Rationale and UML Explanation for LIMKOKWING MINI LIBRARY SYSTEM

UML Diagram Representation

Below is a textual UML diagram representation of the system’s data structure and major functional relationships.
This representation shows how functions interact with data collections such as dictionaries, lists, and tuples.

Rationale

The design of the LIMKOKWING MINI LIBRARY SYSTEM is based on Python’s native data structures: lists, dictionaries, and tuples. Each structure was carefully selected to optimize data manipulation, enhance readability, and ensure flexibility in future enhancements.

Use of Dictionaries

Dictionaries were chosen to store structured information about books and members. In Python, dictionaries provide key-value mapping, allowing quick access and updates to data. For instance, each book’s ISBN acts as a unique key in the 'isbn_book_details' dictionary, making it efficient to search, update, or delete book records. Similarly, member details are stored as individual dictionaries within a list, enabling unique referencing through their 'Member_Id'. This structure mimics relational database design while maintaining the simplicity of in-memory operations.

Using dictionaries also allows flexible storage of diverse data types under a single entity. Each dictionary can contain strings, integers, and lists (e.g., Borrowed_Books), allowing the system to track not only static information like names and titles but also dynamic ones like borrowed book lists.

Use of Lists

Lists were implemented to manage collections of items that change frequently, such as members and borrowed books. The list data type allows appending, removing, and iterating through elements dynamically. In the system, 'members' is a list of dictionaries, each representing a library member. This allows easy addition of new members, deletion of existing ones, and iteration for searches or updates.

Within each member record, the 'Borrowed_Books' list enables storing multiple titles that a member has borrowed. This dynamic nature of lists ensures scalability since a member's borrowed books can increase or decrease without the need for complex restructuring.

Use of Tuples

Tuples were selected for storing predefined, unchangeable data, in this case, 'book_genres'. The genre list represents a fixed set of allowable categories such as FICTION, NON-FICTION, HORROR, etc. Using a tuple ensures that these values remain immutable throughout the program execution, maintaining consistency.

Tuples are also faster to access compared to lists and safeguard against accidental modification. This is particularly important for fixed data like genres, where the integrity of classification is essential for proper data entry validation and reporting.

Integration of Data Structures

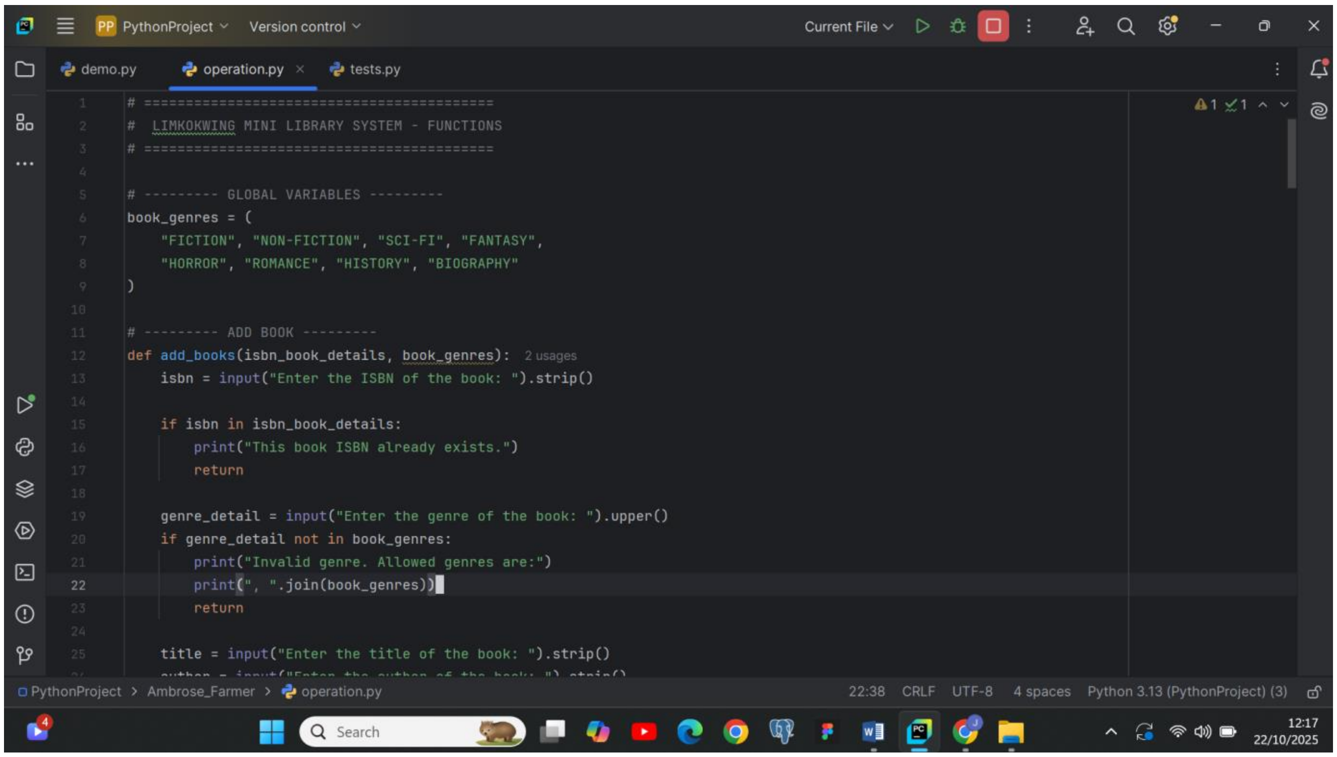
The system combines these three structures harmoniously. Dictionaries serve as containers for structured records, lists handle multiple similar entities, and tuples define unchangeable reference data. For example, when adding a book, the system verifies the entered genre against the tuple 'book_genres', then stores the book details in the 'isbn_book_details' dictionary. Likewise, when a member borrows a book, the book title is added to that member's 'Borrowed_Books' list, while the available copy count is updated within the dictionary.

