Project Python

Book recommendation system

1. Instructions & General information:

-This project is to be carried out exclusively in Python language.

-Attachments about the project:

·Initial list of books in the books.txt repository

·For part III: the readers.txt and booksread.txt files

·For part III: The scoring matrix corresponding to the files provided

-Final rendering: A .zip archive containing

·The project code containing the .py and .txt files

·The report in .pdf

·A README.txt file listing the programs and how to use them in practice. This file must contain all the instructions necessary for execution; it is very important in explaining to the user how to use the tool.

-Code organization:

The notation of the code will mainly take into account:

·The implementation of the requested functionalities: make the best possible progress but NOT off topic

·Books, various courses on the web. PLAGIATE ALERT !! it is not about copying entire programs.

·Efrei teachers during project monitoring sessions

1. Preamble

The goal of this project is to create a computer tool allowing to suggest (recommend) to readers books according to their profiles and their previous readings.

To achieve this goal, this project is divided into 3 parts:

- Partie 1 : Managing reader profiles

· Add reader

· View a reader

· Modify a reader

· Remove a reader

- Part 2: Managing the Book Depository

· Display the list of books in the repository (initial list provided in the books.txt file)

· Add a book to the repository

· Change the title of a book in the repository

· Remove a book from the repository

- Part 3 : Recommend a book

· Rate a book

· Suggest books

Launching the program must therefore display a menu to the user to choose between 3 functionalities which represent each of the parts described above:

① Reader Profiles(Profils des lecteurs)

② Visit the books repository(Visiter le dépôt des livres)

③ Recommendation (Recommandation)

1. Part I: Reader profiles (8 pts)

The parameters related to a reader's profile will be at a minimum (you can add others depending on the utility and your imagination):

· The pseudonym

· Gender

Male(HOMME),

Female(FEMME)

No Matter(PEU IMPORTE)

· Age

<= 18 years old

Between 18 and 25 years old

> 25 years

· Reading style

Science-fiction

Biographie

Horreur

Romance

Fable

Histoire

Comédie

· The list of books read among those present in the repository

All this information will be stored in lists and files to be able to assign them to profiles and / or display them to users.

**Representation and storage of reader profiles**

All reader profiles will be stored in two text files containing one line per registered reader:

· In the readers.txt file, the reader will be stored as a single line in the following format:

Nickname, gender number, age number, reading style number

Example:

The following line from the readers.txt file:

Casanova, 1, 2, 6

Means that the reader has registered under the pseudonym Casanova, he is a MAN, between the ages of 18 and 25 and who enjoys reading history books.

· In the booksread.txt file, a line will be dedicated to each registered reader which will be in the form:

Nickname, read book number 1, read book number 2,…

Indeed, during the phase of entering the reader's profile, the program must display the list of books in the repository (books.txt file) and the user enters the numbers of the books he has already read and which will be stored on the same line as its nickname in the booksread.txt file.

Example:

For this list of books:

1 - Débuter la programmation en langage Java

2 - Apprendre Python

3 - Les Citations du Président Mao Tse-Toung

4 - Don Quichotte de la Manche

5 - Un conte de deux villes

6 - Le Seigneur des Anneaux

7 - Le Petit Prince

The following line from the booksread.txt file:

Casanova, 1, 4, 6

Means that the reader Casanova has already read the books:

·Débuter la programmation en langage Java

·Don Quichotte de la Manche

·Le Seigneur des Anneaux

**Features to achieve:**

①View books: Each new reader should be able to consult the list of books in the repository to select those they have already read.

Entry: the books.txt file

Output: display of the list of books in the books.txt file, each preceded by a sequential number.

② Adding a reader: a new reader must be able to enter their profile by successively answering the questions allowing them to fill in the information described above.

Entry: readers.txt and booksread.txt files

Output: The same files, each enriched with a new line.

Procedure: For fields with precise values, perform secure entry so that only these values are entered by the user. The user must be able to enter multiple books that they

have already read è Show him the list of all books for each entry.

③ View Reader: The user must be able to view the profile of a given reader.

Entry: readers.txt and booksread.txt files

Output: Display of information (understandable by a human) of this reader.

④ Modify a reader: The user must be able to modify the profile information of a given reader.

Entry: readers.txt and booksread.txt files

Output: updated readers.txt and booksread.txt files.

⑤ Delete a reader: The user must be able to delete the profile of a given reader.

Entry: readers.txt and booksread.txt files

Output: The same files, removed from the line for the specified drive.

1. Part II: Visiting the books repository (6 pts)

The books.txt file already contains an initial list of books to read. For the sake of simplicity, we just define each book by its title. Each title is stored on a line of the file. We can already display this list thanks to the “display books” feature present in part I.

**Other features to achieve:**

① Add a book: Each user must be able to add a book title to the repository, but only if it did not exist before. If the addition is successful, the title will be added automatically at the end of the file.

Entry: the books.txt file

Output: The books.txt file with a new book title.

② Modify a book: Each user should be able to modify the title of a book in the repository only if it already exists.

Entry: the books.txt file

Output: The updated books.txt file.

③ Delete a book: Each user should be able to delete a book from the repository only if it already exists. Before deleting, it is necessary to locate its rank in the list of books (books.txt) and to delete this number on all lines of readers in the booksread.txt file.

Entry: the books.txt and booksread.txt files

Output: The single-line stripped books.txt file and the updated booksread.txt file.

