Menu-Driven Program for Personal Academic Records, AcaRec

Instructions

- You'll be given four files to complete this term project which are:
    1. `AcaRec.py`
    2. `AcademicModule.py`
    3. `ReadWriteModule.py`
    4. `academicrecords.csv`
- You'll zip all related files into a single file called termproject.zip and you must upload this one zip file to the Microsoft Teams Class Assignment before the due date.
- At the beginning of your Python files, write down your student id, and name as comments. As an example, each of the Python files should look like this:

```
# Id: 5919449
# Name: Ada Lovelace

... your program continues here ...
```

Problem Statement

You have to write a program to build a menu-driven program for personal academic records for an individual student, named AcaRec. The brief system architecture is shown in Figure 1.

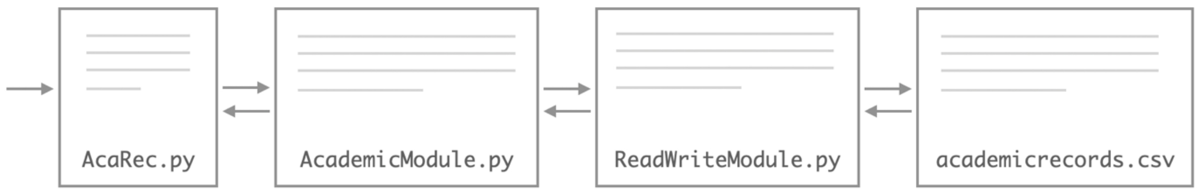

Figure 1: AcaRec system architecture

The `AcaRec.py` is a main program where the user can interact with. The `AcaRec.py` provides the console interface as a menu-driven program. You can see the example of starting console interface in Figure 2. The main program must not contain any functionality logics, but you must import and use the functionality logics which defined in `AcademicModule.py`.

```
AcaRec Version 1.0
PERSONAL ACADEMIC RECORDS
---- ---- ---- ---- ---- ---- ----
1. Add new academic record
2. Remove a specific academic record
3. Display academic records by semester
4. Display academic records by grade result
5. Display the number of courses for each grade results
6. Update a grade result to a specific academic record
7. Calculate the overall G.P.A. and total credits
0. Exit the program
---- ---- ---- ---- ---- ---- ----
Enter option (1-7, 0 to exit): _
```

Figure 2: Starting console interface

The menu-drive program concept allows the user to select the option that he/she wants to perform and once the selected option performed completely, the program continues ask the user to select the next option over and over, as shown in Figure 3. The program stops if the user selects 0 option. The personal academic records information will be stored as a file in `academicrecord.csv`. You can see the example of the information structure there. There are six main functionalities you have to write as a method module in `AcademicModule.py` which are:

1. **Add new academic record**

   This method takes academic record information from the user; academic year, academic term, course code, section number, and grade result. Before adding the new academic record to the list, the existing academic records must not contain the adding academic record's course code in the same semester. The given information will then be added to the end of the list in academicrecords.csv. The example of this process shown in Figure 3.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 1
---- ---- ---- ---- ---- ---- ----
Enter academic year: 2020
Enter academic term: 1
Enter course code: CSX3001
Enter section number: 541
Enter grade result: A
Enter course credit: 3
CSX3001 added.
---- ---- ---- ---- ---- ---- ----
Enter option (1-7, 0 to exit): _
... program output message continues ...
```

Figure 3: Add new academic record

If the input's course code does already exist in the given semester, the program displays the error message to the user as shown in Figure 4.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 1
---- ---- ---- ---- ---- ---- ----
Enter academic year: 2020
Enter academic term: 1
Enter course code: CSX3001
Error: CSX3001 already exists in 1/2020!
... program output message continues ...
```

Figure 4: Error message adding existing course code

2. **Remove a specific academic record**

   This method removes an academic record by using a course code as a key. The example of this process shown in Figure 5.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 2
---- ---- ---- ---- ---- ---- ----
Enter course code: CSX3001
CSX300 removed.
... program output message continues ...
```

Figure 5: Remove a specific academic record

If the input's course code does not exist in the file, the program displays the error message to the user as shown in Figure 6.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 2
---- ---- ---- ---- ---- ---- ----
Enter course code: CSX3001
Error: CSX3001 does not exist in the file!
... program output message continues ...
```

Figure 6: Error message removing course code that does not exist

3. Display academic records by semester

This method allows the user to see his/her academic records by semester. The displayed semester order must also sort in the order of latest to the past. The example of this process shown in Figure 7.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 3
---- ---- ---- ---- ---- ---- ----
Semester 2/2019
BG14038 (564)     S
CS4409  (541)     A

Semester 1/2019
BG14037 (508)     S
CS3414  (541)     A
CS3436  (541)     A
CS4200  (541)     A
CS4402  (541)     A
HTM4112 (541)     A

