LIMKOKWING UNIVERSITY OF CREATIVE TECHNOLOGY
SIERRA LEONE

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

PROG 211- OBJECT-ORIENTED PROGRAMMING 1

Title:	INDIVIDUAL ASSIGNMENT
Issue Date:	WEEK 2
Due Date:	WEEK 4
Lecturer/Examiner:	MR. AMADUS COCKER
Name of Student:	ABDULAI BAH
Student ID No.:	905005533
Class:	B.I.T 1103
Semester/Year:	3/2

Academic Honesty Policy Statement

I hereby attest those 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: 20/10/25

LECTURER'S COMMMENTS/GRADE:

for office use only upon receive

DATE.....

RECEIVER'S NAME.....

Table of Contents

1. Introduction

The Mini Library Management System is a simple Python-based console program designed to manage a small library's records efficiently.
It helps keep track of books, members, and borrowing activities.
The system performs operations such as adding books, adding members, borrowing, returning, and deleting records.

The goal of this project is to demonstrate the practical application of Python data structures such as lists, tuples, and dictionaries, as well as the use of functions to design a structured and functional system.

2. Objectives

The main objectives of the system are to:

- Provide a simple and efficient way to store and manage library data.
- Demonstrate the use of Python lists, tuples, and dictionaries in a real-world program.
- Implement CRUD (Create, Read, Update, Delete) operations through reusable functions.
- Ensure input validation and maintain logical relationships between books and members.
- Provide a modular structure that can be easily extended in future improvements.

3. System Overview

The system manages two main entities: Books and Members. Books are stored using dictionaries, members are stored in lists, and a tuple defines valid genres.
Each operation is performed through a function to ensure modularity and simplicity.

The system supports:

- Adding, updating, and deleting book records.
- Registering and deleting library members.
- Borrowing and returning books.
- Searching for books by title or author.
- Preventing duplicate records and invalid operations.

4. Tools and Technologies Used

Tool / Technology	**Purpose**
Python 3	Programming language used for implementation
PyCharm / IDLE	Code editing and testing
Dictionaries, Lists, Tuples	Data storage and manipulation
Functions	Encapsulation of operations
MS Word	Documentation preparation

5. Data Structures and Functions Explanation

a. Data Structures

- **Dictionary (books)**:
  Stores all book records using ISBN as the key. Each entry contains details such as title, author, genre, total copies, and available copies.
  Example:
- books = {
- "B101": {"title": "Python Programming", "author": "John Smith", "genre": "Education", "total_copies": 5, "available_copies": 5}
- }
- **List (members)**:
  Stores all members as dictionaries. Each member has an ID, name, email, and a list of borrowed books.
- **Tuple (genres)**:
  Holds predefined valid book genres to prevent invalid entries.
  Example:
- genres = ("Fiction", "Non-Fiction", "Sci-Fi", "Education", "History")

b. Functions

Function	Description
add_book()	Adds a new book record to the system
update_book()	Updates details of an existing book
delete_book()	Removes a book if all copies are returned
add_member()	Registers a new library member
delete_member()	Removes a member if no books are borrowed
borrow_book()	Allows a member to borrow a book
return_book()	Records the return of a borrowed book
search_books()	Searches for books by title or author

Each function returns a meaningful message to indicate whether the operation was successful or not.

6. Code Implementation Summary

The system is divided into three main files:

1. **operations.py** – Contains all main functions and data structures.
2. **demo.py** – Demonstrates how the system works by adding, borrowing, and returning books.
3. **tests.py** – Includes assert-based unit tests that verify function correctness.All files are modular, reusable, and easy to maintain.