This integration results in a clean, modular, and efficient structure that reflects real-world database relationships. It also provides a foundational model that can easily be extended to an actual database (e.g., SQL or Access) in the future.

Conclusion

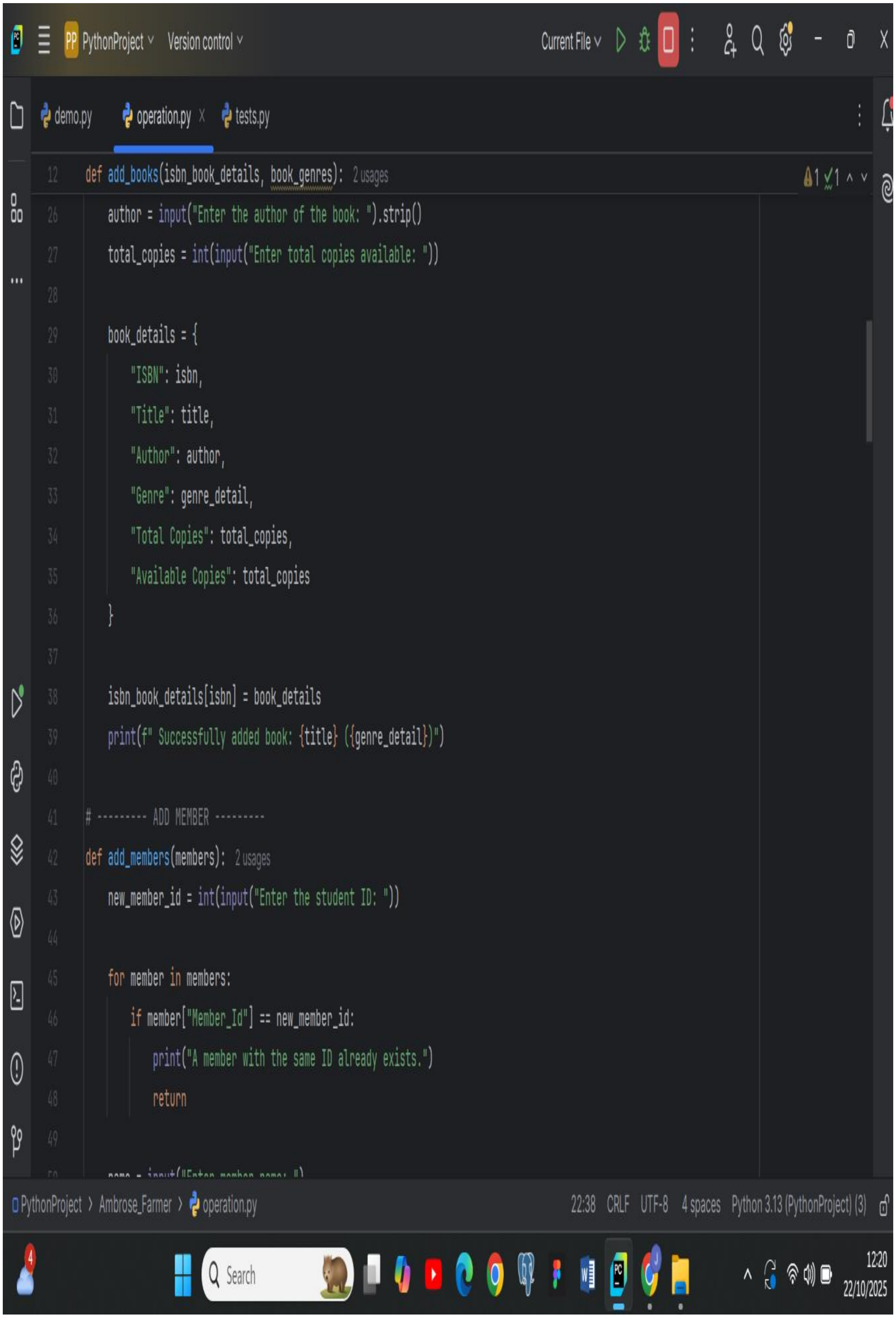
In conclusion, the combined use of lists, dictionaries, and tuples in the LIMKOKWING MINI LIBRARY SYSTEM provides an optimal balance of flexibility, performance, and data integrity. Each data type was purposefully chosen for its strengths: dictionaries for key-based access, lists for dynamic collections, and tuples for immutable reference data. Together, they form a robust foundation that aligns with good programming practices and structured data management principles.

