2021

Book Recommendation System

PROJECT PYTHON FIRST SEMESTER

PAUL MAIRESSE & LOUIS LE MEILLEUR

python™

INT-1

eFrei PARIS
Engineering school of digital technologies

INTRODUCTION

As part of our course of first year of Python, we had to make a project in a little more than a month in a group of two. The goal of this project is to create a program that will suggest books to readers based on their profiles and previous reading.

For this project we choose to use Tkinter for the interface of the program. Since for a lot of unpredictable reason Tkinter could not works, we also add a project without visual interface in the folder.

This program allows a person to connect himself to the application and read books, rate them or search new books in the database, he can also see information on other reader save in the database.

To organize ourselves in group and save a progression of the project we used a repository GitHub. This allows us to get back if we made mistakes and not lose everything we made if our computer had a problem.

We filled the database with some custom data because the application is more complete with a big database, but the files given on Moodle also work (it need to add the missing file for style of book, format is in the README).

Please consider reading the read.me file in both versions of the project and using the ‘help’ command in the command prompt version.

SUMMARY

FUNCTIONALITIES

See a user

Search him in the database and double click on the result, this opens a new window with the information on the user following:

- His name
- His age
- His gender
- His favorite style of book

Add a reader

Go to the connection portal like if you want to connect with your user but click on create.

Edit a user

Connect in the third tab with the user you want to edit and click on the edit button to edit the data you want to change.

Delete a user

Connect in the third tab with the user you want to delete and click on the delete button.

See a book

Search him in the database or find it on the main page if it is there and double click on the panel, this opens a new window with the information following:

- The title
- The average of all notes
- The note the user which is connected has given
- The style of the book

Add a book

To add a book, you need to click on the 4th tab of the main window, A pop up will allow you to add a book.

Edit a book

When you see a book there is a button edit to edit the book. You can edit its name and its style.

Delete a book

When you see a book there is a button to delete the book.

Read a book

When you see a book and if you are connected with a user, there is a button to read or unread it.

Rate a book

When you see a book and if you are connected with a user, there is a list of stars to rate the book, to un-rate it you just have to double click on any star.

Recommend a book

When you are connected with a user, there is a recommendation gallery in the main tab which display last recommendation for the user. This function takes the matrix of ratio of similarity between two users and sort them by decreasing order. For each user it checks if the ratio is higher than 0.35, then it adds all the book that we don't read but this user read and rate more than 2 star (like it).

If the list of all books add is lesser than 10 books, then it adds more book selected by the following criteria by decreasing order so final nu:

- Average note (1 to 5pt or 2pt if there is not enough note)
- Style is the favorite style of the user (2.5pt or 0pt)

OPTIONAL FUNCTIONALITIES

See the style of a book

Because we needed to have the style of every book to choose the color of the panel in the galleries of the home page, we choose to save it in a new file "books_extended.txt". We also add it to the function to update a book so you can change its title or/and its style.

Check data

To be sure that the program won't run with corrupted data that would make it crash or corrupt them even more, we made a function which at the start of the program check every user and book to see if there is not an error with a file or if the file is missing.

Unread a book

Since it was not precise in the project instruction, but it would have made sense to be able to unread book if we accidentally click on the button, we choose to add a function to un-read a book in the database.

See the average note of a book

We used the average note of a book in the calcul of our recommendations, so we choose to also show it in the information of a book

Search Bar

We decided to use a search bar instead of a list of all books since we already had lists in the main page and since we think that, using a big database, a search bar would be more efficient than a big list of books. It works with weights given for every similarity between a result and all the elements in the search bar and then show every result by decreasing order (if weights>max//2).

DESCRIPTION OF THE CODE

Code organization

The project is organized around 3 folders: the root, lib/ (library) and ect/ (executable). The main.py and App.pyw are located at the root, they contain the main architecture of the interface and all call of main functions.

