

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

page1image9370432 **(PROG211OBJECT-ORIENTED PROGRAMMING 1)**

page1image9370432

Title : Individual Assignment

Issue Date : Week 2

Due Date : Week 4

Lecturer/Examiner : Mr. Lavalie

Name of Student/s : Sahr Kobba

Student ID No. : 905005608

Class : BSEM 1102F

Semester/Year : 3/2

Academic Honesty Policy Statement

page1image9370432

I/We, 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: November 2023

page1image9370432

LECTURER’S COMMMENTS/GRADE:

for office use only upon receive

Remark

DATE :  
 TIME :

RECEIVER’S NAME :

**Rationale for Mini Library Management System**

**1. Introduction**

The Mini Library Management System was developed using Python’s fundamental data structures — dictionaries, lists, and tuples to efficiently handle books, members, and genres.  
The design emphasizes **simplicity, readability, and alignment with assignment guidelines**, avoiding the use of complex data structures or external libraries.

**2. Data Structure Selection**

**a). Dictionary for Books**

A **dictionary** was selected to manage books since each book can be uniquely identified by its **ISBN (International Standard Book Number)**, which naturally serves as a dictionary key.  
This allows **fast searching, updating, and clear organization** of book records.

Structure

Books = {

"111": {"title": "software engineering", "author": "Mr. santigie kamara", "genre": "Fiction", "copies": 3}

}

**Reasons for choosing a dictionary:**

* ISBN provides a unique and efficient key for each book.
* Enables quick access and updates with constant time complexity (O(1)).
* Offers an intuitive way to store and represent detailed book information.

**b). List for Members**

A **list** was chosen to store library members since membership data grows dynamically and lacks a natural unique key like in dictionaries.  
Each member is stored as a dictionary containing their ID, name, email, and borrowed books.

Structure

members = [

{"member\_id": 1, "name": "kobba", "email": "sahrkobba808h@gmail.com", "borrowed books": []}

]

**Reasons for using a list:**

* Supports easy addition and deletion of members.
* Simple iteration for searching or updating members by ID.
* Well-suited for managing multiple similar objects, such as users.

**c. Tuple for Genres**

A **tuple** was used to define valid genres because tuples are **immutable**, meaning their contents cannot be modified after creation.  
This ensures that the list of genres remains fixed and prevents unintentional changes.

Structure:

genres = ("Fiction", "Non-Fiction", "Sci-Fi", "Mystery", "Biography")

**Reasons for using a tuple:**

* Immutability maintains data integrity.
* Tuples are faster and more memory-efficient than lists.
* Ideal for static reference data that should not be altered.

**3. Function Design**

All system operations such as **adding, updating, deleting, borrowing, and returning books** are implemented through functions.  
This **modular programming approach** enhances readability, reusability, and testing.

**Examples:**

* add\_book() - Adds a new book record.
* borrow\_book() - Decreases available copies and updates the member’s borrowed list.
* return\_book() - Restores book copies and updates member data.