operationo.py



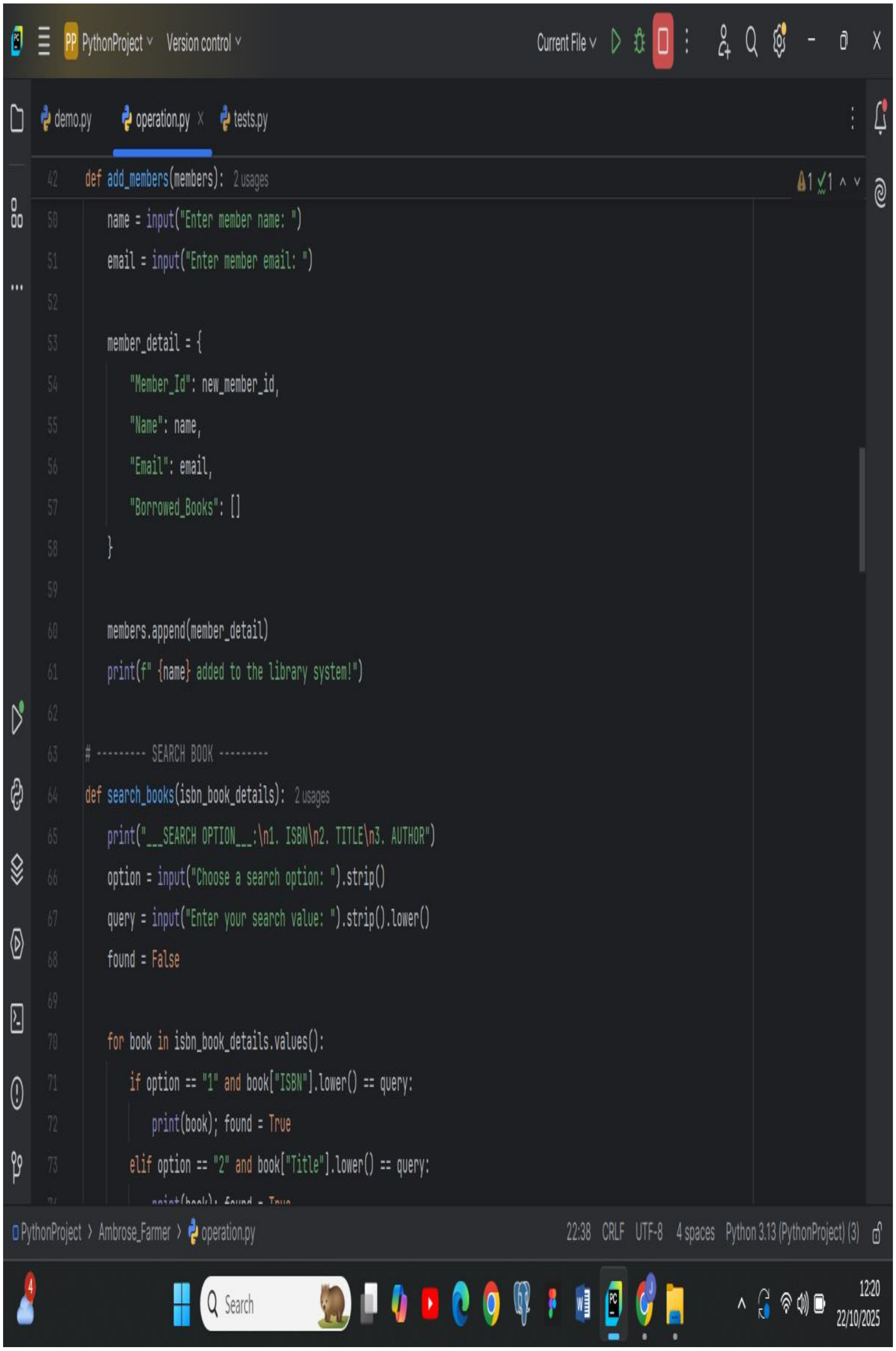
PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject > Ambrose_Farmer > operation.py
22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:17
22/10/2025

```python
# ==========================================
#  LIMKOKWING MINI LIBRARY SYSTEM - FUNCTIONS
# ==========================================

# --------- GLOBAL VARIABLES ---------
book_genres = (
    "FICTION", "NON-FICTION", "SCI-FI", "FANTASY",
    "HORROR", "ROMANCE", "HISTORY", "BIOGRAPHY"
)

# --------- ADD BOOK ---------
def add_books(isbn_book_details, book_genres):  2 usages
    isbn = input("Enter the ISBN of the book: ").strip()

    if isbn in isbn_book_details:
        print("This book ISBN already exists.")
        return

    genre_detail = input("Enter the genre of the book: ").upper()
    if genre_detail not in book_genres:
        print("Invalid genre. Allowed genres are:")
        print(", ".join(book_genres))
        return

    title = input("Enter the title of the book: ").strip()
    author = input("Enter the author of the book: ").strip()
```

