Cmpe 150 Bonus Lab: General Advice and Common Errors

Problem Description

- Read the problem description carefully. Unless you understand it, you won’t be able to solve it.

- You don’t need to check the points described as assume that … It is provided to help you, not to make your job harder.

Inspect the Code Provided

- We want you to fill in the function part, yet the stuff below is also understandable, not something super strange.

- You ought to be able to see
  - 1) Reading necessary inputs from the user
  - 2) Calling the function with those inputs
  - 3) Printing the returned value

Think Before Coding

- If you are not sure about the exact solution methodology you will use (Algorithm), you cannot describe it to someone else. Similarly, you cannot explain it to a computer, either.

- SIMPLE SOLUTIONS MIGHT BE GOOD. ALSO, COMPUTERS ARE FAST, AND THEY DO NOT GET TIRED, SO USE IT.

Indentation and Parameters

- Indentation is super important. Whenever we use “:” (ESPECIALLY FOR FUNCTIONS), do not forget to adjust the indentation of the following lines accordingly (use tab). In addition, where the indentation ends is also very crucial. For each line inside the function, 4 spaces (tab) before the statement.

- If something is stated as a parameter of a function, do NOT try to read it from the user. It is our job to read that and pass it to the function.

Functions

- Inside the function, you can use only the parameters of the function. Any variable defined before is no longer usable (even if the name is the same) if it is not passed as a parameter. In fact, all the parameters provided must be enough to write the function. What we have is only parameters. However, of course, you can define new variables, yet if you are reading something from the user for them, with 99% probability, you are doing something wrong.

Functions (Cont.)

- “return” and “print” are different things. When return is executed, that function immediately stops and the execution jumps back to where it came from along with the returned value. If we say return, do NOT print anything since it is our job to print the returned value after the function call. We do a string comparison between the output of your program and the expected output, so anything printed additionally would make it wrong.

Testing Code at Intermediate Steps

- Writing the whole function at once is harder than dividing it into subtasks and writing each of them.

- For each subtask, you can use a new function or at least check if you are OK with the part that you wrote by using things like print.

For Loops

- If you put a return statement inside the loop without any if statement, it would return in the first iteration. Usually, we want to put it after the loop is over. (INDENTATION)

- If you define a variable inside the loop, it cannot be used to accumulate the summation or something similar.

Variable Naming

- Information type names or function names cannot be a variable name:

  int, float, str, list, dict, set, tuple, print, input

- However, any small change is sufficient to solve the issue:

  INT or Float or my_str or List or dict1 or …

None as Output

- When you run your code, if you are getting an output containing “None” or getting and error saying something like “TypeError: unsupported operand type(s) for +: 'NoneType' and 'int' ” the reason is, almost always, calling a function which does not return anything. If it is your function, then return the appropriate value inside it.

- READ ERROR MESSAGES IN GENERAL. THEY ARE SHOWN TO HELP YOU, NOT TO HURT YOU. IT IS BETTER THAN JUST SAYING “ERROR”. LOOK AT THE LINE CAUSING THE ERROR.

Tester

- IT DOES NOT GUARANTEE CORRECTNESS. MAKE SURE THAT YOU TEST YOUR CODE WITH OTHER INPUTS THAT YOU PROVIDE BY RUNNING Main.py

- Sometimes, try to execute your code by imagining all the steps that a computer would take to run the code. It will help you understand better.

If Multiple Functions Defined

- If you want to define new functions to be used in the original function, that is fine, but be sure that you test each of these new functions as well. You can use main.py (Not Main.py) in the exams for that.

- In addition, be sure that you do not do anything new outside # lines.

To Improve

- Practice
- Doing projects to really improve if you are planning to focus on programming in the future.