COSC2452 Introduction to Programming

Week 8 Major Assessment Project

(v.2021.10.15 – Refer to section 6: Document revision history)

Assessment Type: Individual assessment (no group work). Do not even show your code to anyone as both parties could be found breaching RMIT Academic Integrity rules. Marks are awarded for meeting requirements as closely as possible by exclusively using the concepts demonstrated by Gayan during Weekly Live Lectorials. Clarifications/updates will only be made via Canvas->Discussions-> Week 8 Major Assessment Project Clarifications forum by Gayan. (No email clarifications).

Deadline: Deadlines may be extended. Please always follow dates under Canvas->Assignments.

Extensions: If you do not have existing special consideration or a medical certificate for 1-7 days, apply for special consideration online; getting in touch with your tutor/instructor first will lead to delays. Those who have existing special consideration or a medical certificate for 7 days or less must email gayan.wijesinghe@rmit.edu.au using only their official @student.rmit.edu.au account.

Marks: 20 + bonus marks

Table of Contents

1. Expectations 2
2. Learning Outcomes 2
3. Assessment details 3
  3.1 Functional Requirements 3
  3.2 Code+Justification Requirements (+20 marks) 4
    3.2.1 Code Requirements 4
    3.2.2 Justification Requirements 6
  3.3 Documentation Requirements (-6 mark penalty if not met) 6
  3.4 Application Theme Requirement 7
  3.5 Receiving Supervision 7
4. Submission 8
    4.1 Assessment declaration 8
5. Academic integrity and plagiarism (standard warning) 8
6. Document revision history 9

1. Expectations

There is no book containing the music to every song that will be written. There is no book containing the answers to every mathematical calculation that we will need to perform. Similarly, there is no book, set of lecture slides, video, etc. that will give a programmer (you) the solutions to every programming problem.

This assessment requires you to demonstrate exclusively a set of concepts and approaches covered during the weeks 1-7 (inclusive) Weekly Live Lectorials by creating one application of your own choosing from a list of application themes provided for inspiration. Your application must adhere to constraints and requirements in this document which will help you complete this assessment.

Consider the simplest solution that meets the requirements is the best solution (Occam's razor principle). There are functional requirements, code justification requirements and documentation requirements that must be met to receive the full mark for this assessment.

Develop your code in an iterative fashion (as opposed to completing it in one sitting). You can and should get started now.

2. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

1. Demonstrate knowledge of basic concepts, syntax and control structures in programming.

2. Devise solutions to simple computing problems under specific requirements.

3. Encode the devised solutions into computer programs and test the programs on a computer.

4. Demonstrate understanding of standard coding conventions and ethical considerations in programming.

3. Assessment details

Note: Please ensure that you have read sections 1-2 of this document before going further.

This assessment requires you to demonstrate exclusively a set of concepts and approaches covered during the weeks 1-7 (inclusive) Weekly Live Lectorials* by creating one application of your own choosing from a list of application themes provided for inspiration. Your application must adhere to constraints and requirements in this document which will help you complete this assessment.

Please follow the following steps:

1. Read and understand the functional, code+justification and documentation requirements under sections 3.1-3.3 of this document.

2. Investigate a suitable and permitted theme from the provided list that would allow you to meet the above requirements, as detailed in section 3.4 of this document. Submit the form.

3. Implement the program and get general supervision from your tutor as detailed in section 3.5 of this document. For consistency, specific clarifications can only be obtained via the Week 8 Major Assessment Project Clarifications forum from Gayan.

4. Submit as per the instructions in section 4.

* Unless otherwise stated.

3.1 Functional Requirements

When choosing an idea for the program in section 3.4, ensure that you can adapt it to meet the functional requirements below. Section 3.2 Code+Justification requirements explain how you should implement the program when meeting these functional requirements.

F1: The program is a console-based menu-driven program which allows the user to add new records and display added records.

F2: Each record must include values of at least two different data types.

F3: The program must allow the records to be saved to and loaded from files of the CSV format. The user is able to specify the names of these files via the keyboard.

F4: When an invalid input is entered, the program repeatedly asks the user to re-enter until they enter a valid input.

3.2 Code+Justification Requirements (+20 marks)

To receive marks for Code+Justification requirements, you must use the following code concepts as demonstrated during Weekly Live Lectorials and add code justification comments when meeting the 3.1 Functional Requirements.

