtract from. But there is no testing in place against zero or noise. This could potentially be enhanced to prevent overflows or data corruption.

**Loops and Branches**

All the loops, and logic constructs are properly nested, and correct, but they are incomplete for where the program is to go. However, for its current state, and no need for the program to be anything other than what it is currently, it is complete and functional. There are only two cases present. Each if/else case is presented in the ‘withdraw’ and ‘userchoice’ functions of the code.

The if/else cases are covered by else clauses in the form of various print statements.Every case statement has this default. But offers no further functionality outside of the print statement.

The loop termination conditions are obvious, and achievable, as they are absolute. Each function is a single operation that immediately terminates the program after the function is performed. This will be enhanced to allow for better, more efficient, meaningful functionality of the program.

The functions are defined and initialized prior to the loop as appropriate.And there are no statements enclosed in the loop that should be placed outside the loop.The code avoids any further use outside the defined function as each function terminates the program after it is performed. Each time a function is to be used the program must be reopened. This is a noted design flaw, but somehow a security feature in its present state. This too, will be enhanced be a secure closed loop that allows for greater, more secure functionality of the program.

**Defensive Programming**

Currently, the only index tested against a file bound in place is in the ‘withdraw’ function to check the withdraw amount against the current balance. As such, it is also the only input argument test in place.

The output variables are assigned, although limited.The correct data is operated on in each of the functions.

Memory allocation is deallocated. Although, there is no current memory deallocation method in place as Python automatically allocates and deallocates memory via a built-in memory management system.

Currently, there are no time outs or error traps of any kind. Each function performs its assigned task and quits the program after completion. This will be enhanced to accommodate the more robust program.

Files are currently checked for existence prior to access, and if the file does not exist, there is a print statement that notes ‘Not a valid choice.’ when an invalid choice is selected.

All the files are closed entirely after performing the selected function. There is no way to continue the program. Each function is performed upon being called, then the program is immediately terminated. This will be enhanced by offering more functionality in a much more efficient manner.

**Code Analysis Summary:**

The code as it stands is very rudimentary and simple. Areas in desperate need of improvement are Functionality, Security, Testing, Documentation, and Structure. First, Functionality, it offers limited use and functionality. The need for expanded functionality and the option to do more than one function per use of the program is dire.

Security, the ability to manage any other account, protected by passcodes unique to the unique user or account number. Also, an end to the self-termination of the program at each call, by securely allowing the user to exit the program.

Testing, there is not much in the way of testing currently in place. Only utilizing an if/else statement in the ‘withdraw’ function to ensure fund availability. There are no current other tests in place.

Documenting is minimal, but accurate. The documentation will need to be expanded as more complex code is fitted and functionality is broadened.

Structure, the structure will need to be reformatted upon improving all the aforementioned areas of improvement. The entire program will need to be remade to accommodate the design improvements and meaningfulness of the program. Currently, the program terminates after a single function call. It performs the called function and terminates without any further options given. This will be enhanced to promote a more meaningful experience.

**Enhancements Summary:**

The enhancements planned are to add a secure way to view individual accounts. No longer just a part of a view of one account. This will need to be secured by a unique passcode or PIN. Improved functionality of the existing code with expanded options to better suit the user’s needs. The ability to better navigate to, and perform multiple functions in one session, and the ability to exit the account securely after doing so.

Things like improving the existing functionality, making it more efficient, and giving the ability to transfer funds to another account. To ensure security and functionality, there will need to be testing in place to prevent overflows and security vulnerabilities.

Documentation of all this new code will need to explicitly describe what the code does at each function and loop.

**Thank you for your time and consideration.**