Homework 10 is an Python EXTRA CREDIT homework that counts for 5% extra credit in your overall grade.

This homework is **due Friday May 4th, by 6pm**

- **Develop in Cloud9:** For this assignment, write and test your solution using Cloud9.
- **Submission:** All components must be fully completed by the submission deadline for your homework to receive points. Partial submissions will not be graded.
    1. Make sure your Cloud9 workspace is shared with your TA: Your recitation TA will review your code by going to your Cloud9 workspace. TAs will check the last version that was saved before the submission deadline.
        - Create a directory called *Hmwk10* and place all your file(s) for this assignment in this directory.
    2. **Submit to the Moodle Autograder:** Head over to Moodle to the link *Hmwk10*. You will find one programming quiz question for each problem in the assignment. You can modify your code and re-submit (press Check again) as many times as you need to, up until the assignment due date.
    3. **Submit a zip file to Moodle:** After you have completed all the questions and checked them on Moodle, you must submit a zip file with the .py file(s) you wrote in Cloud9. Submit this file going to *Hmwk10 (File Submission)* on moodle.

Points for Homework 10

- 100 points from Moodle autograder

Library System

Homework 10 will require you to re-implement Homework 7 in Python. There will be no differences for someone interacting with the program as an end user, although there will be significant differences in the implementation.

You will be implementing a Library system similar to the ones you see on Netflix, Amazon, or Barnes & Noble. Your Library class will read a set of reader ratings for a set of books and provide functions/methods to analyze the data and suggest new books for a given reader.

As in Homework 7, you will need to put all your knowledge together into solving a real world problem. We will provide you with a test data set with which you can test your code, but your grade for your assignment will be based on processing a different data set. You will not know the exact data being used, but will be told the purpose of each test case and the outcome.

For this assignment, you need to submit one file called ***homework10.py***.

You are not required to write test cases for this assignment.

Data Files used in your Library System

There are two files that are used in this assignment: a list of books (books.txt) and a list of ratings from readers (ratings.txt). There are ratings associated with each book in the books file for each reader. The order of the ratings by an individual reader is the same as the order of the books in the books.txt data file.

The input files will always be named books.txt and ratings.txt. (i.e., You can hardcode these names into your program.) Load your book data and user data from these files.

Format of books.txt

Every non-empty line of **books.txt** will have the following format:

```
<Author>,<Title>
```

For example:

```
Douglas Adams,The Hitchhiker's Guide To The Galaxy
Richard Adams,Watership Down
Mitch Albom,The Five People You Meet in Heaven
Laurie Halse Anderson,Speak
Maya Angelou,I Know Why the Caged Bird Sings
```

Book data should be case insensitive. Convert all titles and author names to lowercase before assigning them to the member data for the Book objects. For example:

```
Douglas Adams,The Hitchhiker's Guide To The Galaxy
```

should become

```
douglas adams,the hitchhiker's guide to the galaxy
```

Format of ratings.txt

Every line of ratings.txt will have the following format: a row with the user's name, followed by a comma, followed by this user's ratings for all books.

Username and ratings are separated by comma, and each rating is separated by a single space.

```
<name>,<book_1_rating> <book_2_rating> <book_3_rating> ...
```

The number of ratings a user has will always be equal to the number of books. For example, if there were 5 books in books.txt, every user will have 5 ratings in ratings.txt.

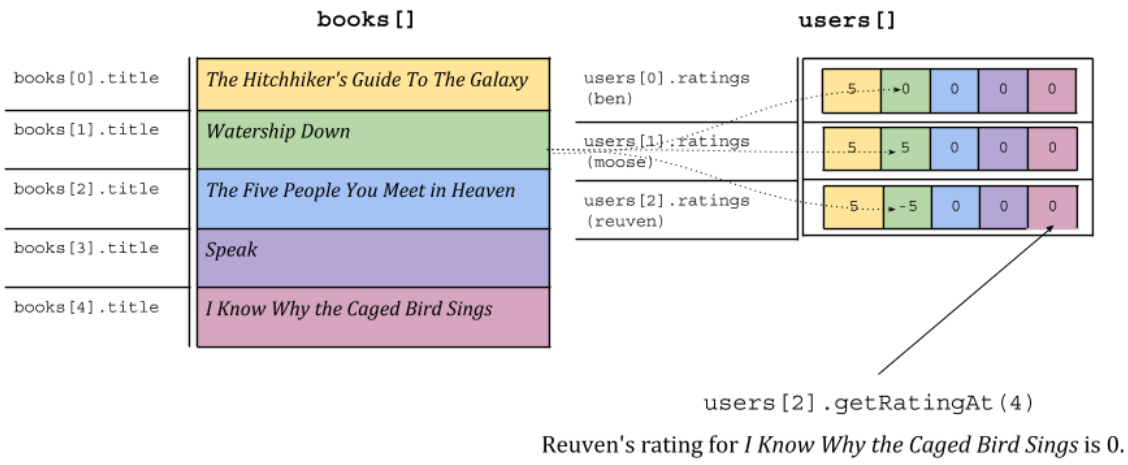
Convert names to lowercase before creating the User object. For example, "Ben" should become "ben".

