

**FACULTY: OF INFORMATION COMMUNICATION TECHNOLOGY**

**(PROG211) object orIented programming**

Title: ASSIGNMENT

Issue Date: Week 2

Due Date: Week 4

Lecturer/Examiner: MR AMANDUS B CAULKER

Class: BBITF 1102

ID Number: 905005018

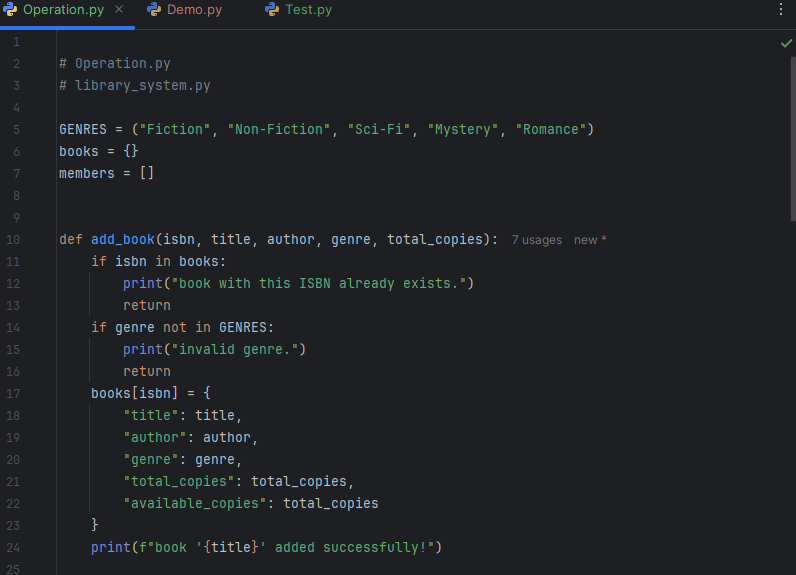
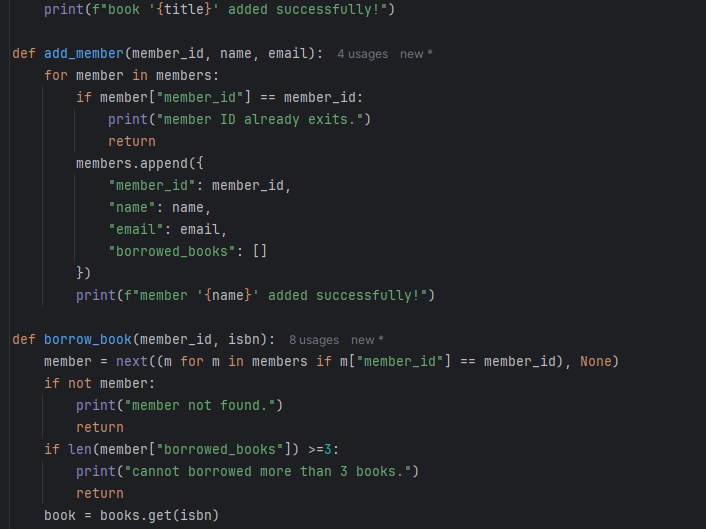
Semester/Year: SEMESTER 3 of 1

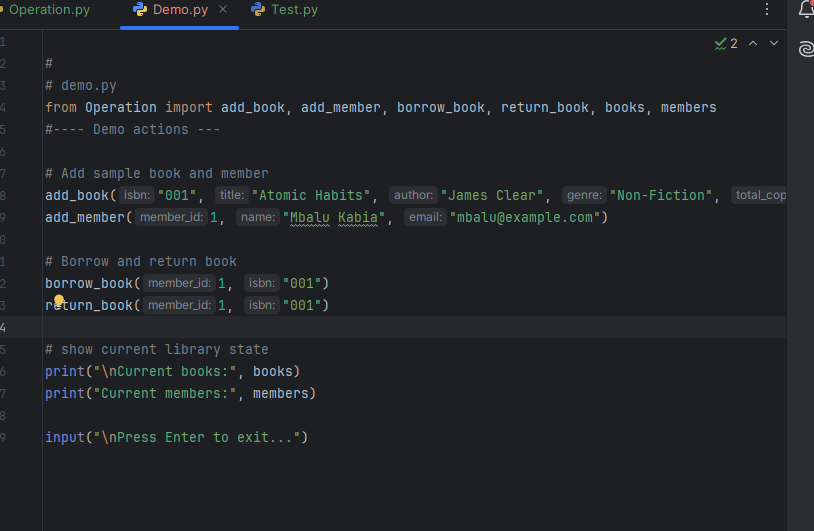
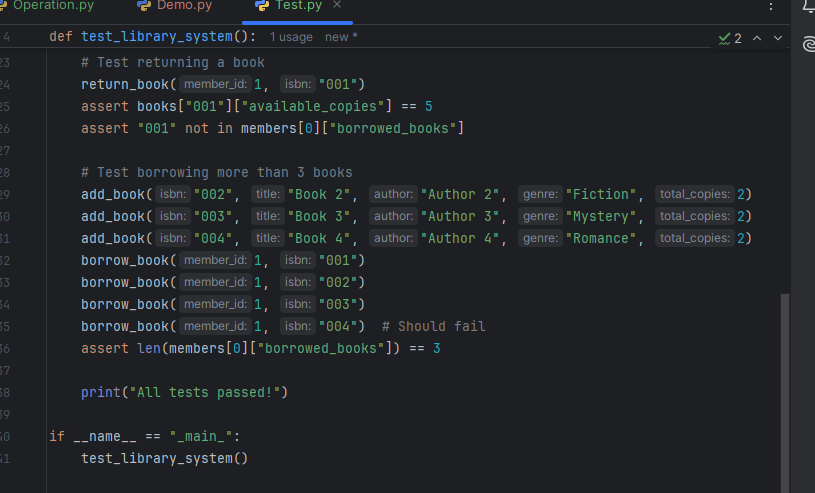
Academic Honesty Policy Statement