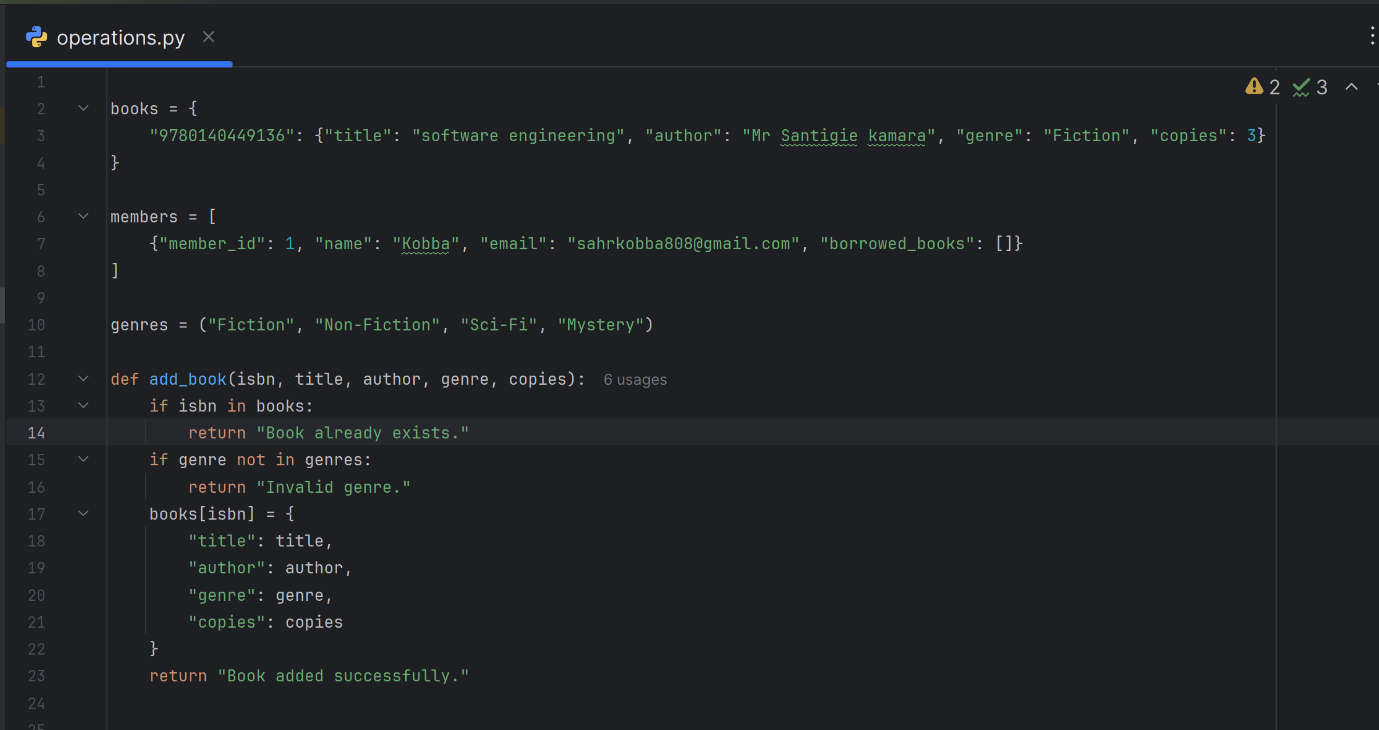
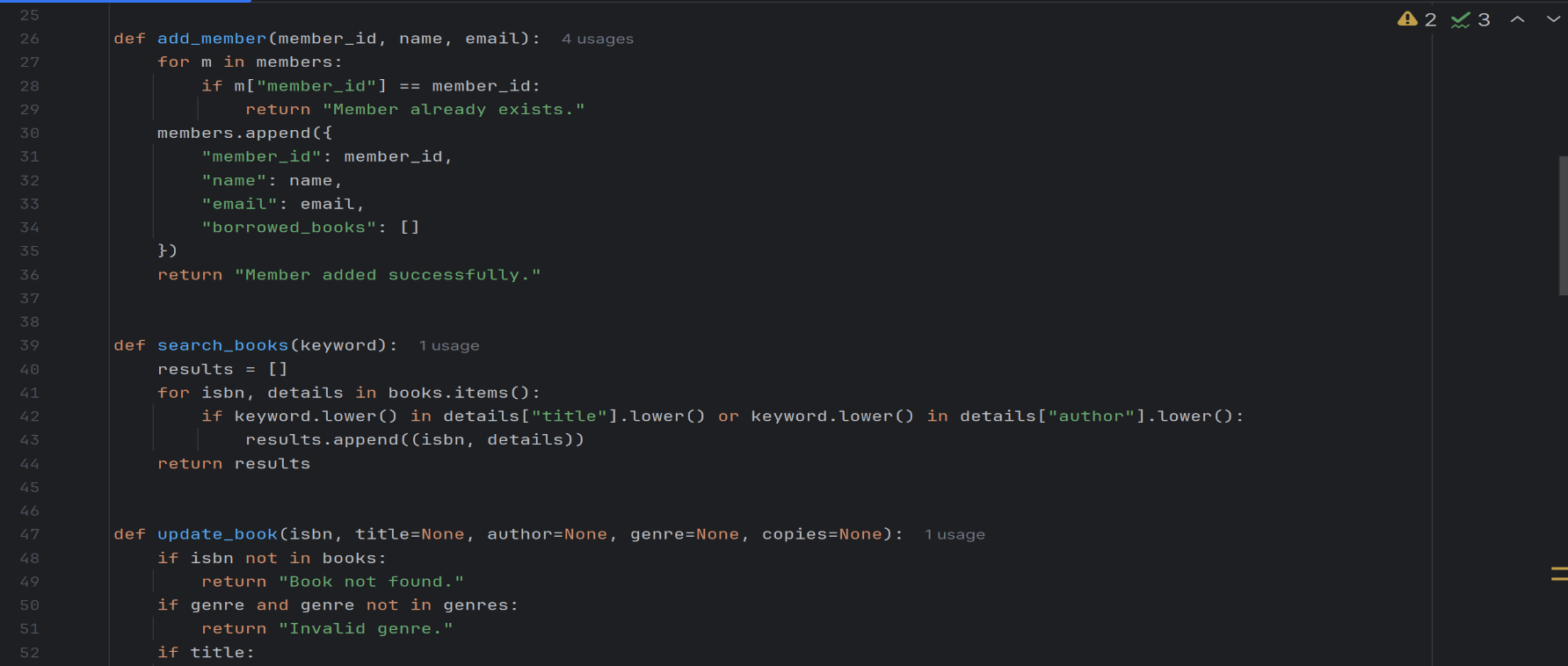
Each function produces a **clear output message** to confirm successful or failed operations, aiding both demonstrations and testing.