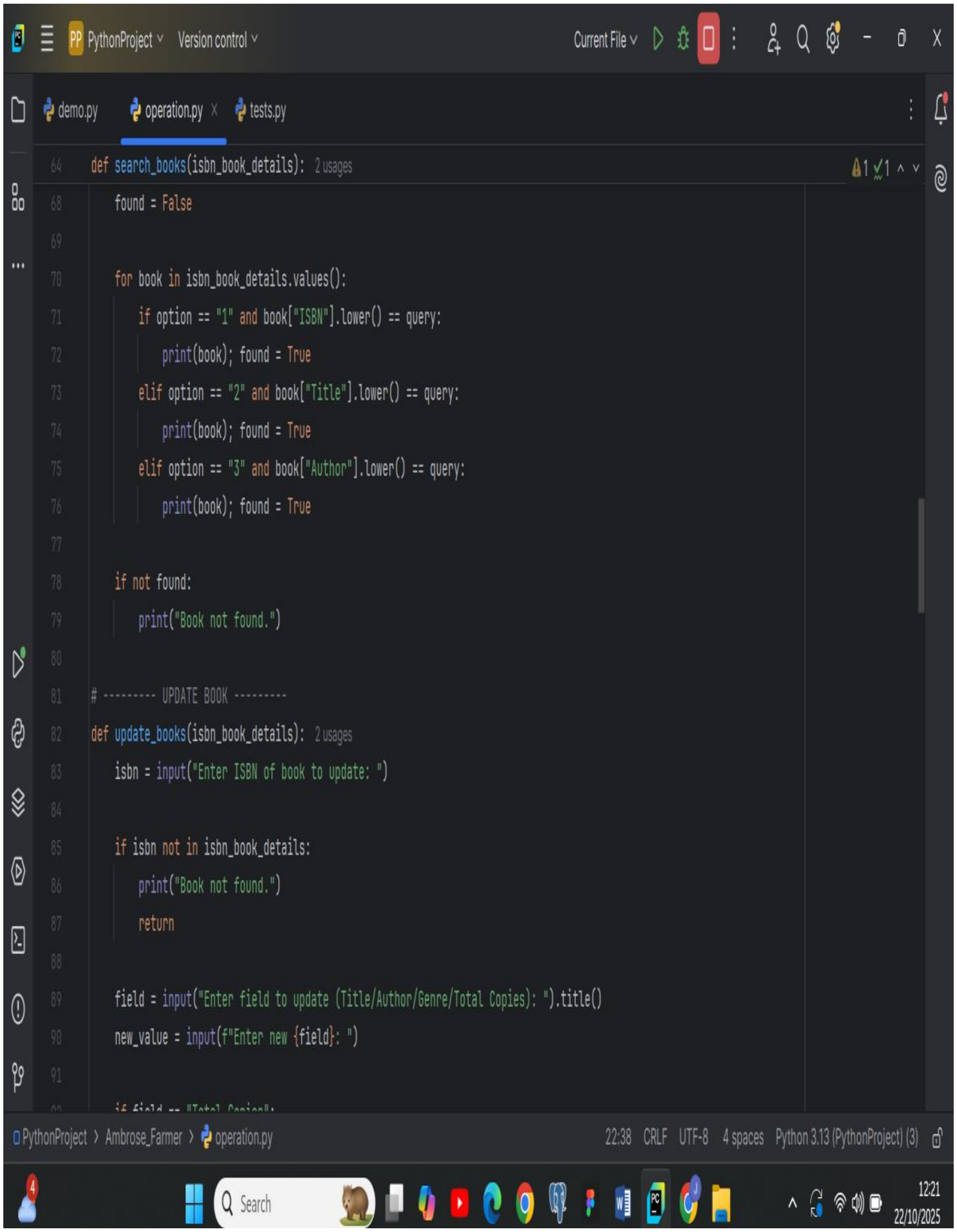


```python
def add_books(isbn_book_details, book_genres):  2 usages
    author = input("Enter the author of the book: ").strip()
    total_copies = int(input("Enter total copies available: "))

    book_details = {
        "ISBN": isbn,
        "Title": title,
        "Author": author,
        "Genre": genre_detail,
        "Total Copies": total_copies,
        "Available Copies": total_copies
    }

    isbn_book_details[isbn] = book_details
    print(f" Successfully added book: {title} ({genre_detail})")

# --------- ADD MEMBER ---------
def add_members(members):  2 usages
    new_member_id = int(input("Enter the student ID: "))

    for member in members:
        if member["Member_Id"] == new_member_id:
            print("A member with the same ID already exists.")
            return

```

PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject › Ambrose_Farmer › operation.py
22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:20
22/10/2025


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
2 usages
PythonProject › Ambrose_Farmer › operation.py
22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:20
22/10/2025

```python
def add_members(members):
    name = input("Enter member name: ")
    email = input("Enter member email: ")

    member_detail = {
        "Member_Id": new_member_id,
        "Name": name,
        "Email": email,
        "Borrowed_Books": []
    }

    members.append(member_detail)
    print(f" {name} added to the library system!")

# -------- SEARCH BOOK --------
def search_books(isbn_book_details):
    print("__SEARCH OPTION__:\n1. ISBN\n2. TITLE\n3. AUTHOR")
    option = input("Choose a search option: ").strip()
    query = input("Enter your search value: ").strip().lower()
    found = False

    for book in isbn_book_details.values():
        if option == "1" and book["ISBN"].lower() == query:
            print(book); found = True
        elif option == "2" and book["Title"].lower() == query:
            print(book); found = True
```