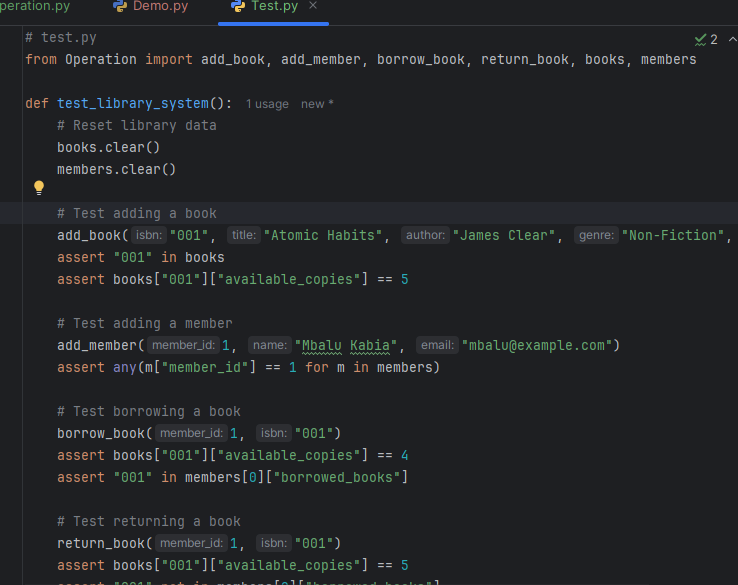
I/We, hereby attest that contents of this attachment are my own work. Referenced works, articles, art, programs, papers or parts thereof are acknowledged at the end of this paper. This includes data excerpted from CD-ROMs, the Internet, other private networks, and other people’s disk of the computer system.

