Darshan UNIVERSITY
योग: कर्मसु कौशलम्

ASSIGNMENT – Python Programming (2304CS401)

Problem Statement: Library Management System

Input:

1. book name
2. author name
3. search book

Output:

Design a console-based library management system program that will show the following results:

1. Display book details.
2. Display no of books or not (book status)
3. Display books availability
4. Display borrow/return books transactions
5. Provide advanced filtering options

Note:

1. Code should include all the **Object-Oriented concepts**:
    a. **Class and Object:**
        - Define classes for books, members, librarians, and the library system
    b. **Inheritance:**
        - Create a parent class (Person) for shared functionality between Librarian and Member
    c. **Encapsulation:**
        - Use methods to interact with private attributes
    d. **Polymorphism:**
        - Use method overriding for role-specific actions
    e. **Functions:**
        - **strip(), upper(), lower(), title**(): For cleaning and formatting input/output.
        - **find(), replace():** For searching and modifying strings.
        - **split(), join():** For handling multi-word strings.
        - **isalpha(), isdigit(), isalnum():** For validation
        - **strip():** Used to remove leading/trailing whitespace in inputs (e.g., book title and author).
        - **title():** Formats names and titles (e.g., "harry potter" → "Harry Potter").
        - **lower():** Standardizes searches to be case-insensitive.
        - **find():** Locates substrings in the search functionality.
        - **replace():** Modify or clean strings if needed.

This is the bare minimum solution which is expected for the problem.

You can add some more features once you are done with these, like user display with all Borrow books/Return books history, add new collections of books, Mark books as returned and calculate any late fees if applicable, Keep a record of all transactions, Late fees collected.

please **try to complete the bare minimum first.**

All the Best

1. Introduction to Python

- Used print, input, and basic example programs to display instructions and messages.
- Python datatypes for handling book and member information (e.g., str, int, float).
- Tokens and variables for input and output processing.
- Operators for calculations like fines.

2. Python Data Structure, Branching, and Looping

- **String operations:** Format strings for displaying book and member details.
- **Branching:**
  - if-else and if-elif for menu options and conditions (e.g., checking book availability).
- **Looping:**
  - for and while loops for processing lists of books and members.
  - break and continue for specific actions.

3. Python Data Structures and Functions

- **Data Structures:**
  - **List:** Store book and member records.
  - **Dictionary:** Manage book details with keys (e.g., {'Title': 'Book1', 'Author': 'Author1'}).
  - **Set:** Manage issued book IDs.
  - **Tuple:** Store unmodifiable details (e.g., book categories).
- **Functions:**
  - Defined reusable functions for operations (e.g., add_book(), issue_book()).
  - Used lambda expressions for sorting book data.
  - Recursion for searching specific data.
  - Utilized map, filter, and reduce for aggregations and transformations.

4. Python File IO & Modules

- **File Handling:**
  - Store books and member details in files (books.txt, members.txt).
  - Read/write data to maintain persistence.
- **Modules:**
  - Used built-in modules like math (e.g., for fines), random (generate unique IDs), and datetime (due dates).
  - Created a custom module for library utilities.

5. Object-Oriented Programming Concepts and Exception Handling

- **Classes and Objects:**
    - Book, Member, and Library classes to encapsulate data.
- **Inheritance and Polymorphism:**
    - Specialized classes for Member (e.g., StudentMember and FacultyMember).
    - Overrode methods for fine calculation based on membership type.
- **Encapsulation:**
    - Private attributes for sensitive data (e.g., __fine).
- **Abstraction:**
    - Abstracted operations like book search and issue via methods.
- **Exception Handling:**
    - Handled errors like invalid book IDs, missing files, and invalid inputs using try-except.
    - Implemented user-defined exceptions for specific errors.

Week	Dates	Tasks	Deliverable
Week 1	**3rd - 9th February 2025**	▪ Implemented basic Python program structure with sample input/output. ▪ Designed the Book and Member classes with attributes (title, author, member_name). ▪ Used Python operators for basic calculations. **Introduction to Python:** ▪ Python program structure, datatypes,variables,operators, reading input, printing output. ▪ Class and object creation.	**Basic Book and Member classes created with methods for basic operations**
Week 2	**10th - 16th February 2025**	▪ Added methods for adding, updating, and removing books and members. ▪ Implemented branching and looping for menu-based user navigation. ▪ Introduced string operations for formatting messages (e.g., success or error messages). | **Python Data Structures, Branching, and Looping:** ▪ String operations, indexing, slicing, and formatting. ▪ if-else, for, and while loops for menu navigation. ▪ Use of break, continue, and pass statements.	**Interactive menu system for adding, updating, and removing records.**
Week 3	**17th - 23rd February 2025**	▪ Enhanced Library class to manage books and members using list, dict, and set. ▪ Added transaction history using lists and dictionaries. ▪ Implemented functions for issuing, returning, and displaying transactions. **Python Data Structures and Functions:** ▪ List, dictionary, set usage for data organization. ▪ Creating functions with arguments, lambda expressions, and basic recursion. ▪ Use of map, filter, and reduce for processing lists.	**Library class with CRUD operations and transaction logging.**
Week 4	**24th February - 2nd March 2025**	▪ Introduced file handling to persist book and member data. ▪ Implemented methods to read and write to files (books.txt and members.txt). ▪ Used built-in modules like datetime for handling due dates.	**Persistent data storage with file handling and custom module integration.**

		Python File IO and Modules: ▪ File operations (open, read, write, append). ▪ Built-in modules (datetime for due dates). ▪ Custom modules for reusability.	
Week 5	**3rd - 9th March 2025**	▪ Applied OOP concepts to enhance the system ▪ **Encapsulation**: Private attributes (e.g., __fine). ▪ **Inheritance**: Specialized member classes (StudentMember, FacultyMember). ▪ **Polymorphism**: Overrode fine calculation methods. ▪ **Abstraction**: Abstracted common operations (e.g., search books). **Object-Oriented Programming Concepts:** ▪ Classes and objects, inheritance, polymorphism, encapsulation, abstraction.	**OOP-based Library system with specialized classes.**
Week 6	**10th - 16th March 2025**	▪ Implemented exception handling for invalid operations (e.g., missing book or invalid input). ▪ Added custom exceptions for specific scenarios. ▪ Improved console interface for better user experience. **Exception Handling:** ▪ Built-in exceptions (e.g., ValueError). ▪ Try-except-else-finally blocks. ▪ Custom exceptions.	**Exception handling and refined console interface.**
Week 7	**17th - 23th March 2025**	▪ Fine calculation for overdue books. ▪ Unique book and member IDs using random module. Report generation for issued books and transactions. **Python Modules and Advanced Concepts:** ▪ Math, random, and datetime modules for additional functionality. ▪ Enhanced report generation	**Advanced features implemented and system finalized**