STEM College

Assignment 2

COSC2676/COSC2752 Programming Fundamentals for Scientists

March 2021

	Assessment Type: Individual assignment. Submit online via Canvas → Assignments → Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
	Due date: Week 8, Friday 30th April 2021, 11:59pm. Late Submissions/Extensions: A penalty of 10% per day is applied to late submissions up to 5 days, after which you will lose ALL the assignment marks. Extensions will be given only in exceptional cases; please refer to the special consideration process.
	Weighting: 35%

1 Overview

For this assignment, create a study/simulation/tool/utility of your choice related to your program of study in Python that comprises a number of features, as described in Section 3.1, to demonstrate the code concepts detailed in Section 3.2, and analyse and evaluate your code and provide adequate documentation, as described in Section 3.3.

Note: Your program must relate to your program of study.

You must not just "throw in the concepts" to your program just because they need to be there; it should be clear from the code why a certain concept should be there, and you must further explain these through your comments. You will also need to debug your code on your own and document any issues, etc. You are given marks on your ability to fulfil all requirements of this document.

There are design requirements (6 marks), code requirements (18 marks), and documentation requirements (11 marks) for a total of 35 marks.

2 Learning Outcomes

This assessment relates to all of the learning outcomes of the course which are:

- CLO 1: Demonstrate knowledge of basic concepts, syntax and control structures in programming.
- CLO 2: Devise solutions to simple computing problems under specific requirements.
- CLO 3: Encode the devised solutions into computer programs and test the programs on a computer.
- CLO 4: Demonstrate understanding of standard coding conventions and ethical considerations in programming.

3 Assessment Details

There are design requirements (6 marks), code requirements (18 marks), and documentation requirements (11 marks) for a total of 35 marks.

3.1 Design Requirements

The following design requirements must be clearly stated in your report.

3.1.1 The objective(s) and specifications of the study/simulation/tool/utility are clearly outlined. **(3 mark)**

3.1.2 There are at least 3 features that serve the objective(s) of the proposed program. A feature is a distinguishing characteristic of a program (i.e., what a program can do). The mechanics of a program (i.e., how a program works) is not a feature. **(3 marks)**

3.2 Code Requirements

The following code requirements must be applied to serve the objective(s) of the proposed program.

3.2.1 Presentation and formatting. The filename(s) must be chosen to match the application (default names such as assignmentX.py will not be accepted). Names of identifiers (including functions, variables, etc.) must be descriptive. Code formatting is clean and consistent. Must not include any unused/irrelevant code. **(3 marks)**

3.2.2 Data types. The program must process at least four different data types with at least one data type of the following: list, tuple, set, or dictionary. **(3 marks)**

3.2.3 Control flow. The program must have at least two control flow blocks, i.e., if/elif/else and/or for/while loops. **(3 marks)**

3.2.4 Files. Your program must use external input/output file(s) (e.g., .txt, .csv, or .png). Note: you will need to submit these files. **(3 marks)**

3.2.5 Functions. Demonstrate concepts of modular code organisation using at least two functions. When the program is run, there should be a pathway to run/call all functions (i.e. no unused functions). **(3 marks)**

3.2.6 Design. All libraries (if any) are imported at the top of the file. Functions and other constructs are defined at the appropriate level. The design and organization of the overall program is sensible, modular, and readable. **(3 marks)**

3.3 Documentation Requirements

3.3.1. Line 1 of your .py file must be an example of the command(s) needed to run your program. **(1 mark)**

3.3.2 Comments. Adequate code comments provided to improve the readability of your program. Write comments after function definitions, before code blocks, and in line with important variable declarations. The code comments in this assignment must focus more on the analysis of your approaches and their evaluation instead of simply translating the Python code to English (see rubric in Section 9 of this document). Explain any code requirements that you have not met and all bugs (situations that might cause the program to crash or behave abnormally) in the approximate locations they originate within your code. If you do not have bugs, you must explicitly say that there are none. A good programmer knows the limits of their program. **(4 marks)**

3.3.3 Submit a report in a text file (.txt) or PDF format that contains the following. **(3 marks)**

1. Introduction (including objective(s) and specifications of the study/simulation/tool/utility)
2. Relation of the study/simulation/tool/utility to your program of study
3. Features
4. Methodology (e.g., design decisions, devolvement process, etc.)
5. Results (e.g., sample inputs and outputs, figures, etc.)
6. Discussion (e.g., explain any code requirements that you have not met and all bugs (situations that might cause the program to crash or behave abnormally) in the approximate locations they originate within your code. If you do not have bugs, you must explicitly say that there are none. A good programmer knows the limits of their program.)
7. Conclusion
8. References

3.3.4 Submit a 2-5mins video or a screen recording that contains the following. **(3 marks)**

1. Introduction (including objective and specifications of the study/simulation/tool/utility)
2. Relation of the study/simulation/tool/utility to your program of study
3. Features
4. Discussion on the methodology/design of the program.
5. Demonstration of how to run the program and sample output.