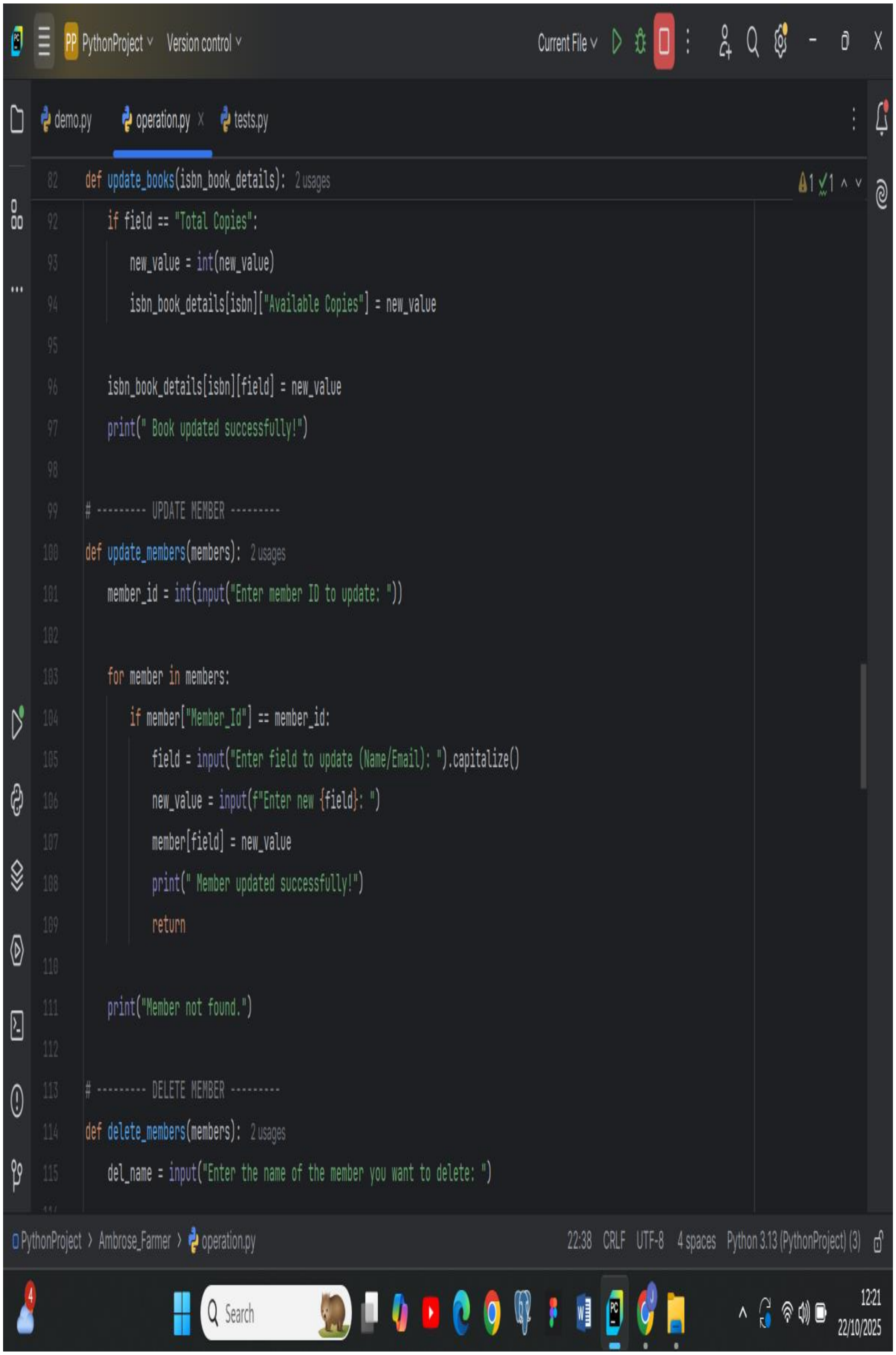
PythonProject ∨ Version control ∨ Current File ∨