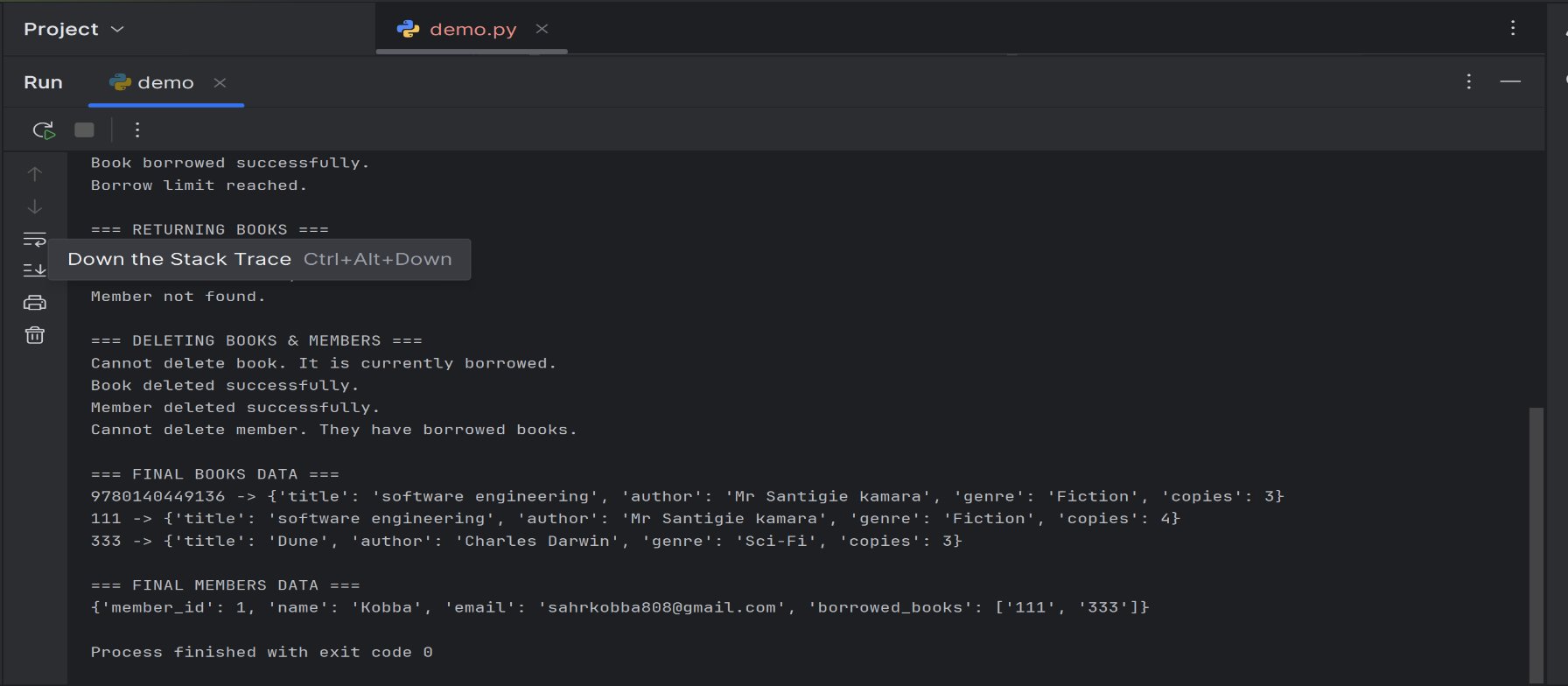
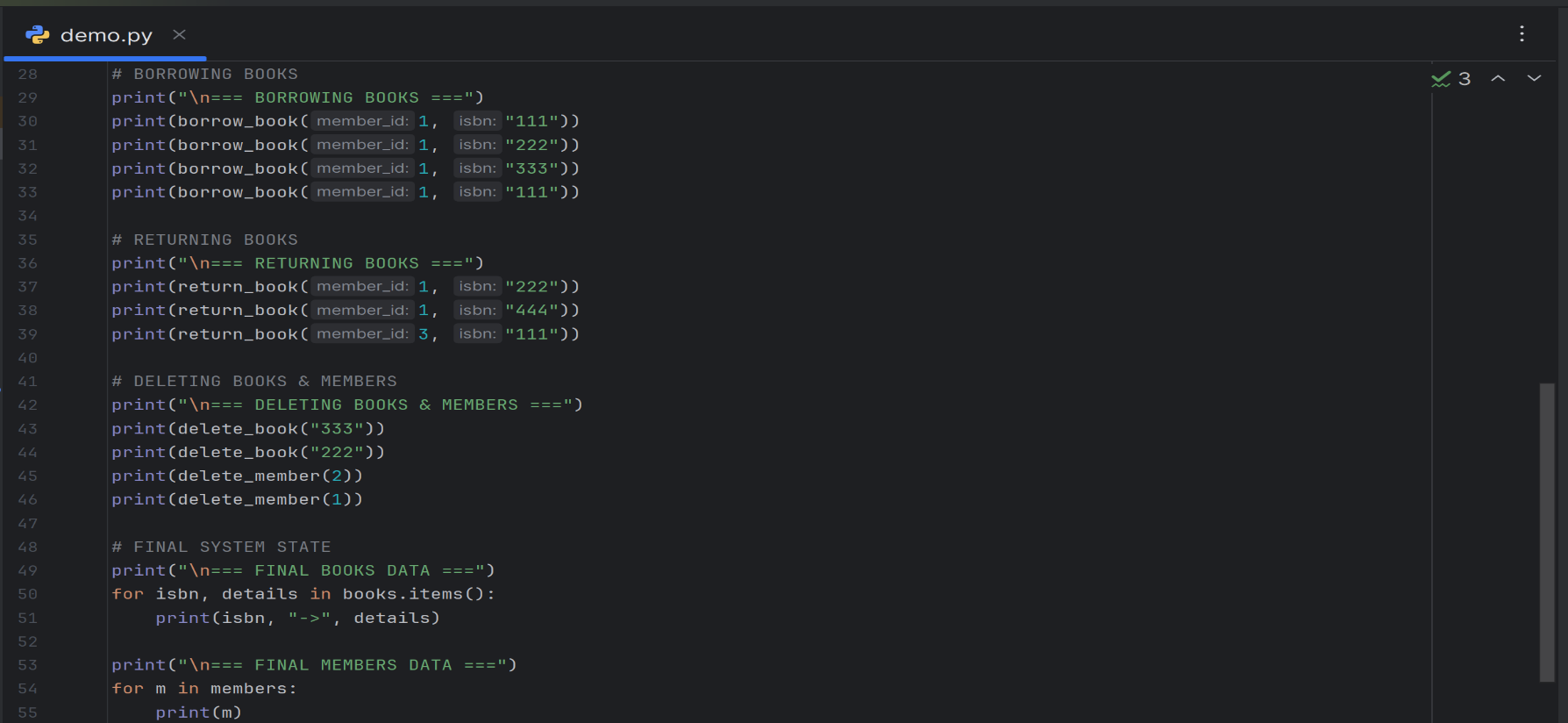