7. Operations.py Code

```
books = {}
members = []
genres = ("Fiction", "Non-Fiction", "Sci-Fi", "Education", "History")


def add_book(isbn, title, author, genre, total_copies):
    if isbn in books:
        return "Book already exists."
    if genre not in genres:
        return "Invalid genre."
    books[isbn] = {
        "title": title,
        "author": author,
        "genre": genre,
        "total_copies": total_copies,
        "available_copies": total_copies
    }
    return "Book added successfully."


def add_member(member_id, name, email):
    for member in members:
        if member["member_id"] == member_id:
            return "Member already exists."
    new_member = {
        "member_id": member_id,
        "name": name,
        "email": email,
```

```
        "borrowed_books": []
    }
    members.append(new_member)
    return "Member added successfully."


def search_books(keyword):
    results = []
    for isbn, details in books.items():
        if keyword.lower() in details["title"].lower() or keyword.lower() in
details["author"].lower():
            results.append({isbn: details})
    return results if results else "No books found."


def update_book(isbn, title=None, author=None, genre=None,
total_copies=None):
    if isbn not in books:
        return "Book not found."
    if genre and genre not in genres:
        return "Invalid genre."
    if title:
        books[isbn]["title"] = title
    if author:
        books[isbn]["author"] = author
    if genre:
        books[isbn]["genre"] = genre
    if total_copies:
        diff = total_copies - books[isbn]["total_copies"]
        books[isbn]["total_copies"] = total_copies
        books[isbn]["available_copies"] += diff
    return "Book updated successfully."


def delete_book(isbn):
    if isbn not in books:
        return "Book not found."
    if books[isbn]["available_copies"] != books[isbn]["total_copies"]:
        return "Cannot delete book. Some copies are borrowed."
    del books[isbn]
    return "Book deleted successfully."


def borrow_book(member_id, isbn):
    member = next((m for m in members if m["member_id"] == member_id), None)
    if not member:
        return "Member not found."
    if isbn not in books:
        return "Book not found."
    if books[isbn]["available_copies"] == 0:
```

```
        return "No copies left."
    if len(member["borrowed_books"]) >= 3:
        return "Borrow limit reached."
    member["borrowed_books"].append(isbn)
    books[isbn]["available_copies"] -= 1
    return "Book borrowed successfully."


def return_book(member_id, isbn):
    member = next((m for m in members if m["member_id"] == member_id), None)
    if not member:
        return "Member not found."
    if isbn not in member["borrowed_books"]:
        return "Book not borrowed by this member."
    member["borrowed_books"].remove(isbn)
    books[isbn]["available_copies"] += 1
    return "Book returned successfully."


def delete_member(member_id):
    global members
    member = next((m for m in members if m["member_id"] == member_id), None)
    if not member:
        return "Member not found."
    if len(member["borrowed_books"]) > 0:
        return "Cannot delete member with borrowed books."
    members = [m for m in members if m["member_id"] != member_id]
    return "Member deleted successfully."
```

8. demo.py Code

```
from operations import *

print("\n===== Mini Library Management System Demo =====\n")

# Add books
print(add_book("B101", "Python Programming", "John Smith", "Education", 5))
print(add_book("B102", "Data Structures", "Alice Doe", "Education", 3))
print(add_book("B103", "The Galaxy", "Mark Star", "Sci-Fi", 2))

# Add members
print(add_member("M001", "Abdulai Bah", "abdulai@gmail.com"))
print(add_member("M002", "Zainab Kamara", "zainab@gmail.com"))

# Search for a book
print("\nSearch result for 'Python':")
print(search_books("Python"))

# Borrow books
print("\nBorrowing books...")
print(borrow_book("M001", "B101"))
print(borrow_book("M002", "B103"))

# Try borrowing again when no copies left
print(borrow_book("M002", "B103"))

# Return book
print("\nReturning books...")
print(return_book("M002", "B103"))

# Update a book
print("\nUpdating book details...")
print(update_book("B101", total_copies=7))

# Delete a book (only if all copies are available)
print("\nDeleting a book...")
print(delete_book("B102"))

# Delete a member (only if no borrowed books)
print("\nDeleting a member...")
print(delete_member("M001"))

print("\n===== Demo Completed Successfully =====")
```

