Student Name: **Weight:** **15%**

Student ID: **Marks:** **/100**

Assignment: Functions, Scoping and Abstraction

Type: Group Assignment

Needed Modules: 1, 2 and 3 ONLY

- Students should **ONLY use** programming constructs covered in modules 1, 2 and 3.
- **Submissions using programming concepts that are not covered in modules 1, 2, and 3 will be penalized.**
    - **Penalty with no limit could be applied.**
- **Late submissions will not be accepted.**

Scenario: Students Registration Program

An educational institution asked you to a develop students registration program. The program allows the following features:

- Storing and loading students' information into/from a file
- Adding a new student.
- Displaying a list of registered students.
- Editing a registered student.
- Searching for a registered student.
- Deleting a registered student.
- Calculating GPA average of the registered students.

Equipment and Materials

For this assignment, you will need:

- Python IDE

Instructions

- This assignment will need to be completed outside of class time. See the course schedule and Brightspace for exact due dates.
- There are many details involved in this assignment which could be challenging to get sorted out and working correctly. Therefore, it is strongly recommended to work on the assignment as early as possible and in Team.
- The program MUST be implemented using functions.

  - You MUST use the provided functions names.
  - You MUST implement the functions as specified.
- The program MUST use files to store/retrieve students' information.
- It is also recommended to equally divide the functions between the group members.
- You MUST use the provided TXT file (i.e., students.txt)
- You MUST exactly implement the provided sample runs.
  - You MUST use the values provided in the sample runs.
  - All messages and the program menu MUST be exactly as given.
  - GPA average MUST be formatted as given.

Group Submission

- Divide the assignments into tasks and assign these tasks to members.
  - Mainly, equally divide the functions between group members.
  - Each member is responsible for implementing and testing his/her assigned functions and ensuring it is working properly.
  - Members should review the work completed by other members.
- Integrate all the functions and test the whole program to ensure that the program meets all the requirements.
- Check your solution against the detailed marking criteria at the end of this document or posted by Bright Space.
- Submit this final version of the code as a group. Only one copy is required per group, and any of the group members may submit the following to Brightspace:
  - The code of the program that you implemented (.py file).
  - A doc file for screenshots/text of the test output.
  - A PDF file peer assessment document.

Program Requirements

- The program MUST use functions to ensure the usability of the code.
- Students' information is loaded from the students text file when starting the program.
- The program should continue displaying a menu of options until the user quits the program.
  - Review the provided sample runs.
- The user chooses a menu option by entering a small or capital letter.
- When adding a new student, deleting a registered student, or editing a registered student, the file students.txt should be updated.
- The program should not allow adding a student who is already registered.
- The program should not allow deleting/editing a student who is not registered.
- The program allows editing all student's information including student id.
- Searching for, editing, or deleting should be done using student id.
- The program should have a minimal code redundancy.
  - It is permitted to implement functions which are not provided to reduce the code redundancy.
- The program should calculate the GPA average using the updated file contents.

- **The program must implement the following functions:**

Menu Option	Function	Description
	1- print_menu()	• It does not have parameters. • It displays the welcome message and the menu options. • It asks the user to select a menu option and returns user' selection.
	2- format_student()	• It receives a list of a student's information. • It returns a CSV formatted string of the student's information. ○ Example is "111,John Smith,99.0"
	3- display_std_header()	• It does not have parameters. • It displays the student header which includes Student ID, Student Name, and GPA. • It is used when listing students and finding a student.
	4- display_student()	• It receives a student information. • It displays the report/student header and the student information.

L/l	**5- list_students()**	• It receives a list of students' information.
		• If the received list is empty, it will display "Students list has no students" error message.
		• Otherwise,
		  ○ It calls display_std_header() to display the student header.
		  ○ It iterates over the students list and display their information.

A/a	**6- add_student()**	• It receives the file name where students' information is saved and the list of students.
		• It asks the user to enter the student id.
		• It checks whether the student exists or not
		  ○ If the student exists, it displays the students already exists.
		  ○ If the student does not exist, it will ask the user to enter the student information (name, GPA), append these information to the student list, and update the students text file.

E/e	**7- edit_student()**	• The program should check whether the student exists or not before calling this function. • It receives the student ID and the students list. • It iterates over the list of students until finding the student. • Once, the student is located in the list, it asks the user to enter student name and GPA, then updates the student record in the list. • After executing this function, the program should update the students text file.

D/d	**8- delete_student()**	• The program should check whether the student exists or not before calling this function. • It receives the student information and the students list. • It deletes the student from the students list. • After executing this function, the program should update the students text file.
F/f	**9- find_student()**	• The program should check whether the student exists or not before calling this function. • It receives the student ID and the students list. • It iterates over the list of students until finding the student and returns the student information. • After executing this function, the program should display the student information if the student exists.
G/g	**10- calculate_average()**	• It receives the students list. • It processes the students list to extract the GPA for each student and calculate the GPA average. • It returns the average rounded to 2 decimal digits.
File Handling Functions		
	11- load_students()	• It receives the text file name (i.e., students.txt) • It reads students information and load it into a dictionary. • It returns the student's dictionary. • Hint: the student dictionary is a 2-dimensional dictionary. • The key of each dictionary is the student ID, and the value is a dictionary of the student information.
	12- update_students()	• It receives the text file name (i.e., students.txt) and student's dictionary. • It reads students dictionary and format the student's information to CSV format. • Finally, it writes the formatted students information into the students text file.
	13- main()	• The program execution starts by executing this function. • It asks the user to enter the students text file. • It will ensure that the entered file exists before reading the file content and load it to the student list. • It displays the program menu. • It checked the user selection and accordingly call the appropriate functions.
Q/q	exit the program.	

- The solution will require 2-dimensional dictionary to store students information.

Test Plan

Attached in a separate file named: “Assignment 3 - Sample Run.pdf”

Peer Assessment

- Each member should assess the contribution/participation of all other group members.
- Each member should assign a mark out of 10 to other members.
- Ensure that the task description is clear and accurately reflect the actual participation. Otherwise, expect deduction.
- Please use the following table to complete the peer assessment. Marks/tasks are just samples.

Member/Reviewer	Member 1 Name	Member 2 Name	Member 3 Name	Completed Tasks
Member 1 Name	N/A	9	10	• Task 1 description • Task 2 description
Member 2 Name	8	N/A	9	• Task 3 description • Task 4 description
Member 3 Name	9	10	N/A	• Task 5 description • Task 6 description
Average	8.5	9.5	9.5	

A student can receive a deduction on their individual grade for the assignment if they do not meet the expected contribution/participation based on the feedback of other group members.

Grading

Item	Percentage
Code	65
Code Style	15
Test Run	15
Peer Assessment	5
Total	100

Marking Criteria

Rubric available on Brightspace.