Also, we separate Tkinter from data function in most of the case: there won't be file editing outside from handle_data.py and there won't be Tkinter editing outside from main.py or app_function.py.

we made region with vs-code for folding code to not loose ourselves. This allows us to fold code and easily go to function we need.

```
#region onglet menu…

#region PART 1…

#region PART 2…

#region PART 3…

#region PART 4…

#region close all windows open and task…
```

We also create a globals.py file that will store constants that we use in every file such as the number attributed to each reading style. This way we don't have circular import. We also store a special variable: PATH that allow the program to work on Windows, Mac or Linux and with every to execute the program.

```
2   from pathlib import Path
3
4   PATH = Path("/".join(sys.argv[0].split("/")[:-1]) or "\\".join(sys.argv[0].split("\\")[:-1])).absolute() / "data"
```

File editing

All functions that read / write in files in an external file, so we don't have to do any with open in user_function and book_function. This way every interaction with file is defined by simple function use for many cases. For example, we have a list_readers function that return a generator that contain dictionaries of information for every user (name, gender, age, favorite and index).

```
7   def list_readers():
8       """
9       this function return a dictionary of every user in the file
10      """
11      with open(PATH / "readers.txt", "r", encoding="utf-8") as file:
12          for i,line in enumerate(file.readlines(),1):
13              line = line.replace("\n","")
14              user = dict()
15              user["name"], user["gender"], user["age"], user["favorite"] = line.split(",")
16              user["index"] = i
17              yield user
```

Those are also the functions which detect wrong action and raise exceptions accordingly.

Dictionary choice

To handle data in the code we choose to use dictionary for book and user. With this every function can pass user or book like it was an object. We also used dictionary to create some preset widget to gain in code efficiency when we need to update a whole type of a certain widget for example. Dictionaries are ideal contrary as list because we can store a lot of data of different type with a custom index which allow us to easily retrieve it after.

The main file

The main.py file contain the main architecture of the Tkinter's interface, It call all main functions and update the windows:

```
main.py > ...
333    WINDOW.mainloop() #main loop of tkinter
```

the App is divided in 4 tabs and region in the code:

- first the home page where you can see the recommendation of the user connected, the last books added, the last books read
- second the search tab to search a user or a book
- third to connect as/create a user, or see its information, edit it or delete it
- fourth to add a book

The app_function file

We made an app_function.py file because we needed some more complex widget construct by combining Tkinter widget, for that we choose to make functions which would return the preset widget. Create them again every time we needed would have been impossible (a gallery for example is used 3 time in the home page so each version would need modification at 3 places in the code).

Since we couldn't use class, we couldn't separate the file with for example: in a file preset widget and another with more functional function because it would create some circular import between the two files (some preset widgets need functional function and some functional function need preset widget).

The unit_test file

To test rapidly our project, we used a file which call directly the functions from user_function.py and book_function.py, this also allowed us to use the debugger of vs-code to find our different bugs.

```
unit_test.py
1   from lib.users_functions import *
2   from lib.books_functions import *
3   from ect.handle_data import overide_line
4
5   #print(get_reader("Poool"))
6   #update_reader("Poool",name="hello",gender=4,age=5,favorite=19)
7   #remove_reader("f")
8   #add_reader(name="NotJack",gender='1',age='1',favorite='1')
9
10  #book = get_book("Don Quichotte de la Manche")
11  #user = get_reader("hhi")
12  #print(get_note(user,book))
13
14  #remove_book("Don Quichotte de la Manche")
15  #update_book(old_name="Don Quichotte de la Manche",name="ABC",style="4")
```