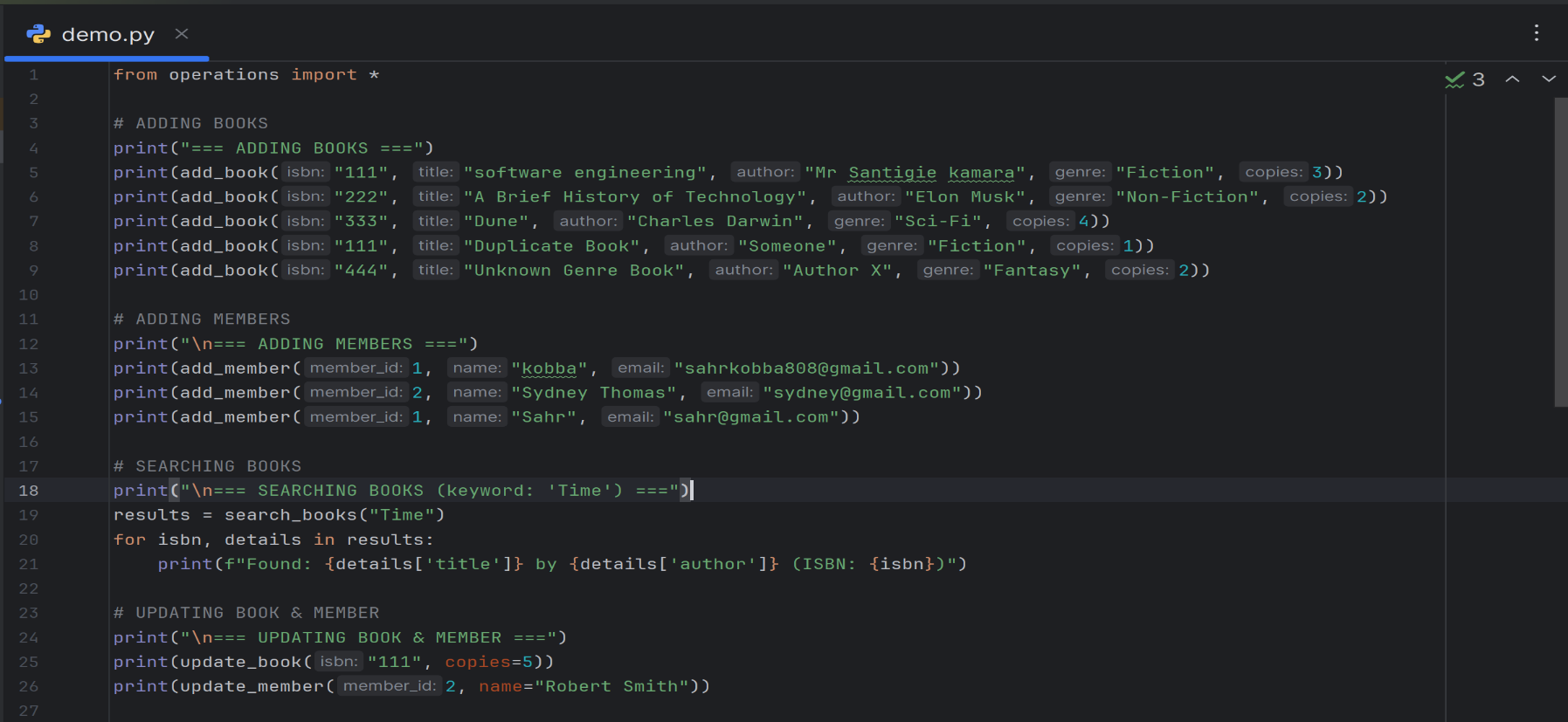
**4. Advantages of the Design**