3.2.1 Code Requirements

Note: Code without justification in the required format would attract no more than 50% of the mark allocated for that component. Comments without code will not attract any marks.

An important note on code validity: A program with 1 or more syntax errors cannot be tested and therefore will attract 0 marks. Code that produces run-time errors would incur a 50% penalty on the entire assessment's mark.

Each + point below attracts +2 marks when justified (No partial marks for a + point if only a part of it is implemented):

+ Have the program separated into exactly two modules named **backend.py** and **frontend.py** by following the logic that was explained during the 4/Oct Weekly Live Lectorial. *Tip: As we have only recently introduced classes, you are not required to use them (however, they will be a requirement in A3).

+ Functions use parameters and return values to communicate with each other.

+ Uses function annotations for all parameters and return values of every function (Refer to Canvas->Modules->3.2.2)

+ Screen outputs use only sys.stdout.write and keyboard inputs use only sys.stdin.readline

+ Follows conventions and all names are descriptive. Also applies to .py file names (e.g. do not include your name or student number in the file name)

+ List-type data structure used appropriately. Only list-type used exclusively for data structures (e.g. no dictionaries, tuples, etc.)

+ if-statements with 'elif' and 'else' for (non-repeating) conditional execution and while-loops used exclusively when repetition required (e.g. no for-loops, recursion, etc.). There must be a compound condition with either 'or' or 'and' (or both)

+ try..except used in the manner shown during 20/Sep/2021 Live Lectorial.

+ Uses 'open' to create file objects for reading and the 'readline' function is used exclusively when reading from file(s) (as shown in 11/Oct/2021 Live Lectorial).

+ Uses 'open' to create file objects for writing and the 'write' function is used exclusively when writing to file(s) (as shown in 11/Oct/2021 Live Lectorial).

Each - point below would also attract -2 marks (No partial marks for individual – points):

- For each type of restricted operator used: 'in' (containment), bitwise operators, 'is' (identity), 'not' (logical negation), shift operators, '@' matrix manipulation, slicing and similar operators not shown in Weekly Live Lectorials.

- For each unique use of the following Python keywords: await, pass, break, raise, continue, as, from, nonlocal, assert, global, with, async, yield

- Uses something other than double quotes " to enclose strings.

- For each instance of defs inside defs.

- For each instance of classes inside classes.

- Uses 'return' before the last line of a function.

- Multiple values returned from a function or values of different data types returned from a function. (Returning one list with multiple values, different types, etc. is acceptable).

- Uses exit-like operations that reduce reusability of code (e.g. sys.exit, os.exit, etc.)

- Comments added on the same line with code.

- Comments do not use professional/academic English.

- Irrelevant or unreachable code+comments included. (E.g. commented out code, non-justification comments, etc.)

- The same variable name used for storing more than one type of data. Note: The *None* value is excluded from this constraint.

- Conditions include tautologies or if/else/elif/while constructs have redundant pathways or unreachable blocks.

- Too much or too little code inside try..except or uses try..except to hide deficiencies in code (as opposed to using it to enhance the program)

- Does not close opened file objects or keeps files open longer than they need to be.

- Hard coded filenames instead of allowing the user to enter the filename at run-time.

3.2.2 Justification Requirements

Even after writing code for all code + (plus) points and not doing any code – (minus) points, you must still write justification comments as required below or you will receive at most 50% of the code mark. Justifications must be based on the rationales given in the Weekly Live Lectorial explanations by Gayan.

All justification comments must be inside your .py files and must be on a line immediately before the code that is being justified. Note that doc strings will not be accepted as justification comments.

Each variable name and its data type:

> E.g. What might be thought of as a suitable alternative name for this variable and why is what you have chosen more appropriate (or not worse)? What might another programmer argue is a more suitable data-type for the value stored in this variable and why is what you have chosen the more appropriate (or not worse)?
>
> Note/tip: The above also applies to parameters (not arguments).

Each code block:

> E.g. Is there a simpler way to meet requirements without creating this code block? What is the closest alternative to creating this code block? What statements have you kept inside the code block and why haven't you put them outside? Why are these statements in this order? What haven't you added into the code block and why?

Each condition:

> E.g. Can the same condition be written in a simpler way? If it's a if-elif-else, is there an alternative arrangement of the conditions that is better?