demo.py operation.py tests.py

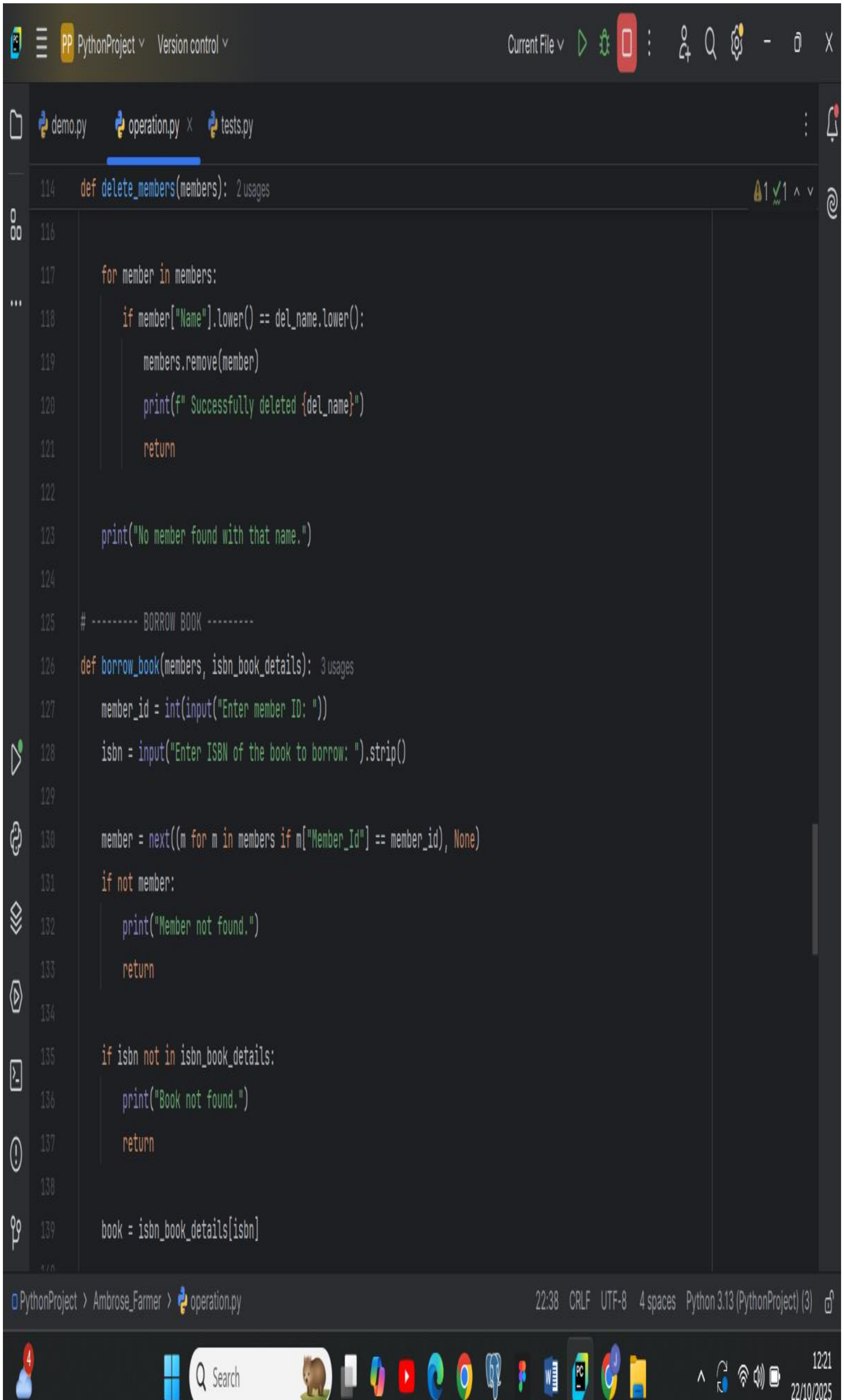
```
def search_books(isbn_book_details):  2 usages
    found = False

    for book in isbn_book_details.values():
        if option == "1" and book["ISBN"].lower() == query:
            print(book); found = True
        elif option == "2" and book["Title"].lower() == query:
            print(book); found = True
        elif option == "3" and book["Author"].lower() == query:
            print(book); found = True

    if not found:
        print("Book not found.")

# -------- UPDATE BOOK --------
def update_books(isbn_book_details):  2 usages
    isbn = input("Enter ISBN of book to update: ")

    if isbn not in isbn_book_details:
        print("Book not found.")
        return

    field = input("Enter field to update (Title/Author/Genre/Total Copies): ").title()
    new_value = input(f"Enter new {field}: ")
```

PythonProject › Ambrose_Farmer › operation.py   22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)

12:21 22/10/2025


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject › Ambrose_Farmer › operation.py
22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:21
22/10/2025

```python
def update_books(isbn_book_details):  2 usages
    if field == "Total Copies":
        new_value = int(new_value)
        isbn_book_details[isbn]["Available Copies"] = new_value

    isbn_book_details[isbn][field] = new_value
    print(" Book updated successfully!")

# -------- UPDATE MEMBER --------
def update_members(members):  2 usages
    member_id = int(input("Enter member ID to update: "))

    for member in members:
        if member["Member_Id"] == member_id:
            field = input("Enter field to update (Name/Email): ").capitalize()
            new_value = input(f"Enter new {field}: ")
            member[field] = new_value
            print(" Member updated successfully!")
            return

    print("Member not found.")

# -------- DELETE MEMBER --------
def delete_members(members):  2 usages
    del_name = input("Enter the name of the member you want to delete: ")
```