For example:

The **ratings.txt** will look like:

```
Ben,5 0 0 0 0
Moose Potter,5 5 0 0 0
Reuven ravenclaw,5 -5 0 0 0
```

The i^{th} rating for each user in **ratings.txt** corresponds to the i^{th} book in **books.txt**. In this example, a user's second rating is always for the *Watership Down.*



Reuven's rating for *I Know Why the Caged Bird Sings* is 0.

Rating System:

Rating	Meaning
0	Haven't read it
-5	Hated it
-3	Didn't like it
1	Ok. neither hot nor cold about it
3	Liked it!
5	Really liked it!

Create a class *Library*

Each part of the remaining writeup depends on methods from the previous step working correctly.

This strategy of writing a little bit of code and testing it, then adding more functionality and testing it, is known as a ***bottom up*** implementation strategy. It builds the base (most

fundamental) functions and methods first, then adds new layers of complexity that takes advantage of the methods already tested. By implementing and testing each layer independently, you isolate the possible places where problems can occur and makes the debugging process easier.

Final Note

When writing your functions and class methods, you will need to write code that tests those methods. You can place your testing code anywhere in your python source file, however we suggest that you create a "my_tests()" function similar to how you added your test code into a main function in a C++ source file. Although Python does not explicitly call that function for you, you can mimic that behavior by placing a call to "my_tests()" at the bottom of your source file.

Library Class

You need to create a class ***Library*:**

Instance Variables:

- **book_list :** list to store the books read from file.
- **user_dictionary:** a python dictionary which stores the users and their ratings for the books.

Class Methods

- **constructor:** provide a constructor that takes two filenames (books_filename, ratings_filename) as parameters and creates the required dictionaries and lists as class data members.
- ***read_books(file_name)*** method should add entries to a member list variable for books from the file.
- ***read_users(user_file_name)*** method should add entries to a member dictionary variable for ratings from the file.

  After the data is loaded, print the following:
  `Data Loaded successfully!`

You will also provide other methods as described below to provide the recommendations

of books to read for a given user.

Part A

Write a method ***login()***. After the data is loaded, welcome the user to the library and ask them to enter their name

```
Welcome to the Library, What is your name?:
```

If a user has interacted with the system before, the system should remember the user's previous ratings.

```
Data Loaded successfully!
Welcome to the Library, What is your name?:
Ben
Welcome back, ben
```

Login should be case insensitive. Convert all logins to lowercase before searching for the user. If a user "ben" or "Ben" was in the array of users, all of the following logins should be successful:

```
Ben, BeN, BEN, ben etc.
```

Keep in mind the user might make mistakes. Your program should account for user error at any step. For example, if upon asking the user to entering their name, the user accidentally presses ENTER, your program should recognize the fact that the input is basically “empty” and prompt the user to enter their name again.

```
Data Loaded successfully!
Welcome to the Library, What is your name?:

You provided an empty username, Please provide a valid user name to login or register
Enter your name again:
AlBus DumbledOrE
Welcome back, albus dumbledrE
```

If it is a user's first time interacting with the system, they should be added to the user dictionary of the class. As the user hasn't rated any book yet, the ratings for all the books will be 0.

```
Data Loaded successfully!
Welcome to the Library, What is your name?:
Anya
Welcome to the Library, anya
```

Part B

Write a method ***menu()*** which provides the user a menu of options. The menu should look like the following:

After the name is entered, the following options should be provided:

```
Would you like to (v)iew your ratings, (r)ate a book, (g)et
recommendations, or (q)uit?:
```

Part C

Write a function, ***view_ratings(current_user_name)*** which takes in the logged in user name and prints all the books and their ratings which have been read by the user. Recall that a rating of 0 means a user has not rated that book and hence shouldn't be displayed. Here is an example of how to display the user's rating values:

```
Here are the books that you have rated:
Title : the hitchhiker's guide to the galaxy
Rating : 5
------------------------------
Title : the sisterhood of the travelling pants
Rating : 1
------------------------------
Title : the da vinci code
Rating : 1
------------------------------
Title : the princess diaries
Rating : -3
------------------------------
Title : ender's game
Rating : 5

...
```

Part D

Write a function ***rate_book(current_user_name, book_name, rating)*** which takes in the logged in user name, the book name that user wants to rate and the rating of that book.