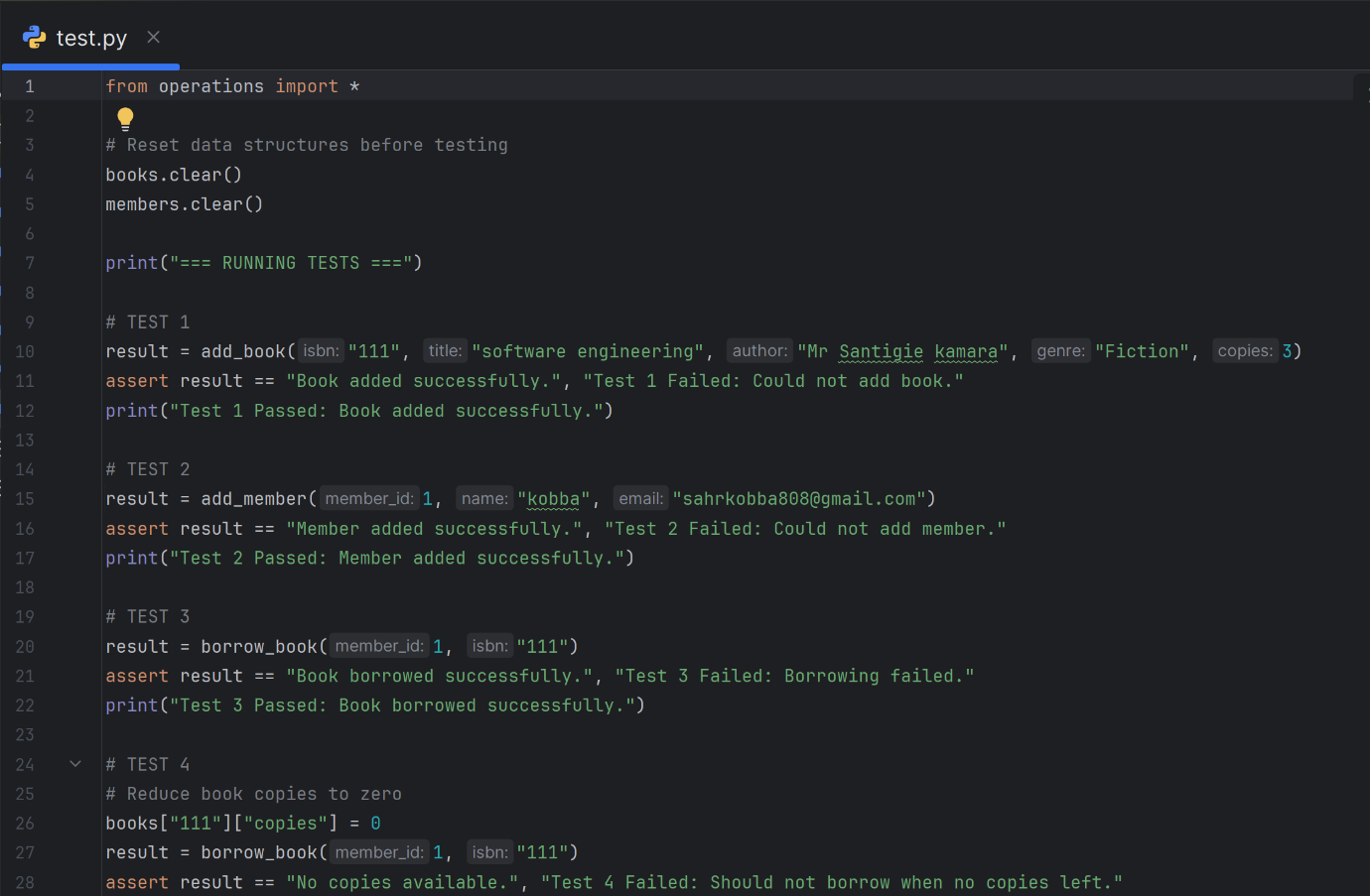
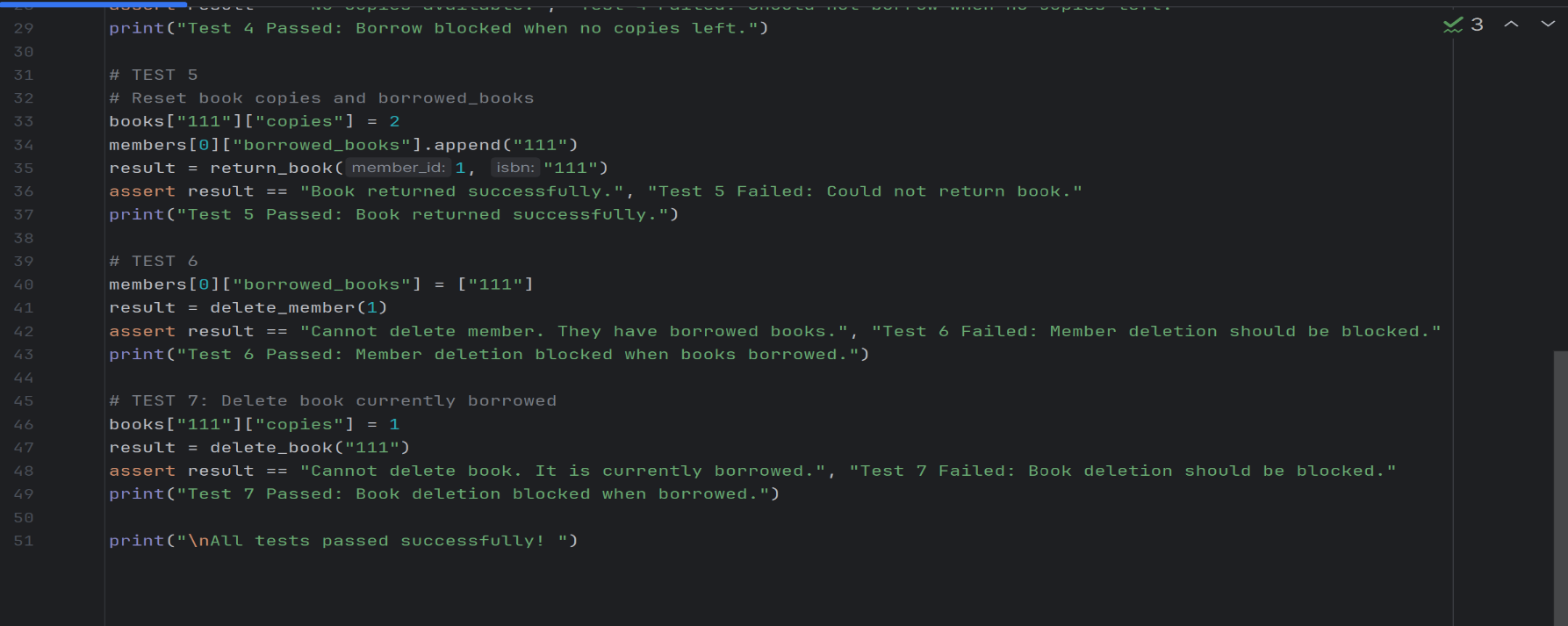
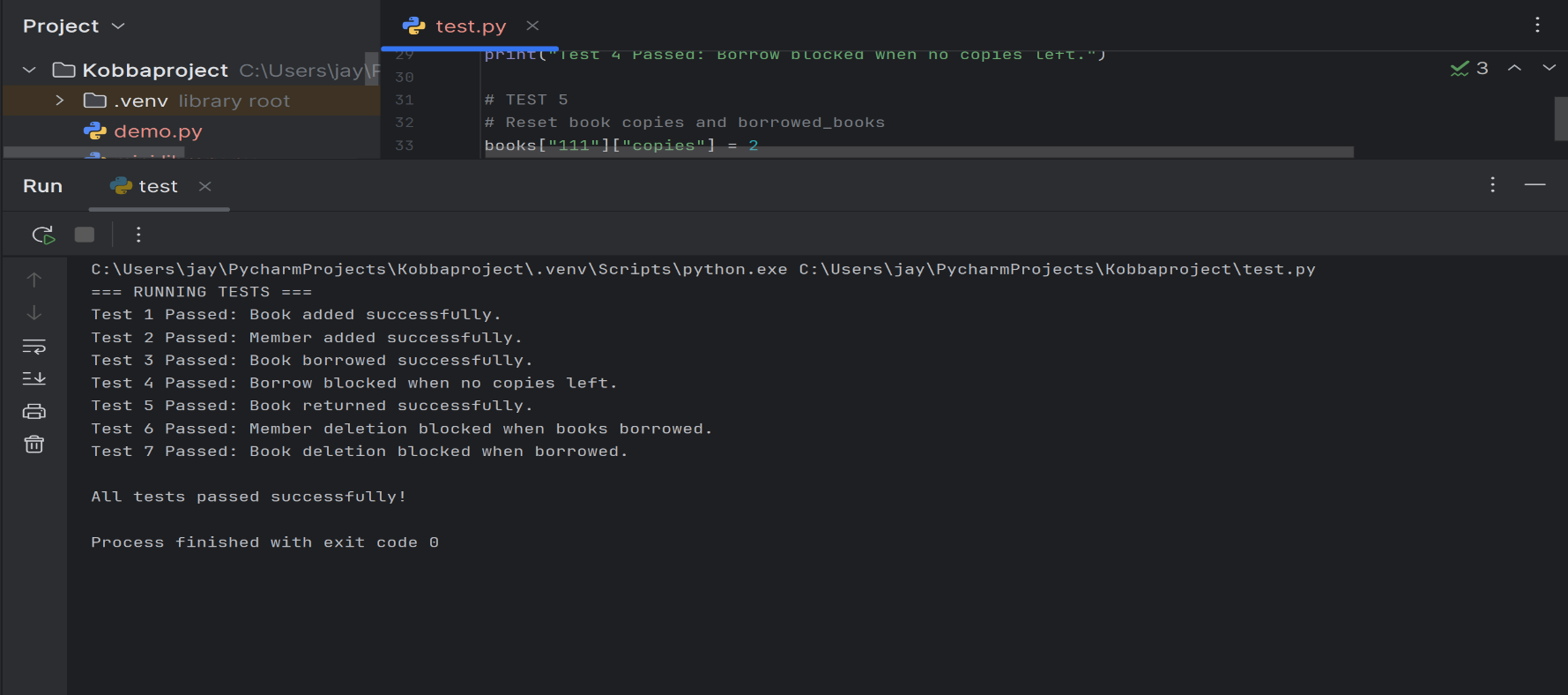
* **Simplicity:** Easy to follow for beginners.
* **Efficiency:** Uses Python’s lightweight and high-performance built-in data types.
* **Reusability:** Independent functions can be tested and reused in other systems.
* **Scalability:** Can be expanded later with database integration or a graphical interface.