Semester 2/2018
BG14036 (508)     S
CS3200  (541)     A
CS4401  (541)     A
CS4413  (541)     A
LA4606  (541)     B+
MT4323  (541)     A
... program output message continues ...
```

Figure 7: Display academic records by semester

If there are no academic records added to the file, the program displays the error message to the user as shown in Figure 8.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 3
---- ---- ---- ---- ---- ---- ----
Error: No academic records found!
... program output message continues ...
```

Figure 8: No academic records in the file

4. **Display academic records by grade result**

   This method allows the user to see his/her academic records by grade result. The user must enter the grade result. The program will then lookup any academic records where grade result matched. The example of this process shown in Figure 9.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 4
---- ---- ---- ---- ---- ---- ----
Enter grade result to lookup: B-
DA2103  (541)     B-
CS2201  (541)     B-
... program output message continues ...
```

Figure 9: Display academic records by grade result

If there are no academic records with the given grade result, the program displays the error message to the user as shown in Figure 10.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 4
---- ---- ---- ---- ---- ---- ----
Enter grade result to lookup: C
Error: No academic records found for C grade!
... program output message continues ...
```

Figure 10: No academic records found for given grade result

5. **Display the number of courses for each grade results**

   This method allows the user to see the number of academic records for each grade result. The example of this process shown in Figure 11.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 5
---- ---- ---- ---- ---- ---- ----
Grade A     26 course(s)
Grade A-     6 course(s)
Grade B      2 course(s)
Grade B-     2 course(s)
Grade B+     5 course(s)
Grade S      8 course(s)
... program output message continues ...
```

Figure 11: Display academic records by grade result

If there are no academic records added to the file, the program displays the error message to the user as shown in Figure 12.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 5
---- ---- ---- ---- ---- ---- ----
Error: No academic records found!
... program output message continues ...
```

Figure 12: No academic records in the file

6. **Update a grade result to a specific academic record**

This method allows the user to update the grade result for a specific academic by the given semester and course code. The example of this process shown in Figure 13.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 6
---- ---- ---- ---- ---- ---- ----
Enter academic year: 2020
Enter academic term: 1
Enter course code: CSX3001
Enter grade result: D
Updated grade result 1/2020 on CSX3001 as D
... program output message continues ...
```

Figure 13: Update a grade result

If there is no academic record with the given semester and course code, the program displays the error message to the user as shown in Figure 14.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 6
---- ---- ---- ---- ---- ---- ----
Enter academic year: 2020
Enter academic term: 1
Enter course code: CSX3001
Error: CSX3001 not found for 1/2020!
... program output message continues ...
```

Figure 14: No academic records found in specific semester and course code

7. Calculate and display the overall G.P.A. and total credits

This method allows the user to check the overall G.P.A and total credits that he/she earned so far. The example of this process shown in Figure 15.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 7
---- ---- ---- ---- ---- ---- ----
G.P.A. is 3.77
Total Credit is 126
... program output message continues ...
```

Figure 15: Overall G.P.A. and total credits

If there are no academic records to be calculated, the program displays the error message to the user as shown in Figure 16.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 7
---- ---- ---- ---- ---- ---- ----
Error: No academic records found!
... program output message continues ...
```

Figure 16: No academic records to be calculated

The G.P.A. can be calculated from the following formula. The calculation must not include any courses that has 0 credit (*e.g.* BG14031-8).

$$\frac{\sum_{c=0}^{n} Grade\ Result\ Weight_c\ x\ Credit_c}{Total\ Credits}$$

The grade result's weights are defined in Table 1.

A	4.00
A-	3.75
B+	3.25
B	3.00
B-	2.75
C+	2.25
C	2.00
C-	1.75
D	1.00
F	1.00

Table 1: Grade result's weights

The AcaRec will stop when the user select option 0, the example is shown in Figure 17.

```
... previous program output message ...
Enter option (1-7, 0 to exit): 0
---- ---- ---- ---- ---- ---- ----
AcaRec Stopped. See you next time!
```

Figure 17: AcaRec Stopped

We also provide you the `ReadWriteModule.py` to read/write the specific csv file. This module contains two main methods which are:

1. **Read the file**

   This method reads the information from the given file's name. The result of this method is in the format of `[[String]]`. For example, if the file that you want to read contains the following information:

   ```
   2015,3,CS4408,541,A,3
   2016,1,DA1121,541,A,3
   2016,1,GE2101,411,A,3
   ```

   the result from this method will be

   ```
   [ [ "2015", "3", "CS4408", "541", "A", "3"],
     [ "2016", "3", "DA1121", "541", "A", "3"],
     [ "2016", "3", "GE2101", "411", "A", "3"] ]
   ```

2. **Write the file**

   This method writes the given records to the given file's name. The given records must be in the format of `[[String]]`, same output's format when you call read the file method. This write method will entirely replace the existing file's information with the given records.

End of Term Project