Coursework Assignment Specification
(70 % of the module assessment)
Submission Dateline: 3:30 PM, 16th December 2021

Library Management System

1 Problem Description

1.1 Overview

Your task is to write a simple library management system for **a librarian**. The librarian should be able to search for books by title to check their availability, check out available books, and return any books the members currently have. Details of all books in the library should be stored in an external text file. For each book the following information should be stored: Genre, title, author, purchase date, current loan status (is it available, and if not, who has it?), and a unique ID number which can be used to identify different copies of the same book.

1.2 Member

Members should be identified using their email address. For simplicity it is suggested that you use 4 letters as a member-ID which is the first part of email addresses (e.g.: "coai" for coai@lboro.ac.uk), and you may assume that every combination of four letters are valid members.

1.3 The Text File

Your text file storing the information on each book should contain all the data mentioned in the overview section. You must organise the file as follows:

ID	Genre	Title	Author	Purchase Date	Member Id
1	Sci-Fi	Book_1	Author_1	1/8/2010	coai
2	Fantasy	Book_2	Author_2	1/8/2014	0
3	Sci-Fi	Book_1	Author_1	1/8/2010	0
...		...	...	...	...
...		...	...	...	...
...		...	...	...	...
n	Classics	Book_n	Author_n	1/8/2014	0

where each line provides all the information for a particular book, with the book ID on the left, and the status on the right given either by a member-ID (meaning that the respective member currently has the book), or a value indicating that the book is currently available (in this case 0).

1.4 Searching For Books

Your program should include functionality to search for a book based on its title. Given a search term (e.g. Book_1), your program should return a complete list of books with that title and all their associated information. Additionally, it must highlight the books which are on loan more than 60 days.

1.5 Checking Out Books

In order to withdraw (or check-out) a book from the library, the librarian should provide a member-ID and the book's ID number (note it must be ID, not book title, since there could be more than one copy of the same book). Your program should then

- Check that the input is valid (and return an error message if it is not),
- return an error message if the book is not available due to being on loan to someone else,
- if the book is available, allow the librarian to withdraw the book by updating the book's status in text file accordingly.
- if the member is currently holding any books more than 60 days. It should display a warning message to the librarian. The message should include the details of those books.

1.6 Returning Books

The librarian should be able to return books simply by providing the book's ID number. If the ID is invalid, or the book is already available, the program should return an error message. Otherwise, the text file should be updated accordingly. Additionally, if the member is retuning the books after 60 days. It should display a warning message to the librarian. The message should include how long the books have been borrowed.

1.7 Recommending Books

Your program must have functionality to recommend books to its member. The librarian should provide a member-ID to be able to find out a list of recommended books for the member. Number of books in the list must be between 3 and 10. Note the numbers are inclusive.
The system database should have a transaction log, which keeps the loan history of library books. Based on this log, the system can find out the popular books for each genre and preferred genres for each member to give the suitable recommendation.

2 How to Structure Your Program

The following structure is needed as a final submission for your project:

database.txt Stores all the data (see previous section). You need to populate the file with the realistic data (minimum 10 records).
logfile.txt Stores loan history of library books (i.e. book_id, Checkout Date, Return Date etc.). You need to populate the file with the realistic data (minimum 50 records).
booksearch.py: A Python module which contains functions used to allows librarian to input search terms as strings, and returns the output as described in the previous section. You can either: load the contents of database.txt into a list, and store this as a global variable for repeated use or load the contents of database.txt each time a search is performed.

bookcheckout.py: A Python module which contains functions used to ask librarian for borrower's member-ID and the ID of the book(s) they wish to withdraw. Then, after performing the validity checks and functionality described in the previous section, should return a message letting the librarian know whether they have withdrawn the book successfully.

bookreturn.py: A Python module which contains functions used to ask the librarian for the ID of the book(s) they wish to return and provide either an appropriate error message, or a message letting them know they have returned the book(s) successfully.

bookrecommend.py: A Python module which contains functions used to list the recommended books for a member in the popularity order and appropriately visualise the list by using the Matplotlib module. You should come up with the details of the popularity criteria.

database.py: A Python module which contains common functions that the book search, checkout, return and recommend modules use to interact with the database and log files.

menu.py: A python main program which provides the required menu options to the librarian for the program functionalities. The menu could be based on Python Graphical User Interface-GUI (namely the tkinter python module). In that case, the GUI must use only one window.

3 What to Submit

In addition to the files mentioned in Section 2 you may write a short text file called README (max 500 words). This is to provide any special instructions or warnings to the user (or assessor!), or to draw attention to any aspects of the program that you are particularly proud of (please don't waste time by writing an excessive amount).

All the files (including sample data files) should be compressed into a zip file and submitted electronically as directed.

4 Notes on Expectations:

You will be marked according to your overall achievement, marked according to the Assessment Matrix that will be provided to you separately from this document. However below follows a qualitative description of some general expectation that may help you understand the general level of expectation associated with this piece of coursework.

Technical mastery of Python Your programs should show mastery of what you have been taught.

Design Your programs should be well structured for the task in hand so that it is as easy as possible for:

- a user to use the program for any likely purpose,
- a programmer to understand the code structure and be able to develop it further,
- a programmer to be able to re-use as much as possible of the code in a related application.

Clarity and Self-Documentation Given the structure of your programs, they should be as easy to read and understand as possible. *Lay your code out* so that it can be listed sensibly on a variety of devices: avoid having any lines longer than 80 characters as these may wrap (to reduce the number of "problem lines" you should use 4 spaces for indentation rather than tabs). *Sensible names* should be chosen for all variables, methods etc. *Documentation strings* should be included for each:

Program Fully explain what the program does and how it should be used. Also state who wrote it and when.

Function State what each function does and explain the roles of its parameters. In addition, you should include occasional comments in your code; these may be (a) to introduce a new section in the code, or (b) to explain something that is not obvious. Bear in mind that pointless comments make your code harder to read, not easier.

Restriction

1) Your code must **NOT** include any Class type.
2) Your code must **NOT** have any SQL statements.
3) Your code must **NOT** have any nested functions
4) You must use Python v3.7 or above.
5) Your code must use **only** standard python libraries and Matplotlib.