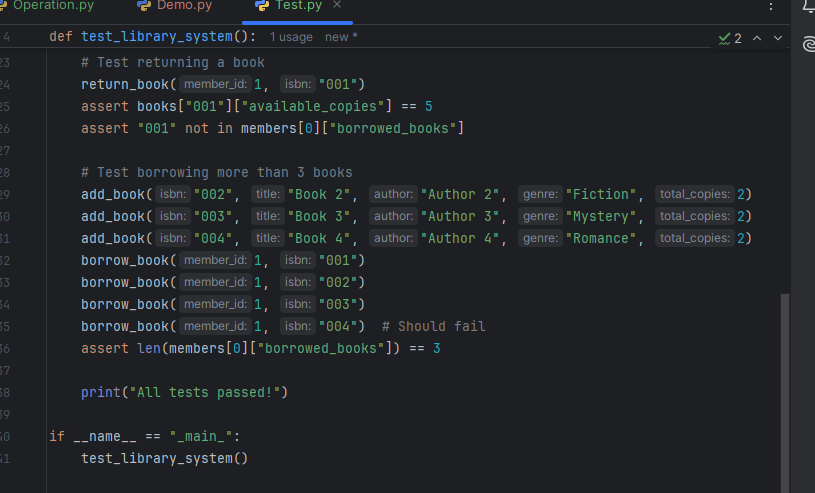
Student’s Signature: Date: / /2025

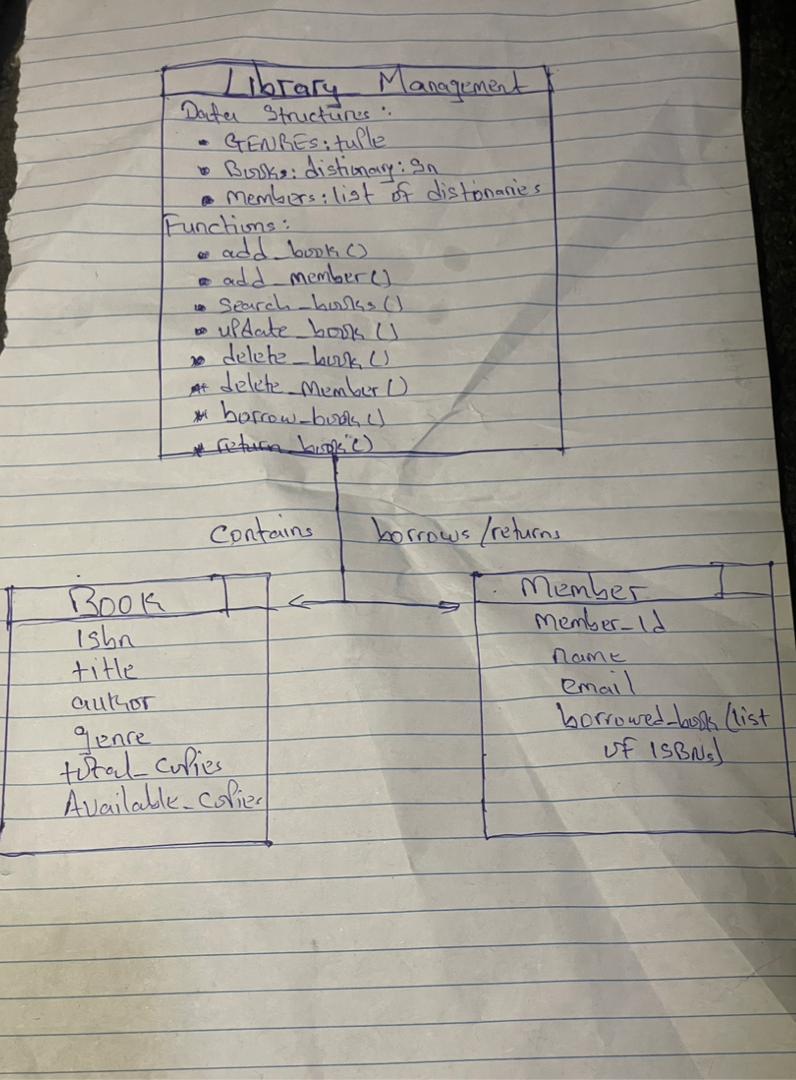
|  |  |
| --- | --- |
| LECTURER’S COMMMENTS/GRADE: | *for office use only upon receive*  Remark  DATE :  TIME :  RECEIVER’S NAME : |

****

****

****

****

****

**DESIGN RATIONALE**

**INTRODUCTION**

The Mini Library Management System was designed in Python to manage books, members, and borrowing transactions in a simple and efficient way. The main goal was to use built-in Python data structures dictionary, list, and tuple to store and manipulate data without using databases. Each data structure was chosen based on its unique characteristics, speed, and suitability for the required operations.

**1. Dictionary (for Books)**

Books are stored in a dictionary with the ISBN as the key. This structure was chosen because:

* Fast lookups: Each book can be quickly accessed using its unique ISBN key instead of searching through a list.
* Clear organization: Book details (title, author, genre, total copies, available copies) are stored as key value pairs within the dictionary.
* Easy updates: Dictionaries make it simple to add, modify, or delete records by referencing their key.

**2. List (for Members)**

Members are stored in a list of dictionaries. Each dictionary holds a member’s details, including their borrowed books.

* Flexibility: Lists can store multiple records (members) and maintain their order of addition.
* Iteration friendly: It is easy to loop through all members when searching, updating, or deleting.
* Supports complex data: Each member dictionary includes a nested list (borrowed books) to track what they have borrowed.

**3. Tuple (for Genres)**

The list of valid genres is stored in a tuple:

* Immutability: Tuples cannot be changed after creation, ensuring genre names remain fixed and consistent throughout the system.
* Data integrity: Prevents accidental modification of valid categories.
* Memory efficiency: Tuples use less memory than lists and load faster since their contents never change.

By using a tuple, the system guarantees that only valid genres are accepted when adding or updating books.

**4. Why These Structures Work Together**

Each data structure plays a unique role in the overall design:

* The tuple defines the allowed categories for all books.
* The dictionary efficiently stores and manages each book’s data.
* The list holds multiple members, each with their own borrow history and attributes.

This combination ensures:

* + Simplicity of implementation
  + Fast searching and updating
  + Clear organization of data
  + Easy expansion if new features are added later (e.g., due dates or fines)

**Conclusion**

The design of the Mini Library Management System demonstrates how Python’s core data structures can be combined to build a small but functional information system.

Dictionaries provide speed and structure, lists offer flexibility, and tuples ensure consistency.

This makes the system easy to understand, efficient to run, and well-suited for small-scale library management without external databases.