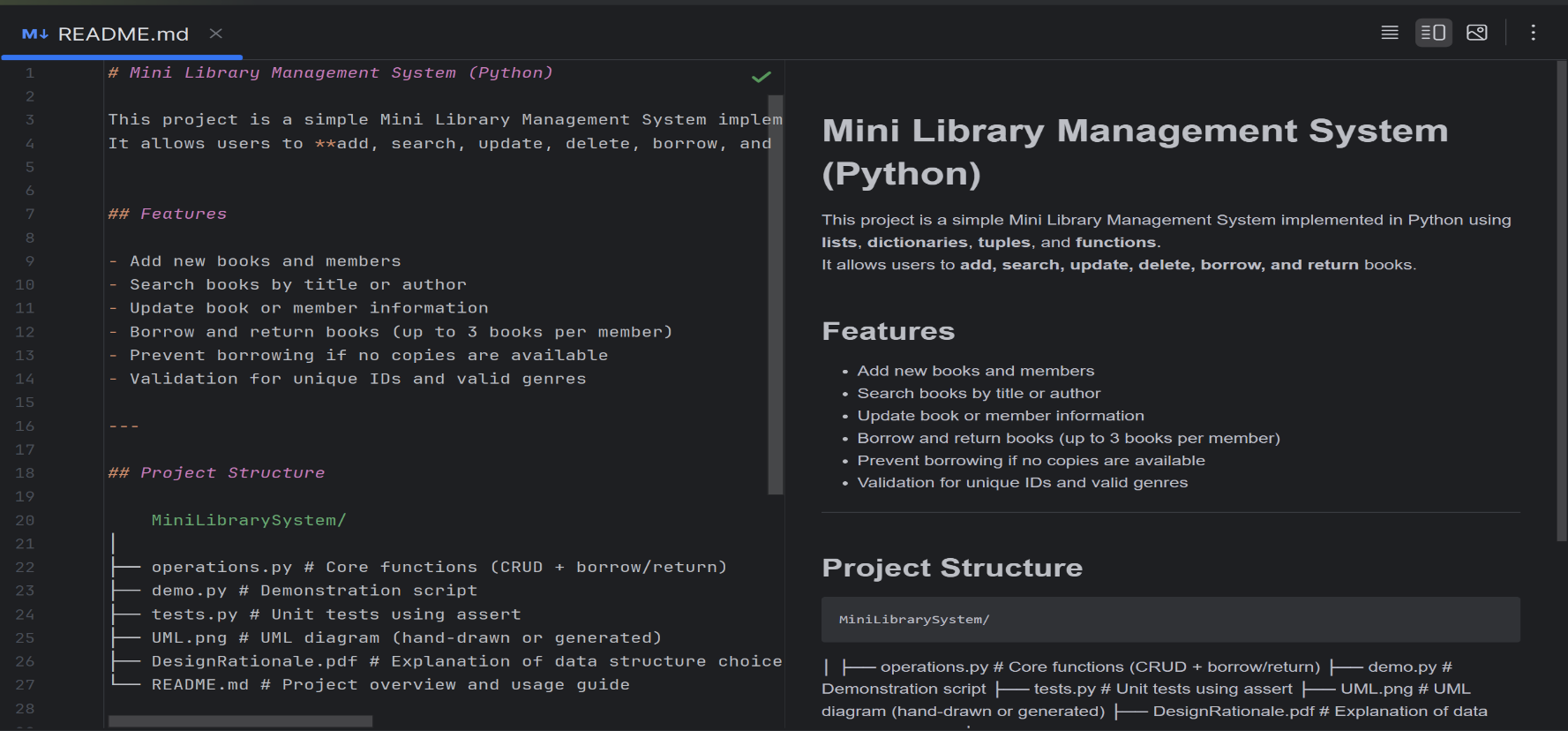
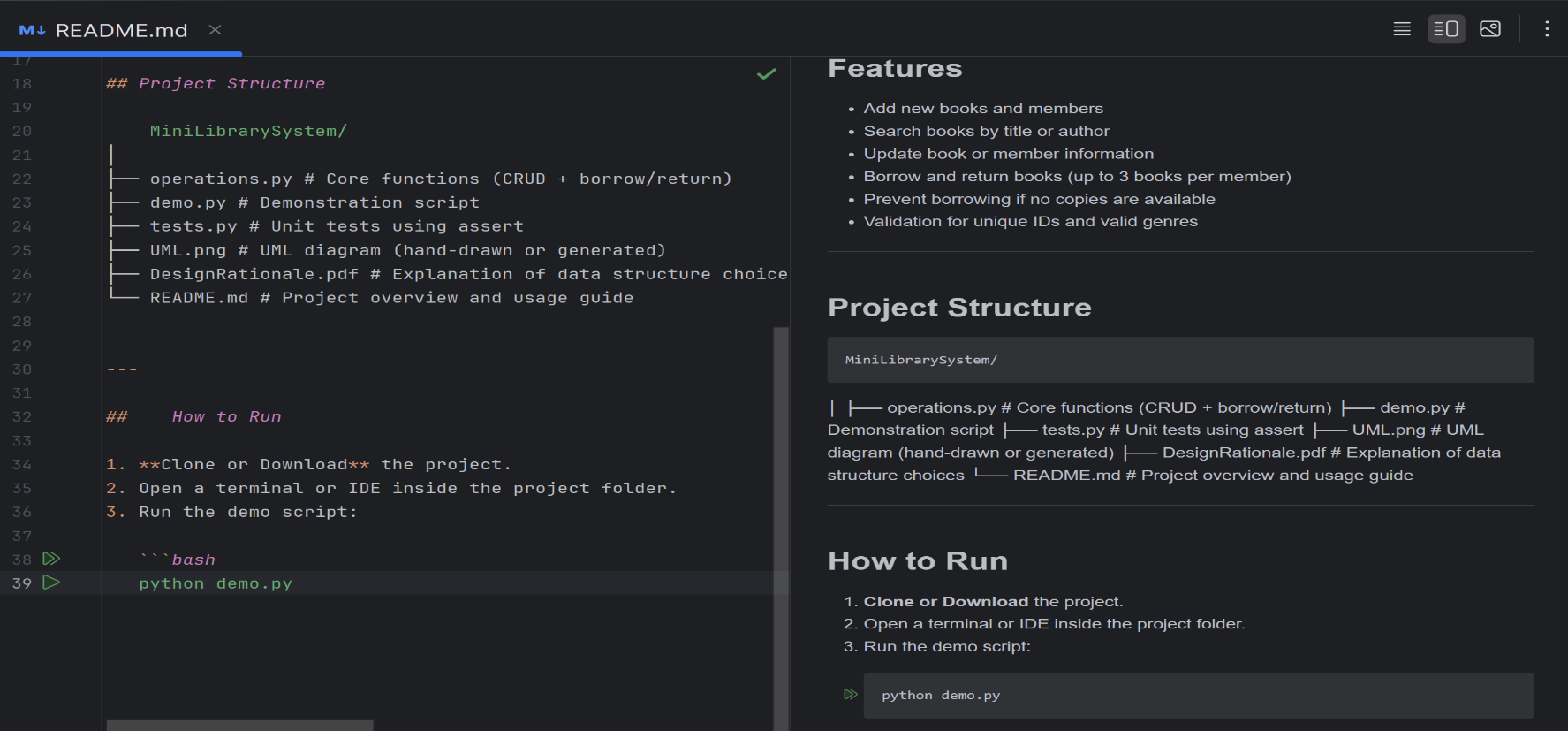
**5. Conclusion**