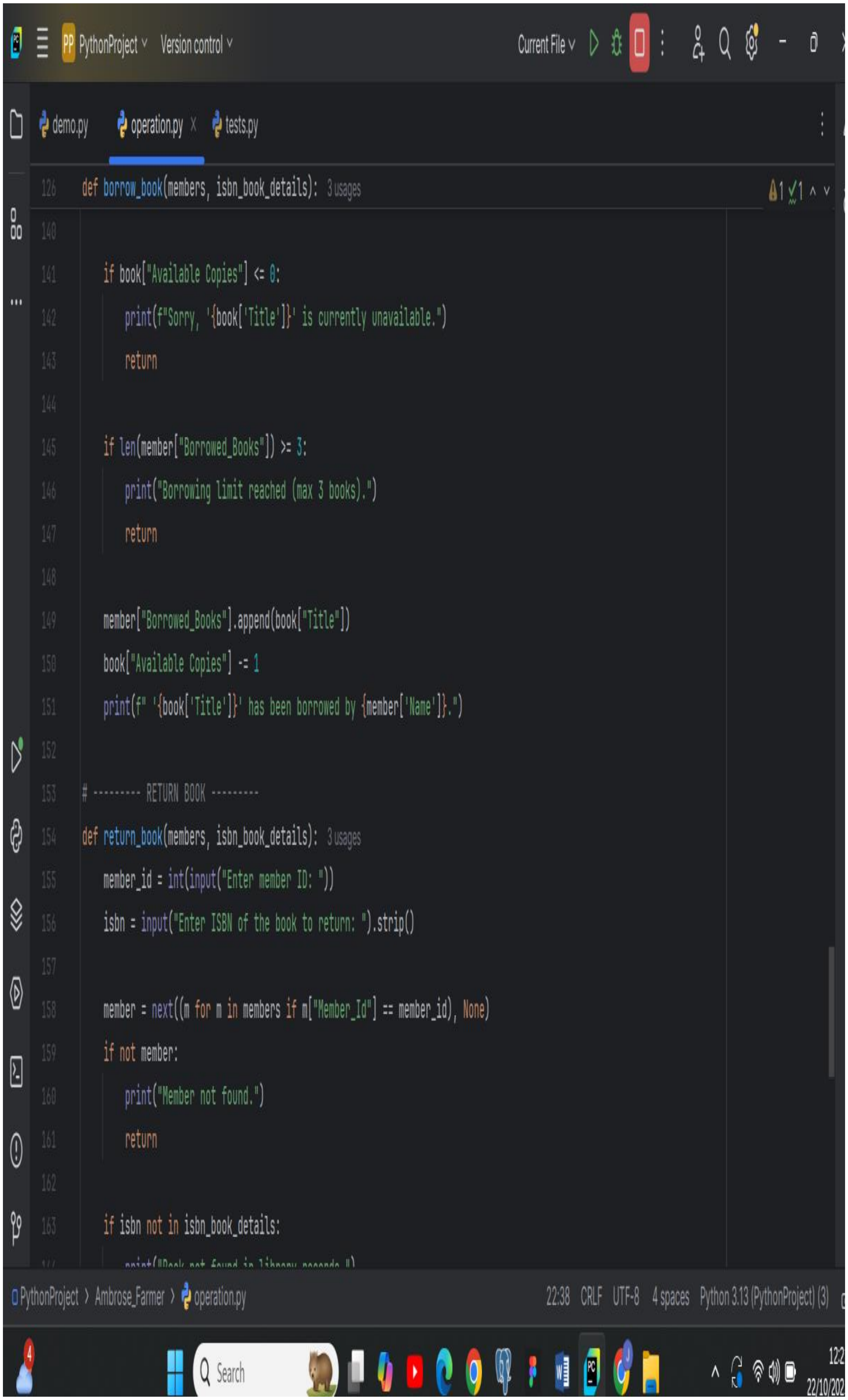

PythonProject   Version control   Current File

demo.py   operation.py   tests.py

```python
def delete_members(members):  2 usages

    for member in members:
        if member["Name"].lower() == del_name.lower():
            members.remove(member)
            print(f" Successfully deleted {del_name}")
            return

    print("No member found with that name.")

# -------- BORROW BOOK --------
def borrow_book(members, isbn_book_details):  3 usages
    member_id = int(input("Enter member ID: "))
    isbn = input("Enter ISBN of the book to borrow: ").strip()

    member = next((m for m in members if m["Member_Id"] == member_id), None)
    if not member:
        print("Member not found.")
        return

    if isbn not in isbn_book_details:
        print("Book not found.")
        return

    book = isbn_book_details[isbn]
```

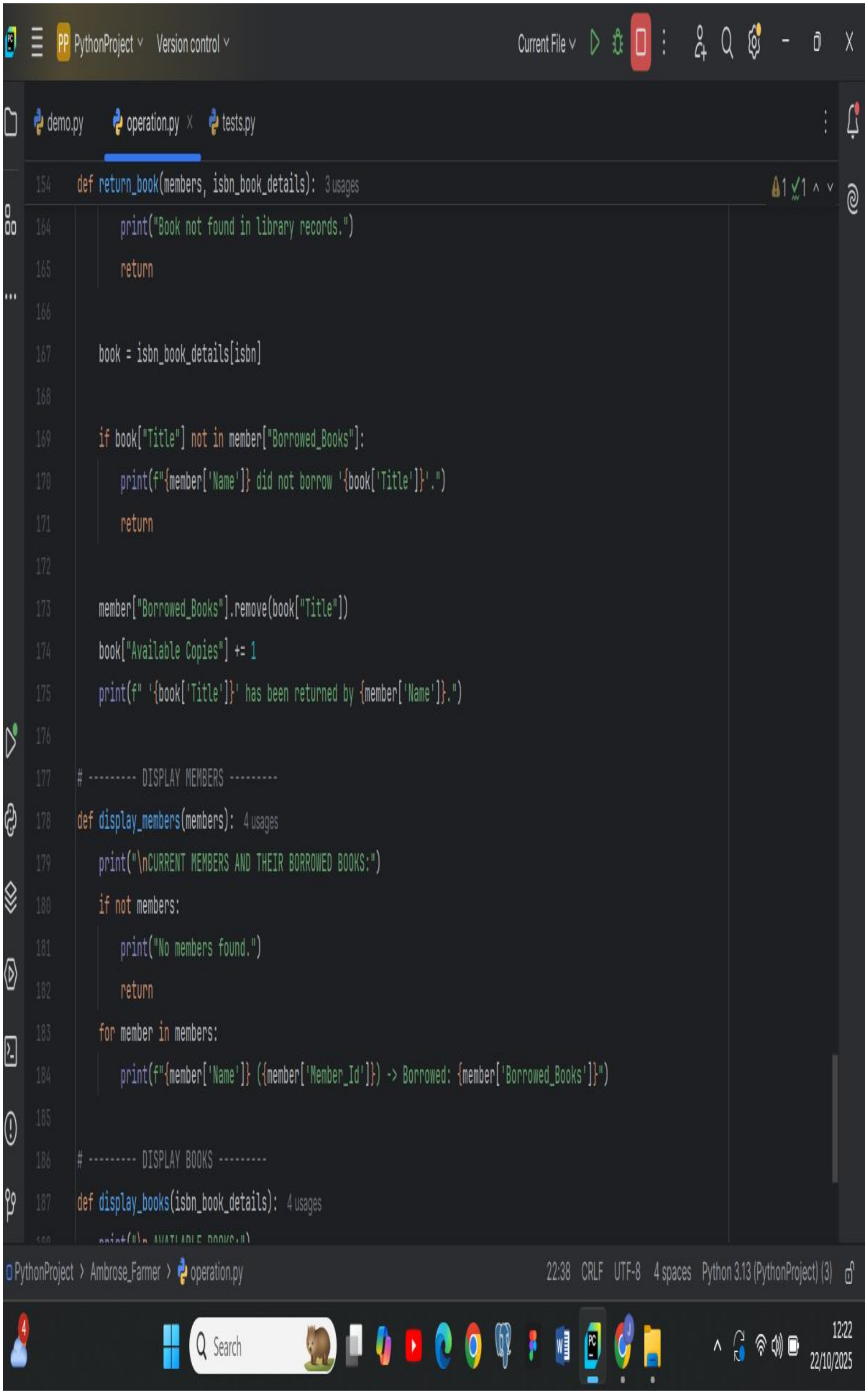
PythonProject > Ambrose_Farmer > operation.py   22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)

