Student Name: **Weight: 25%**

Student ID:

Project: Classes

Type: Group Assignment

- Students should **ONLY USE** programming constructs covered in the course material.
- **Submissions using programming concepts that are not covered in the course material will be penalized.**
    - **Penalty with no limit could be applied**
- **Late submissions will not be accepted**

Scenario

The Reader's Guild Library requires a new system to manage its inventory and borrowing process more efficiently. Your team's task is to develop a Python solution for their library management system.

Equipment and Materials

For this assignment, you will need:

- Python IDE
- Git software
- GitHub repository

Instructions

This assignment consists of two parts, both to be completed outside of class time. See the course schedule and Brightspace for exact dates. The rubric is available in Brightspace.

Project Part 1: (No Submission)

1. Create the Book class as outlined in the document (recommended filename: *book.py*).
2. Test the Book class to verify its correct operation by running the supplied *test_book.py* program.
3. Each student is encouraged to individually complete Part 1. At the very least, all group members MUST participate in Part 1.

Project Part 2: (Submission)

1. Have one member of your group set up a GitHub repository for Part 2 of this project. Make sure that this GitHub repository is **private**. Add group members and your instructor as collaborators.
2. Create a separate branch in GitHub for each group member, containing the part they will work on. The branch name should include the task name and the student name.
3. Develop the code for the Library Management Module as specified in this document. As a team, you will also need to assess each member's Book class and then finalize one version for your group's solution.
4. GitHub must be effectively used for group collaboration.
5. Ensure all group members publish their parts to their respective GitHub branches.
6. The final project code must be published to master/main on GitHub.
7. Check your solution against the detailed marking criteria available in Brightspace.
8. Create a project video.
   Each group will submit a **video** to Brightspace or their GitHub Repository (either as a URL or an uploaded video file). In the video:
   - EACH team member will introduce themselves – camera should be on. Please use laptop camera/screen share to create video (not a phone).
   - Share the **group's final solution** (Python code) on screen and have EACH team member explain a **portion of the code** that they developed. A debugger may be used to step through sections of code (optional).
   - Provide a demonstration of the application covering all the functionality CORRESPONDING TO THE SAMPLE RUN PROVIDED.
   - Ensure your group's video is 5 to 10 minutes (max!) in length and double check that it is playable and has good audio quality.
9. **Submit the following files to Brightspace**:
   - A ZIP file for the whole project folder that includes:
     - The code of the program that you implemented (.py files).
     - A copy of your sample runs (.txt files) that correspond to the sample runs provided.
     - All data files updated by program after the test runs have been completed.
   - A link to your group's GitHub project repository.
   - Video presentation.
   - Peer assessment.

Library Management System Details

The Library Management system needs to provide the following functionality:

- Upon start-up, load all library books from a CSV file into a list
- Allow general users to select and perform any of the following main menu functions (continuously until the user chooses to exit the system):
    1. Search for books
    2. Borrow a book
    3. Return a book
    0. Exit the system
- Entry of the special passcode of “2130” unlocks additional menu options, allowing librarian users to also select and perform the following librarian menu functions:
    4. Add a book
    5. Remove a book
    6. Print catalog
- Upon shut-down, save all library books from the list to the CSV file

A senior software developer has already done some design work on the *Book class* as well as the core functions required for the Library Management module. Please refer to the detailed information in the following two sections as you complete both programming parts of this project.

Part 1 – Book Class

Define a Book class with the following attributes and methods:

Properties/Attributes

- isbn
- title
- author
- genre (integer: 0, 1, 2, …)
- available (True/False)

Implement these attributes as **hidden** (private).

Methods

Method	Description
Constructor	• Initializes all 5 attributes to the corresponding values passed in.
Getters	• Implement standard getters for each of the 5 attributes.
get_genre_name()	• Implement an additional getter method that returns the name of the genre (as a string) according to the following table: **Genre** **Name** 0 Romance 1 Mystery 2 Science Fiction 3 Thriller 4 Young Adult 5 Children's Fiction 6 Self-help 7 Fantasy 8 Historical Fiction 9 Poetry Hint: implementing this table as a class constant would allow your application module to use it too.
get_availability()	• Implement an additional getter method that returns a string based on the available attribute, I.e., If True Then 'Available' Else 'Borrowed'.
Setters	• Implement standard setter methods for isbn, title, author, and genre.
borrow_it()	• Sets the book's available attribute to False
return_it()	• Sets the book's available attribute to True
__str__()	• Returns a string representation of the book formatted for display. E.g.: `978-0450000000 Gone with the Wind Margaret Mitchell Historical Fiction Available`

Part 2 - Library Management Module

Create a **separate module** (e.g. **library_app.py**) for your Library Management application code which will be comprised of various functions, including a *main()* entry function to control overall processing. The functions listed below are required but you may use additional functions too.