If the user chooses to rate a book, the user will be prompted first to enter the title of the book. Keep in mind that we will treat the used input as case insensitive. All of the following user input values should be successful:

- Ender's Game
- ender's game
- ENDER'S GAME
- Ender's GAME

If the title of the book already exists in the system, then the user will be asked to provide the rating value. If the title does not exists, prompt the user to enter a new title. Even if the user has already rated a book, they should be able to enter a new rating value. This will

result in an update of the ratings array for the user. **Note:** You might want to make a helper function to find the index of the book.

Here is an example of the dialog with the user.

```
Would you like to (v)iew your ratings, (r)ate a book,
(g)et recommendations, or (q)uit?:
r
Enter the name of the book that you would like to
rate:
The Princess Diaries
Enter your rating of the book:
3
Success!
Thank-you. The rating for the book titled "the
princess diaries" has been updated to 3
```

If the book doesn't exist in your Library, or if the user presses ENTER accidentally, then print an error message, like below:

```
Would you like to (v)iew your ratings, (r)ate a book, (g)et
recommendations, or (q)uit?:
r
Enter the name of the book that you would like to rate:
Wings of Fire
Enter your rating of the book:
5
The book you requested does not exist in the database
```

Part E

Write a **Helper method *calc_similarity(user1, user2)*** which takes 2 user ids to find and

return the similarity between all of the ratings of the two given users. This method is similar to the method used in the DNA assignment, as it compares the individual ratings from the two users to determine the overall similarity between two users.

The similarity between two users is calculated by treating each of their ratings as a mathematical vector and calculating the dot product of these two vectors. *(Remember that the dot product is just the sum of the products of each of the corresponding elements. See the example below.)*

$$SSD = \sum_i (A_i - B_i)^2$$

For example, suppose we had 3 books and ratings as follows:

```
Terry 5 3 1
Bob 5 1 5
```

The calculation for similarity between Terry and Bob will be:

- Terry and Bob: $(5 - 5)^2 + (3 - 1)^2 + (1 - 5)^2 = 0 + 4 + 16 = 20$

Part F

Write a function ***get_most_similar_user(current_user_name)*** which takes a ***current_user_nme*** and returns the user_name of user whose similarity score with the ***current_user_name*** is the lowest. This method will look through the other users in the dictionary to find the user whose ratings are the most similar to the ***current_user_name***'s ratings. **If 2 users have same (lowest) similarity score, return the first one.**

For example, suppose we had 3 books and ratings as follows:

```
Terry 5 3 1
Bob 5 1 5
Tracey 1 5 3
Kalid 1 3 0
```

The calculation for similarity between Terry and all other users will be:

- Terry and Bob: $(5 - 5)^2 + (3 - 1)^2 + (1 - 5)^2 = 0 + 4 + 16 = 20$
- Terry and Tracey: $(5 - 1)^2 + (3 - 5)^2 + (1 - 3)^2 = 16 + 4 + 4 = 24$
- Terry and Kalid: $(5 - 1)^2 + (3 - 3)^2 + (1 - 0)^2 = 16 + 0 + 1 = 17$

Once you have calculated the pairwise similarity between Terry and every other user, you can then identify whose ratings are most similar to Terry's.
In this case, Kalid is most similar to Terry.

Part G

Write a function ***recommend_books(current_user_name)*** to find a set of recommendations of new books to read for a given user name. The function will print a set of recommendations in a list.

This function should recommend at most 10 recommended books. If there are less than 10 books to recommend, recommend as many as possible. So user will be presented with 0 to 10 recommendations.

This method should use the methods already described above to find the most similar user to the ***current_user_name.*** The method will recommend all the books that the similar user has rated as a 3 or 5 that the ***current_user_name*** **has not read yet.** Remember, if a user has not given any rating for a book that means they have not read that book yet.

For the data currently in ratings.txt, there are no books to recommend for cust1:

```
Data Loaded successfully!
Welcome to the Library, What is your name?:
cust1
Welcome back, cust1
Would you like to (v)iew your ratings, (r)ate a book, (g)et
recommendations, or (q)uit?:
g
There are no recommendations for you at present
```

For the data currently in **ratings.txt**, there are five book recommendations for Jim C:

```
Data Loaded successfully!
Welcome to the Library, What is your name?:
Jim C
Welcome back, Jim C
Would you like to (v)iew your ratings, (r)ate a book, (g)et
recommendations, or (q)uit?:
g
Here are some of the books that we think you would like
nineteen eighty-four (1984) by george orwell
harry potter series by j.k. rowling
bone series by jeff smith
the war of the worlds by h g wells
the chrysalids by john wyndham
```

Submitting to the Autograder:

Because of the current format of moodle, you will need to put all your code together in the answer box.