3.3 Documentation Requirements (-6 mark penalty if not met)

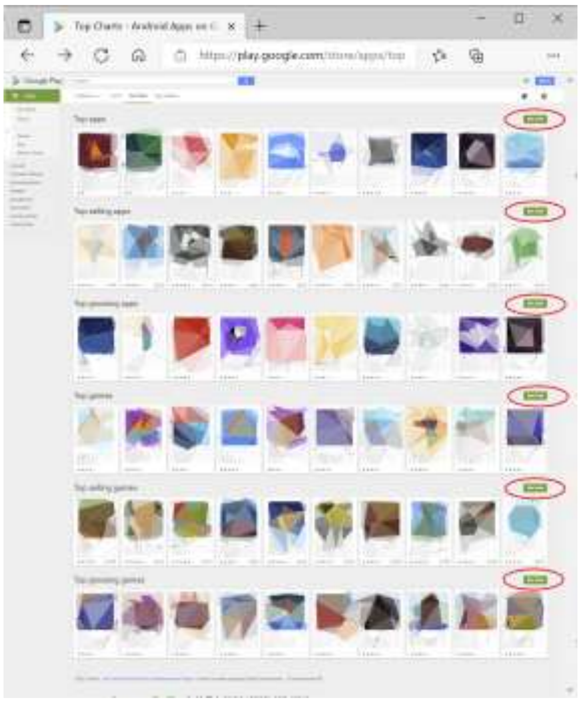
Create an illustrated user guide as a PDF (one file) with:

- Professional/academic presentation and English. Contains a table of contents which refers to numbered pages.
- Most, if not all, text must be searchable.
- Has an introduction explaining the purpose of the application
- Has screenshots of sample inputs and screenshots of corresponding sample outputs.
- Has brief instructions on what the user can and cannot do (e.g. what they can and can't input).
- Does not contain any references to code, even in backgrounds of screenshots. (This document is intended for a non-programmer who only knows how to start your program).

This documentation must match with testable, functional and justified code to attract marks.

3.4 Application Theme Requirement

When meeting the requirements given in sections 3.1-3.3, your program must draw inspiration from an application listed under "Top charts" of https://play.google.com/store/apps/top which can be accessed using any web browser by the public (you can expand the selection selecting "see more" as circled in red):


Top Charts - Android Apps on G
https://play.google.com/store/apps/top

To receive marks for your contribution/submission, your application theme must be different to any examples shown in live lessons and under Canvas->Modules.

Once you have chosen the app, **please complete the following form** (use only your @student.rmit.edu.au to login): https://forms.office.com/r/WrNtUdrEyt

Tip: You do not need to implement all of the features of the application that you have chosen from the app store; choose one, adapt and implement just enough to meet requirements stated in sections 3.1-3.2 of this document.

3.5 Receiving Supervision

As in the industry, assume that your tutor is your immediate supervisor in this project. It is recommended that you use the Canvas->Discussions->Tutor Supervision for Week 8 Major Assessment Project forum and the Weekly Assessment Help Tutorial sessions to consult your group's tutor for general supervision on this project. Your tutor will not choose a program on your behalf or debug your code on your behalf; their advice will only be general. For consistency and fairness to all students, your **tutors will not look at your assignment code.** Instead, you should aim to explain your issues to your tutor in the context of IIEs to receive the necessary help.

If you have a question in the format of "can I do X" or "do we need to do X", you are required to obtain written clarification from Gayan from Canvas->Discussions->Assignment 2 Clarifications forum.

4. Submission

The Canvas->Assignments->Week 8 Major Assessment Project submission system is used by the university for official records of your submitted work. It is the responsibility of the student to ensure that all of the following files are chosen and submitted, each time the 'submit' button is selected. (E.g. submit all required files in one go, do not submit one file after another).

1. **backend.py** **and** **frontend.py** (only these files should contain code and justification comments).

2. **user_guide.pdf** containing the user guide documentation (must not contain any references to code).

3. Relevant **.csv** file(s) with sample data (Do not submit empty files).

Bonus marks: Submit your final submission 1 day before the deadline to qualify for +0.1 bonus marks, or 2 days before the deadline to qualify for +0.2 bonus marks, etc. To qualify any bonus marks, your submission must obtain full marks for the non-bonus component. The total mark for the course is capped at 100.

Important note: No .py files or a .py file with even one syntax error means no mark even if other files are submitted. Please test your code thoroughly before submission.

4.1 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 (standard warning)

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. Document revision history

2021.10.15: Original version.