The combination of **dictionary, list, and tuple** structures provides an excellent balance of simplicity, efficiency, and flexibility for the Mini Library Management System.  
These core Python structures ensure effective **data storage, manipulation, and retrieval**, fulfilling the project’s requirements using only fundamental programming concepts.

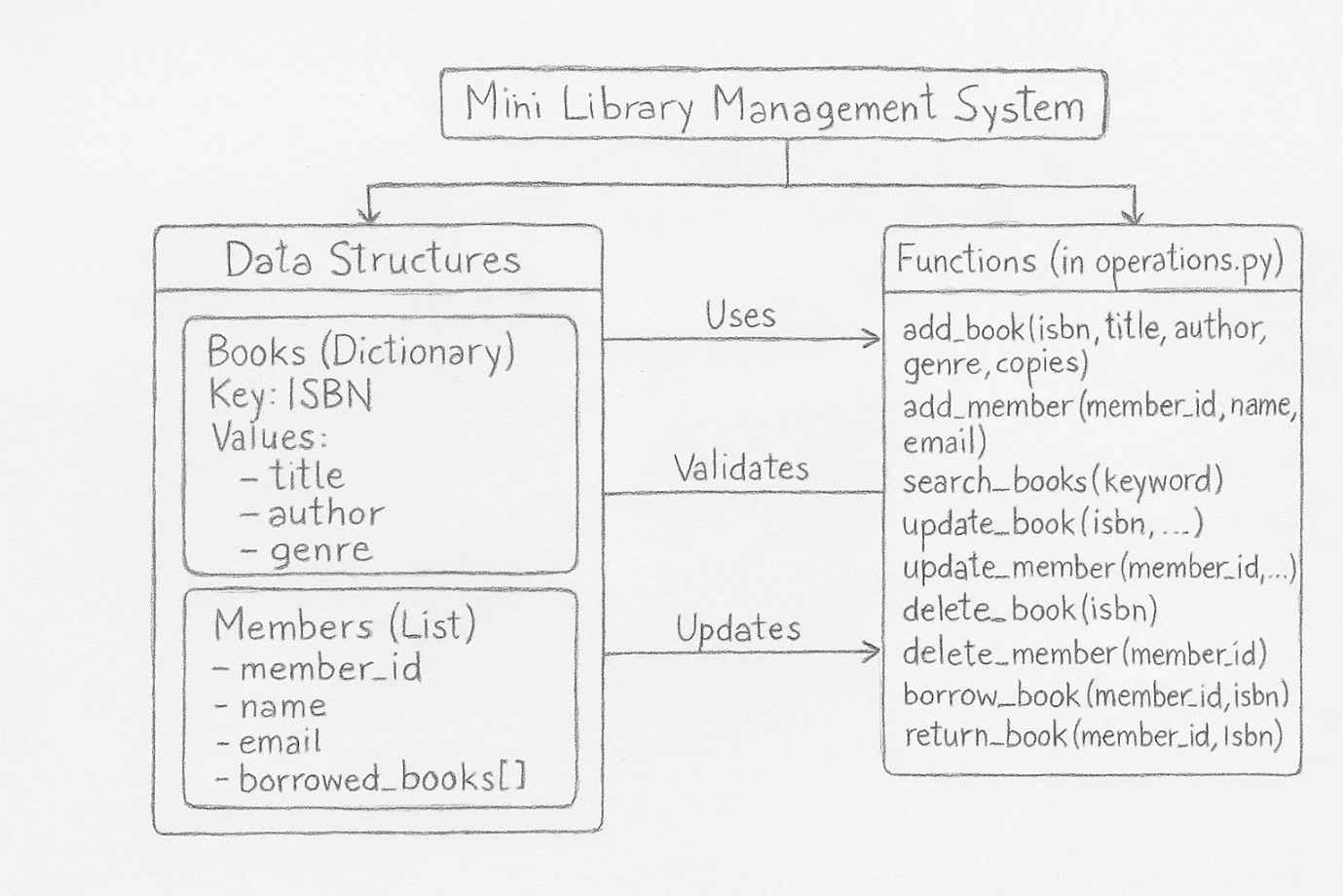
** OPERATION.PY**

**DEMO.py**

**TEST.py**

**README.md**

**UML DIAGRAM**

****