1. Part III : Recommendation (6 pts)

The suggestion system to be developed here is based on the opinions of readers. In fact, in order to be able to suggest a book to a reader, it is essential that they leave notes for the books read. The rating values range from 1 (disliked) to 5 (excellent).

This operation gives rise to a scoring matrix whose rows represent the readers and the columns represent the books present in the repository.



**Scoring matrix**

Obtaining this matrix and ensuring its consistency requires enhancement of some of the previous features.

**Features to achieve:**

① Create the scoring matrix: As soon as the program is launched, the scoring matrix must be created and initialized to 0.

Entry: The files books.txt (number of books) and readers.txt (number of readers)

Output: The scoring matrix created and initialized to 0.

② Rate a book: Each user should be able to rate a book as long as they have already read it. This can be done either when adding a new reader or when the user himself chooses to

note a book.

Input: The scoring matrix, the reader's nickname (optional), the readers.txt, books.txt and booksread.txt files.

Output: The updated scoring matrix.

Steps :

- This function will ask the user to enter their nickname if it is not called from the add reader function.

- Checks if the nickname does exist in the "readers.txt" file

- Asks the user to enter the title of the book they want to rate

- Checks if the title does exist in the "books.txt" file. If so, she recovers without rank.

- Check in the "booksread.txt" file whether the reader in question has read the book correctly.

- If so, she lets him grade the book, making sure the grade is an integer between 1 and 5.

**Features to update:**

① Add a reader: Each time a reader is added to the "readers.txt" file, a line is added to the score matrix where all the values are at 0.

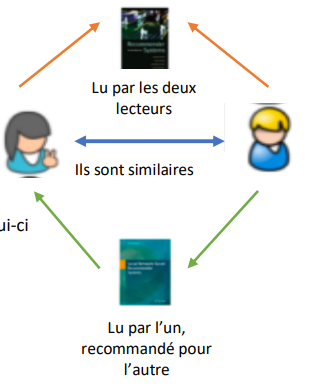
② Delete a reader: Each time a reader is deleted from the "readers.txt" file, the row corresponding to its rank is deleted from the scoring matrix.

③ Add a book: Each time a book is added to the "books.txt" file, a column must be added to the scoring matrix where all of its values are 0

④ Delete a book: Each time a book is deleted from the "books.txt" file, the column corresponding to its rank is deleted from the scoring matrix.

Suggested books

The recommendation algorithm to be implemented here is based on the similarity of reader profiles. In other words, if two readers l1 and l2 have read books together, the system will see them as having similar profiles. So, if one of them (ie l1) has read an additional book, it will be recommended to reader l2.



The method is applied as follows:

- From the scoring matrix, calculate a similarity matrix

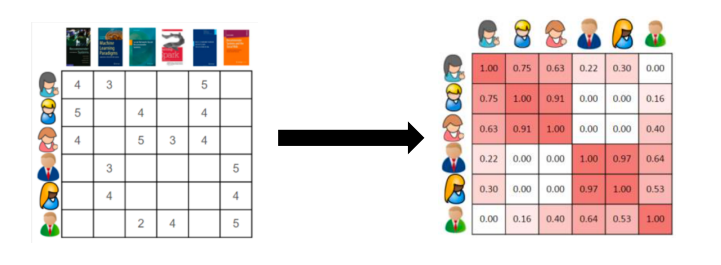
- Each box in the similarity matrix represents the rate of similarity of one reader to another. This rate belongs to the interval [0, 1].

- The diagonal of the similarity matrix is always 1 because any reader is exactly similar to itself.

- The similarity function to be used here is called "Cosine Similarity" and is calculated as follows:

Considering 2 vectors A = [a1, a2,…, an] and B = [b1, b2,…, bn] with n attributes each, the cosine similarity of these two vectors is calculated as follows:

- In our case, each row of the scoring matrix will represent a vector. Thus, by considering a line i1 as the first vector, and a line i2 as the second vector, this amounts to calculating the similarity between the reader of rank i1 and the reader of rank i2.



- Once the similarity matrix has been calculated, it will suffice to:

· Consider a reader (a row of the similarity matrix): let i

· See with whom it is most similar (the max value on the selected row) and identify its column: either j

· Consult in the “booksread.txt” file the books read by reader j and not read by reader i and suggest them to reader i.

Features to achieve:

①Suggest a book: Implement the recommend method described above for a user who requests to suggest books.

Input: The readers.txt and booksread.txt files, the notation matrix

Output: The updated booksread.txt file

Steps :

- Ask the user to enter his pseudonym.

- Check that it exists in the "booksread.txt" file

- Calculate the similarity matrix.

- Suggest all the books read by the reader who resembles him the most and which he has not read before.

- Suggest that they select a title

-When the title is selected, its rank is added to the list of books read for the user in question in the "booksread.txt" file

- Suggest to the reader whether they would like to rate the selected book (assuming they have already read it).

- In case of refusal, display a message reminding him to remember to grade his book after reading.

- To go further: calculate the calculation time of the similarity matrix.

NB: To obtain interesting results during your tests, do not hesitate to use the "readers.txt" and "booksread.txt" files as well as the scoring matrix supplied with the subject of this project.

General remarks :

Þ Secure entries must be made systematically even if they are not always explicitly requested

“Don't hesitate to display messages to users when an action is not possible.

Þ Display the menu at the end of each action to be able to switch to another action.