Function Name	Description
load_books()	• Receives an empty list and pathname to an existing CSV file • Iterates over each line (i.e. book) in the file, parsing the attribute values into separate variables • Creates Book objects from each set of attributes and adds them one-by-one onto the list • Returns the number of books loaded
print_menu()	• Receives menu heading (string) and menu options (dict) • Displays the heading and menu options passed in • Inputs selection from user until valid selection is entered • Returns user's **valid** selection
search_books() (menu option 1)	• Before calling this function, input search string from user • Receives a book list and a search string • Iterates through the list of books and checks if the search string appears in isbn, title, author, or genre name. If any match is found, the book is added to the search result list. • Returns search result list • After calling this function, call print_books() passing to it the search result list
borrow_book() (menu option 2)	• Receives a book list • Inputs an ISBN from the user and calls find_book_by_isbn() • If an index to a matching book was returned and that book is currently available, invokes the book's borrow_it() method. Otherwise displays an appropriate message.
find_book_by_isbn()	• Receives a book list and an ISBN • Iterates through the list of books and compares the ISBN parameter to each book's isbn. Iteration stops when an exact match is found or when the end of the list is reached. • Returns the index of the matching book or -1 if none found
return_book()	• Receives a book list • Inputs an ISBN from the user and calls find_book_by_isbn()

(menu option 3)	• If an index to a matching book was returned and that book is currently borrowed, invokes the book's return_it() method. Otherwise displays an appropriate message.
add_book() (menu option 4)	• Receives a book list • Inputs ISBN, title, author, and genre name from the user. Genre name is validated – it must be one of the names listed earlier in the description of get_genre_name() – and translated into its integer value. • Creates a new instance of Book and appends it to the list
remove_book() (menu option 5)	• Receives a book list • Inputs an ISBN from the user and calls find_book_by_isbn() • If an index to a matching book was returned, removes the book from the list. Otherwise displays an appropriate message.
print_books() (menu option 6, 1)	• Receives a book list • Displays a book information heading, then iterates through the book list displaying each Book object on a separate line
save_books()	• Receives a book list and pathname to a CSV file • Iterates over the list, formatting a comma separated string containing each book's attribute values • Writes each string as a separate line to the file • Returns the number of books saved to the file

main()	• Entry point for the Library Management System
	• Coordinates the overall processing:
	  ○ Set up a list of books
	  ○ Input pathname of CSV data file from user and call load_books() to populate the list of books
	  ○ Present the menu, get and evaluate user's selection, repeating until user chooses to quit:
	  ▪ Get additional user inputs as needed
	  ▪ Call appropriate functions to help perform the actions for each menu option
	  ▪ Display relevant context/status messages
	  ○ Call save_books() to save list of Books to file before ending the program

Starting Data File: see b*ooks.csv*.

```
978-0060000000,To Kill a Mockingbird,Harper Lee,3,True
978-0140000000,Pride and Prejudice,Jane Austen,0,False
978-0320000000,The Catcher in the Rye,J.D. Salinger,4,True
978-0350000000,The Hobbit,J.R.R. Tolkien,7,True
978-1570000000,The Kite Runner,Khaled Hosseini,8,True
978-0540000000,The Lord of the Rings,J.R.R. Tolkien,7,True
978-0070000000,The Chronicles of Narnia,C.S. Lewis,7,False
978-0440000000,The Hunger Games,Suzanne Collins,4,True
978-0760000000,Brave New World,Aldous Huxley,2,True
978-0450000000,Gone with the Wind,Margaret Mitchell,8,True
```

Make sure that your application maintains the comma-separated format established in this file.

Test Plan

Please refer to the accompanying document, *Winter 2024 - Project Sample Run.pdf*.

Make sure your program output matches this sample run. When doing your final test run, be sure to begin with a fresh starting data file.

Peer Assessment

- All group members are expected to equally participate in completing the project tasks (e.g. coding, checking style, documenting, testing, video presentation, etc.).
- Each member should assess the contribution/participation of all other group members.
- Each member should assign a mark out of 10 to other members.
- Ensure that the task description is clear and accurately reflects the actual participation.
- Grades will be assigned/adjusted based on the average achieved by each member.
- Please use the following table to complete the peer assessment. Marks/tasks are just samples.

Member/Reviewer	Member 1 Name	Member 2 Name	Member 3 Name	Completed Tasks
Member 1 Name	N/A	9	10	• Task 1 description • Task 2 description
Member 2 Name	8	N/A	9	• Task 3 description • Task 4 description
Member 3 Name	9	10	N/A	• Task 5 description • Task 6 description
Average	8.5	9.5	9.5	

Grading

Item	Percentage
Working Code	**60**
Style	**10**
Documentation	**5**
Testing	**10**
Video	**5**
Version Control	**5**
Peer Assessment	**5**
Total	**100**

Marking Criteria

Rubric available in Brightspace.