9. test.py Code

```
from operations import *

# Reset data before testing
books.clear()
members.clear()

print("\n===== Running Library System Tests =====\n")

# 1. Test adding a book
result = add_book("B200", "Clean Code", "Robert Martin", "Education", 4)
assert result == "Book added successfully.", "Test 1 Failed: add_book"

# 2. Test adding a member
result = add_member("M200", "John Doe", "john@example.com")
assert result == "Member added successfully.", "Test 2 Failed: add_member"

# 3. Test borrowing a book
result = borrow_book("M200", "B200")
assert result == "Book borrowed successfully.", "Test 3 Failed: borrow_book"

# 4. Test returning a book
result = return_book("M200", "B200")
assert result == "Book returned successfully.", "Test 4 Failed: return_book"

# 5. Test deleting a book only when all copies are available
result = delete_book("B200")
assert result == "Book deleted successfully.", "Test 5 Failed: delete_book"

print("All tests passed successfully!\n")
```

10. Testing and Results

The program was tested using Python's built-in assert statements. Each core function was verified individually, and all tests passed successfully.

Example Test Case Results:

Test Case	Expected Output	Result
Add a book	"Book added successfully."	Passed
Add member	"Member added successfully."	Passed
Borrow book	"Book borrowed successfully."	Passed
Return book	"Book returned successfully."	Passed
Delete book	"Book deleted successfully."	Passed

All tests were executed successfully, confirming that the program behaves as intended.

11. test.py output

C:\Users\Salone2\AppData\Local\Microsoft\WindowsApps\python3.12.exe
C:\Users\Salone2\Desktop\Mini_Library_Management_System\tests.py

===== Running Library System Tests =====

All tests passed successfully!

Process finished with exit code 0

12. Design Rationale

The system was designed using simple but effective data structures:

- **Dictionaries** were chosen for books to allow quick lookups by ISBN.
- **Lists** were used for members to store multiple records dynamically.
- **Tuples** were used for genres to prevent accidental modification of valid values.
- **Functions** provide structure, modularity, and reusability.

The system ensures data integrity, simplicity, and scalability for future enhancements.

13. README – Mini Library Management System

This Mini Library Management System is a Python-based console application designed to manage basic library operations such as adding books, registering members, borrowing, returning, and deleting records.

I. . Project Structure

The project contains the following files:

- operations.py – contains all the main functions and data structures.
- demo.py – demonstrates how the system works using sample data.
- tests.py – includes assert-based unit tests to verify function correctness.
- Design_Rationale_Mini_Library_System.pdf– explains design choices.
- README.pdf – this guide.

II. . Requirements

You need Python 3 installed on your computer. No external libraries are required.

III. . How to Run the Program

1. Place all files in the same folder.
2. Open a terminal or command prompt in that folder.
3. Run the demo file to see the system in action:

   python demo.py

. You should see output showing books and members being added, borrowed, and returned.