4 Submission

Submit .py file(s), any data/input files, report and a link to the video via Canvas → Assignments → Assignment 2 as a single .zip file called *a2_{firstname}_{lastname}_{student_id}.zip.*

It is the responsibility of the student to correctly submit their files. Please verify that your submission is correctly submitted by downloading what you have submitted to see if the files include the correct content. Please make sure your video is accessible via the provided link.

Assessment declaration:
When you submit work electronically, you agree to the assessment declaration:
https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/assessment-declaration

5 Academic Integrity and Plagiarism

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to
https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

6 Marking Guidelines

The following rubric will be used to assess your work.

Additional Penalty:
A 30% penalty will apply if the study/simulation/tool/utility does not relate to your program of study.
A 10% additional penalty will apply for every compile/runtime error in your code, e.g., syntax errors.

	Inadequate	**Partial**	**Complete**
3.1.1	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	The objective(s) and specifications of the study/simulation/tool/utility are clearly outlined.
3.1.2	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	There are at least 3 features that serve the objective(s) of the proposed program.
3.2.1	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	File(s) submitted with all required code. File name suits program and is not an arbitrary or default name. Identifier names are descriptive. Presentation is consistent. Only relevant code and comments included.
3.2.2	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Program processes at least four different data types with at least one data type of the following: list, tuple, set, or dictionary. Demonstrates understanding of bool vs int vs. float vs. string types where relevant. Data inputs must not always be of the same size/length; program must be able to handle, for example, images of

			different dimensions, etc. Above demonstrated as a part of meeting overall functional aims of program.
3.2.3	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	The program must have at least two control flow blocks. Uses if/elif/else appropriately for non-repeating conditional execution. Conditions do not include tautologies and pathways are not redundant. Every code block in every if/elif/else structure is reachable. Uses loops appropriately for repetition. Loop condition must describe all situations under which the loop will repeat, and condition must fail eventually. Conditions do not include tautologies. Above demonstrated as a part of meeting overall functional aims of program.
3.2.4	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	Program uses external input/output file(s) (e.g., .txt, .csv, or .png). Above demonstrated as a part of meeting overall functional aims of program.
3.2.5	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	Has at least one function in addition to the “main” function. Functions used in ways to reduce code repetition. Every created function is used in the program. Methods may contain parameters and/or return values. Above demonstrated as a part of meeting overall functional aims of program.
3.2.6	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	All libraries (if any) are imported at the top of the file. Functions and other constructs are defined at the appropriate level. Only function declarations are at top-level (e.g. no global variables/lists). Avoids hard coding when appropriate. The design and organization of the overall program is sensible, modular, and readable.
3.3.1	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	Includes example of required command(s) to correctly run the program to demonstrate all concepts (must supplement by submitting required input files, etc.).

3.3.2	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	Identifies concepts required and application of each concept is discussed with respect to the overall aims of the program. Explains any code requirements that have not been met, situations that might cause the program to crash or behave abnormally. Able to identify the approximate locations of any issues within the code and comments are given in those locations. Has an even spread of the above near blocks and important variables in plain English where possible. Analyses the breaking down of the programming task to smaller parts (e.g. methods and other blocks) and organization to achieve the overall goals of the program. Evaluates strategies used to meet requirements and compares the quality of one approach to another approach. Has an even spread of the above near blocks and important variables in plain English possible.
3.3.3	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	Report contains introduction, objective, specifications, relation, methodology, results, discussion, conclusion, and references. All sections are sound and correct.
3.3.4	More than one of the criteria in the ‘complete’ level missing/incorrect.	Any one of the criteria in the ‘complete’ level missing/incorrect.	Video contains introduction, objective, specifications, relation, methodology/design/discussion, and results. Presentation is clear, sound, and correct.