COMP 202 - Foundations of Programming

Assignment 4 - General Guidelines

McGill University, Fall 2023

Directions:

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, allowing the TAs to provide better feedback and not waste time on administrative details. To get full marks, you must follow all directions below:

- Make sure that all function names are **spelled exactly** as described in this document. Otherwise, a 50% penalty per question will be applied.
- Make sure that your code **runs without errors**. Code with errors will receive a lower mark.
- Write your name and student ID in a comment at the top of your file.
- Name your variables appropriately. The purpose of each variable should be obvious from the name.
- Comment your code. A comment every line is not needed, but there should be enough comments to fully understand your program.
- Avoid writing repetitive code, but rather call helper functions! You are welcome to add additional functions if you think this can increase the readability of your code.
- Lines of code should NOT require the TA to scroll horizontally to read the whole thing.
- Vertical spacing is also important when writing code. Separate each block of code (also within a function) with an empty line.
- Up to 30% can be removed for bad indentation of your code, omission of comments, and/or poor coding style (as discussed in class).
- The work submitted for this assessment is expected to be your own. The use of technologies such as ChatGPT are prohibited and will be considered a violation of the Code of Student Conduct.

Hints & tips

- Start early. Programming projects always take more time than you estimate!
- Do not wait until the last minute to submit your code. Submit early and often—a good rule of thumb is to submit every time you finish writing and testing a function.
- Write your code **incrementally**. Don't try to write everything at once. That never works well. Start off with something small and make sure that it works, then add to it gradually, making sure that it works every step of the way.
- Read these instructions and make sure you understand them thoroughly before you start. Ask questions if anything is unclear!

- Seek help when you get stuck! Check our discussion board first to see if your question has already been asked and answered. Ask your question on the discussion board if it hasn't been asked already. Talk to your TA during office hours if you are having difficulties with programming. Go to an instructor's office hours if you need extra help with understanding a part of the course content.

  At the same time, beware not to post anything on the discussion board that might give away any part of your solution—this would constitute plagiarism, and the consequences would be unpleasant for everyone involved. If you cannot think of a way to ask your question without giving away part of your solution, then please drop by our office hours.

- If you come to see us in office hours, please do not ask "Here is my program. What's wrong with it?" We expect you to at least make an effort to start to debug your own code, a skill which you are meant to learn as part of this course. And as you will discover for yourself, reading through someone else's code is a difficult process—we just don't have the time to read through and understand even a fraction of everyone's code in detail.

  However, if you show us the work that you've done to narrow down the problem to a specific section of the code, why you think it doesn't work, and what you've tried to fix it, it will be much easier to provide you with the specific help you require and we will be happy to do so.

- The work submitted for this assessment is expected to be your own. The use of technologies such as ChatGPT are prohibited and will be considered a violation of the Code of Student Conduct.

Learning Objectives:

The main learning objectives for this assignment are:

- Correctly create and use variables.
- Learn how to build expressions containing different type of operators.
- Work with lists and dictionaries
- Work with strings
- Work with files input/output
- Work witch catching exceptions
- Apply what you have learned about Object-Oriented Programming OOP – classes, methods, constructors, attributes, objects etc. Note that this assignment is designed for you to be practicing what you have learned up to and including Lecture 21. For this reason, you are NOT allowed to use anything not seen in class at all. You will be heavily penalized if you do so.
- Understand how to write a docstring Note that this assignment is designed for you to be practicing what you have learned in the Lecture 11 (including strings and lists). For this reason, you are NOT allowed to use anything seen after or not seen in class at all. You will be heavily penalized if you do so. For full marks, make sure to add the appropriate documentation string (docstring) and following programming standards to all the functions you write. The docstring must contain the following:
- The type contract of the function.
- A description of what the function is expected to do.
- At least three (3) examples of calls to the function. You are allowed to use at most one example per function from the assignment PDF.

Examples: For each question, we provide several examples of how your code should behave. However, it is your responsibility to make sure your code/functions work for any inputs, not just the ones shown in the examples. When the time comes to grade your assignment, we will run additional, private tests that will use inputs not seen in the examples. You should make sure that your functions work for all the different possible scenarios. Also, when testing your code, know that mindlessly plugging in various different inputs is not enough—it's not the quantity of tests that matters, it's having tests that cover all of the possible scenarios, and that requires thinking about possible scenarios.

Furthermore, please note that your code files should not contain any function calls in the main body of the program (i.e., outside of any functions). It is OK to place function calls in the main body of your code for testing purposes, but if you do so, make certain that you remove them before submitting. You can also test your functions by calling them from the shell.