EXAMPLES OF FUNCTION

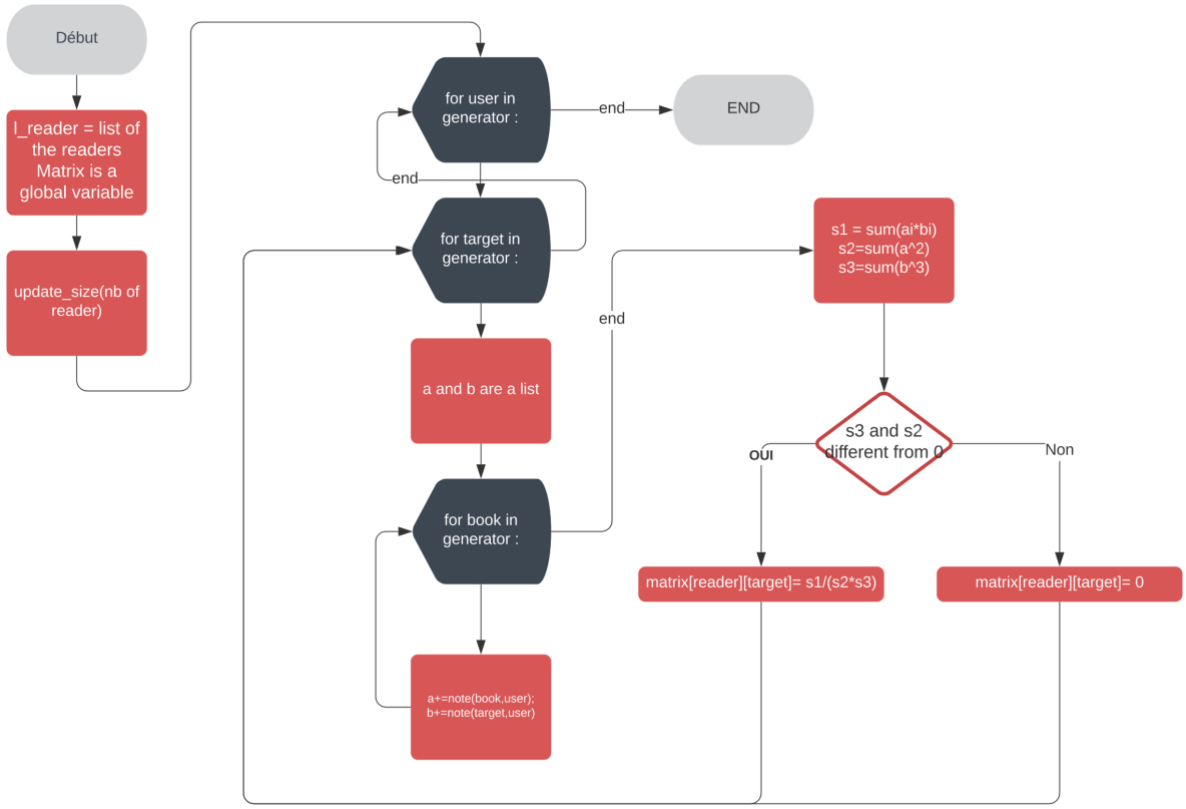
generate_matrix(name):

This is the goal function of the project. It generates a matrix of similarity between every user. This function is call every time we need to update the matrix: add or remove a user, add or remove a book, rate or un-rate a book. (Readings influence on recommendation but not the matrix itself)

The ratio of similarity between two user is find by this following formula where Ai and Bi are the note of the book of index i for respectively user a and b :

$$E = \frac{Sum(a_i \times b_i)}{\sqrt{Sum(a_i^2)} \times \sqrt{Sum(b_i^2)}}$$

Here is a flowchart of the algorithm of the function:



Note: update_size is a function which allow to resize the Matrix to a specific size without changing the object itself.

RESULT

this is the result you must have if you connect with the first user (First), the first four book are book read by the user “Poool” which was made to have similar reading. Then we have other recommendations from other similar readers.

If we connect with a user which read not book like “NotaBigReader”, then we won’t have recommendation from another user, so the program recommend book by note and style (here fantasy).

CONCLUSION

What we learned:

We both agree that using txt file was definitely not the best way to store and use data because of bugs. For example, if you decide to put a comma in the name of the user but the comma is also the separator in your reader's file. We also struggled at the beginning of the project because of the line break at the end of every file we tried to get rid of.

Moreover, the fact that we couldn't use class made Tkinter really hard and really inefficient to use for our preset widget for example, class could also allow us to use personal exception for our raise, we know that the purpose of the project was to verify what we learned during the course, but if we had to do the project again we would use class with inherence and SQLite or JSON for example to store data so the project would be easier to do and more resilient to bugs.

But we learn a lot on GitHub that we used a lot.

How we organized ourselves:

We decided to create a repository in GitHub day one to save the project and every version of the code and to facilitate the work in team. Since Louis struggled using it in visual studio, we also used discord to send .zip files. In term of time management, we started the project the week it was announced so we didn't have any problem with it, we kept updated it along the weeks. Since we already knew Python and made projects with it or in other languages, we had a good vision of the project and the time we would need for every step, so we didn't get surprised by the due date or the time need for a step.