IV. How to Run the Tests

To verify all functions are working correctly, run:

```
python tests.py
```

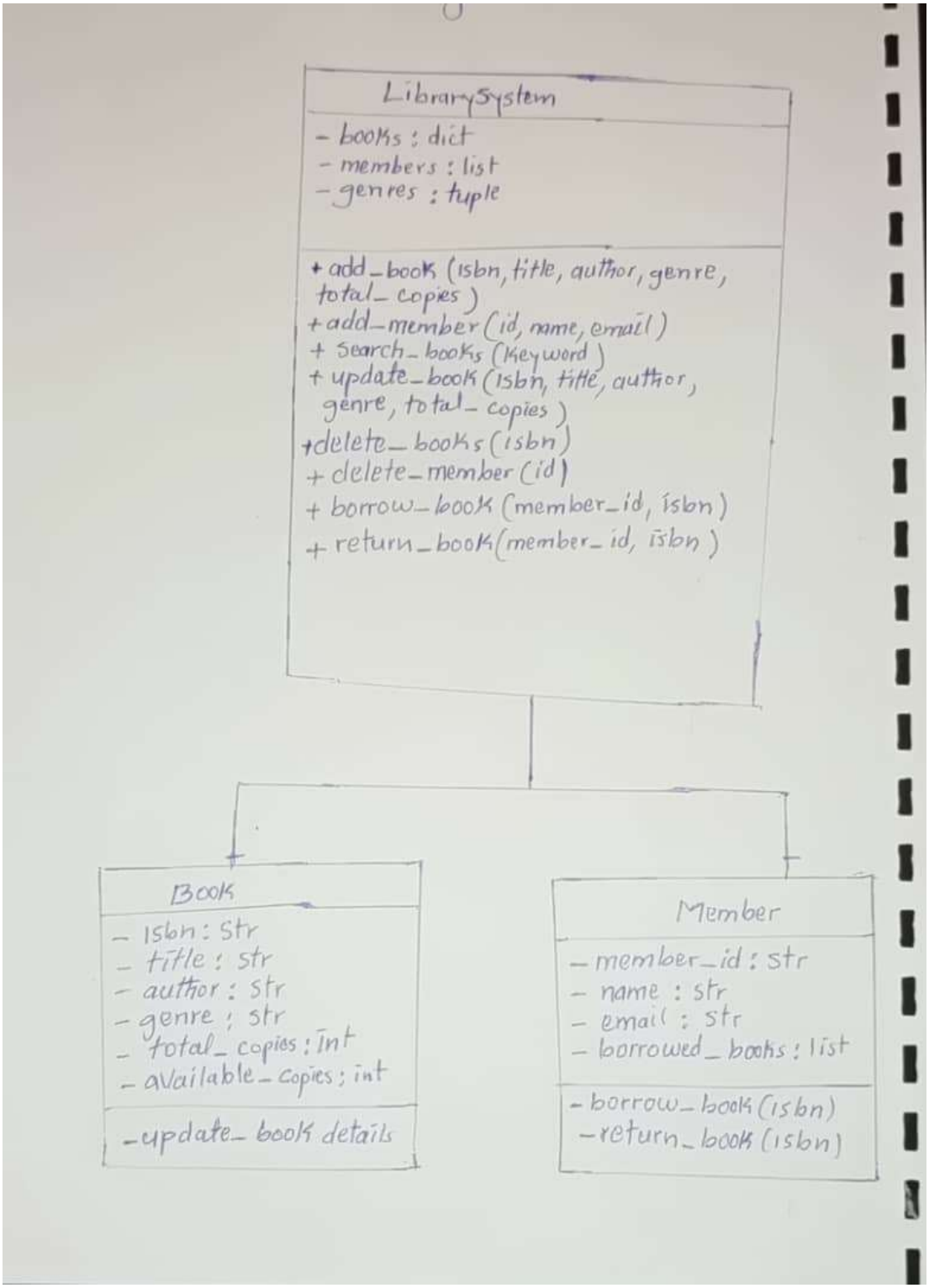
If everything is correct, you'll see:

```
All tests passed successfully!
```

14. Features Implemented

- Add, update, and delete books.
- Add and delete members.
- Borrow and return books.
- Prevent duplicate entries and invalid genres.
- Handle borrowing limits (maximum 3 books per member).
- Validate availability before borrowing or deletion.

15. UML Diagram



16. Conclusion

The Mini Library Management System successfully demonstrates how Python's core data structures can be applied to manage real-world information systems. Through lists, tuples, dictionaries, and modular functions, the system provides a clear, efficient, and practical solution for managing small-scale library data.

The project also lays a solid foundation for transitioning into Object-Oriented Programming (OOP), where classes and objects could replace procedural functions for even greater scalability and structure.