420-LCU-05 Programming in Python - Assignment 2
Due Feb 26, 2021 at 11:59 p.m.

Identification section: This section must be either in a comment, with a '#' preceding each line, or enclosed within triple quotes ('''). The grader and I need this section for the accurate processing of your assignment. Assignments missing this may lose up to 5% of the mark.

```
"""
Your Name and ID
420-LCU Computer Programming, Section #
S. Hilal, instructor
Assignment 2
"""
```

Submission: Submit your assignment in 1 Python file, with the extension .py. Please do not create a ZIP file. Be sure to respect other instructions specified in the assignment. An important part of each assignment is to correctly follow the instructions closely. **Late assignments** are accepted up to 1 week from deadline. But late penalty will be applied.

Learning Objectives:

- Practice using lists, strings, loops and Multidimensional (nested) lists.
- Practice with built-in functions and methods for strings and lists.

A Simple Class Calculator

For this assignment, you will develop a small application that will enable a teacher to **enter**, **analyze** and **report** on the grades of the students in the class. The teacher can enter students' grades in different components of the course. The program can calculate a student's **overall grade**, **letter grade**, in addition to some overall class statistics.

Description of Data:

- A student is identified by a name (first name only) and a 7-digit ID number. For each student, there are 4 grades based on 2 tests and 2 assignments.
- Each test and assignment is marked on 25 and is worth 25% of the total grade for a total of 100%.
- Each grade is entered as an integer or float value.
- Your program should be able to process the information for any number of students but you can limit your tests to 8 students.

Description of Program:

Data Table: Your program defines a 2-dimensional list (list of lists). **list_of_students** = [] which will hold the data for all students. Each list element holds the data for one student. Note that the list is initialized as an empty list and will be filled in by the program.

Data Entry: Your program asks the user to enter the information for each student (student record). The program verifies and stores the information in **list_of_students**.

Important: When a student record is entered, your program should calculate the total grade and letter grade and store them (append them) on the student list. The student list is then added to the list_of_students.

Here is an example of how the list_of_students should look like after entering the information for 3 students:

```
list_of_students =    [["Lea", 1234567, 20, 25, 22, 25, 92, 'A'],
                       ["Bob", 2345678, 22, 25, 21, 18, 86, 'B'],
                       ["Ben", 3456789, 25, 20, 22, 22, 89, 'A']]
```

The list of each student stores the data in the following order: student's name (string), ID (an integer), 2 test scores (out of 25 each) and 2 assignment scores (out of 25 each). **All test and assignment scores should be stored as floats**.

The program menu **Display exactly as shown here in your code.**

Welcome to the Simple Class Calculator. Here's the list of available options:

1- Enter a student record (Name, ID, and 4 grades separated by commas and no spaces).
2- Display the full class data sorted in ***alphabetical order*** based on ***name***.
3- Display the class descriptive statistics.
4- Display a chosen student record.
5- Exit

Select an option by entering its number or 5 to exit:

Running Your Program:

Your program starts by presenting the user with the above **menu** of numbered options. A user selects an option by entering the corresponding number. When the program completes processing an option, **the menu is displayed again** and the user is prompted to make another selection. This continues until the user selects the option **to exit**.
Important: Your program must ensure that the user enters a valid option and that a requested option can actually be processed. E.g. the program cannot process option 2 if **list_of_students** is empty.

Include other validation tests that you see necessary and add comments and appropriate error messages.

Description of the Different Options (above):

For each option, the program output is shown in bold. User input is shown in red

1- **Option 1**: The program asks the user to enter a student record.

Here are 6 examples for option 1. Important Note: After each example, the main menu is displayed and the user selected option 1 again. Not shown here.

1- **Enter Student Record (Name, ID, and 4 grades separated by commas and no spaces):** Lea,1234567,20,25,22,25

Record Accepted

2- **Enter Student Record (Name, ID, and 4 grades separated by commas and no spaces):** Bob,2345678,22,25,21,18

Record Accepted

3- **Enter Student Record (Name, ID, and 4 grades separated by commas and no spaces):** Ben,3456789,25,20,22

Record Incomplete, Record rejected.

4- **Enter Student Record (Name, ID, and 4 grades separated by commas and no spaces):** Ben,34567,15,18,12,15

ID must have 7 digits. Record rejected.

5- **Enter Student Record (Name, ID, and 4 grades separated by commas and no spaces):** Bob,2222222,25,25,22,21

Record Accepted

6- **Enter Student Record (Name, ID, and 4 grades separated by commas and no spaces):** Joe,2222222,20,19,22,22

Duplicate ID number. Record rejected

Here are some additional requirements:

- There may be duplicate names but IDs are unique.
- The program rejects an incomplete record (missing any of the 6 data items). Just check count.
- For each student record, create a **student_list** to hold the data for 1 student
- When a valid record is entered, calculate total and letter grades and store on the **student_list**
- The complete **student_list** is then added to the **list_of_students**.

2- **Option 2**: Display the full list in alphabetical order. Include all information entered in addition to total and letter grades.

3- **Option 3**: Class descriptive statistics are:
   - Number of students entered
   - Class average
   - Average grade for each test and each assignment.
   - Grade distribution (how many of each letter grade)
   - Number of students above (or on) the average.
   - Number of students below the average.

   Your program prints each item as above with the text and value on the same line.

4- **Option 4**: The program asks the user to enter the ID of a student. The program prints the full record of the requested student.

   Enter the ID of a student: 1234567

The program prints:

Name: Lea ID: 1234567

Test Grades: 20, 25 Assignment Grades: 22, 25

Total Grade: 92 Letter Grade: A

Table of the letter grades that correspond to the total score:

Total Grade	**Letter Grade**
87 or above	A
From 75 to 86 inclusive	B
From 65 to 74 inclusive	C
Below 65	F

Writing your program:

1- You can use any of the list/strings built-in functions and methods that we have seen in class.
2- The program will display the main menu following the completion of each option and until the user selects option 5.

Testing Your Program:

The program does not store any data between runs. It is a good idea that you create a few sets of data that you can use to test your program and always use the same data. You can store your data in a comment at the beginning of your program such that you can copy and paste the data for input rather than typing each time. The more records you use to test the better. The records I used for my tests are provided below for you. Feel free to add more.

#Lea,1234567,20,25,22,25
#Bob,2345678,22,25,21,18
#Ben,3456789,25,20,22
#Ben,34567,15,18,12,15
#Bob,2222222,25,25,22,21
#Joe,2222222,20,19,22,22