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*** | **22th - 28th December 2025** | ▪ Set up Python environment.<br>▪ Create Book and Member classes with attributes (title, author, member_name).<br>▪ Practice input/output for adding/displaying books.<br><br>**Introduction to Python:**<br>▪ Introduction to Python, history, installation, IDEs, program structure, indentation, comments, input/output. | **Basic class structure + sample I/O program** |
| ***Week 2*** | **29th - 4th January 2026** | ▪ Implement menu navigation using if-else.<br>▪ Add methods for adding, updating, and removing books/members.<br>▪ Use operators for simple calculations (e.g., book count).<br><br>**Introduction to Python:**<br>▪ Datatypes, tokens, variables, operators (arithmetic, assignment, comparison, logical, identity, membership, bitwise), type conversions. | **Interactive menu system for adding, updating, and removing records.** |
| ***Week 3*** | **5th – 11th January 2026** | ▪ Implement search functionality (case-insensitive).<br>▪ Format book titles/authors with .title().<br>▪ Use find() and replace() for flexible search.<br><br>**Python Data Structure, Branching and Looping:**<br>▪ String functions (strip, title, lower, upper, find, replace, split, join), indexing, slicing, formatting. | **Search & filter system for books.** |
| ***Week 4*** | **12th – 18th January 2026** | ▪ Add looping for menu navigation.<br>▪ Implement borrow/return transactions with loops.<br>▪ Use break and continue for transaction flow.<br><br>**Python Data Structure, Branching and Looping:**<br>▪ Branching (if, elif, nested if), looping (for, while, range`), break/continue/pass. | **Borrow/return functionality.** |
| ***Week 5*** | **19th - 25th January 2026** | ▪ Store books/members in lists/dicts.<br>▪ Use sets for issued book IDs.<br>▪ Add transaction history using lists/dicts. | **CRUD operations + transaction log.** |

| | | | |
|---|---|---|---|
| | | **Python Data Structures and Functions:**<br><br>▪ Lists, sets, tuples, dictionaries, list comprehension. | |
| ***Week 6*** | **26th – 1st February 2026** | ▪ Create reusable functions (add_book, issue_book, return_book).<br>▪ Use recursion for searching books.<br>▪ Apply map/filter/reduce for aggregations (e.g., count issued books).<br><br>**Python Data Structures and Functions:**<br>▪ Functions, arguments, docstrings, recursion, map, filter, reduce. | **Functional programming integration.** |
| ***Week 7*** | **2nd - 8th February 2026** | ▪ Persist data in books.txt, members.txt.<br>▪ Implement read/write operations.<br>▪ Create custom utility module for library functions.<br><br>**Python File IO & Modules:**<br>▪ File IO (open, read, write), modules. | **File persistence + modular design** |
| ***Week 8*** | **9th - 15th February 2026** | ▪ Use random for unique IDs.<br>▪ Use datetime for due dates.<br>▪ Use math for fine calculations.<br><br>**Python File IO & Modules:**<br>▪ Math, Random, Datetime modules. | **Unique IDs + fine calculation system** |
| ***Week 9*** | **16th – 22nd February 2026** | ▪ Refactor system into OOP design.<br>▪ Add constructors (__init__) for Book, Member.<br>▪ Encapsulate attributes.<br><br>**Object Oriented Programming Concepts and Exception Handling:**<br>▪ Classes, objects, constructors. | **OOP-based system with constructors** |
| ***Week 10*** | **23rd – 01st March 2026** | ▪ Create Person parent class.<br>▪ Override fine rules for Student vs Faculty.<br>▪ Implement abstraction for search/issue operations.<br><br>**Object Oriented Programming Concepts and Exception Handling:**<br>▪ Inheritance, polymorphism, abstraction, encapsulation. | **Role-based inheritance + encapsulation** |

| | | | |
|---|---|---|---|
| ***Week 11*** | **2nd − 8th March 2026** | ▪ Add error handling for invalid book IDs, missing files.<br>▪ Create custom exceptions (BookNotFoundError, MemberNotFoundError).<br><br>**Object Oriented Programming Concepts and Exception Handling:**<br>▪ Exception handling (built-in, try/except, user-defined). | **Robust exception handling.** |
| ***Week 12*** | **9th − 15th March 2026** | ▪ Generate comprehensive reports (issued/returned books, fines collected).<br>▪ Refine menu interface.<br>▪ Optimize code with modular design<br><br>**Object Oriented Programming Concepts and Exception Handling:**<br>▪ Exception handling. | **Finalized Library Management System ready for demo.** |