Search   12:21 22/10/2025

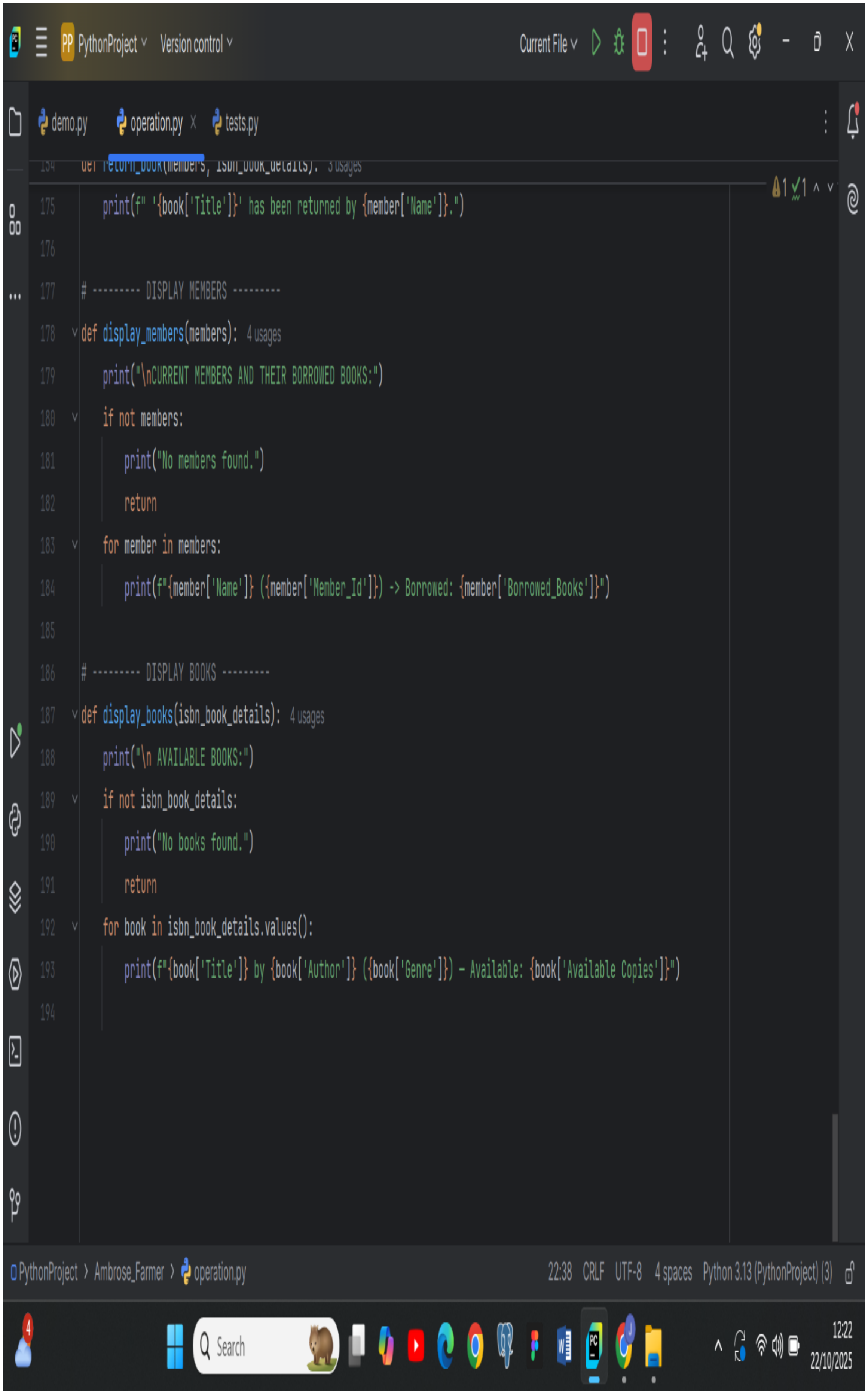


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
3 usages
PythonProject › Ambrose_Farmer › operation.py
22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
22/10/202

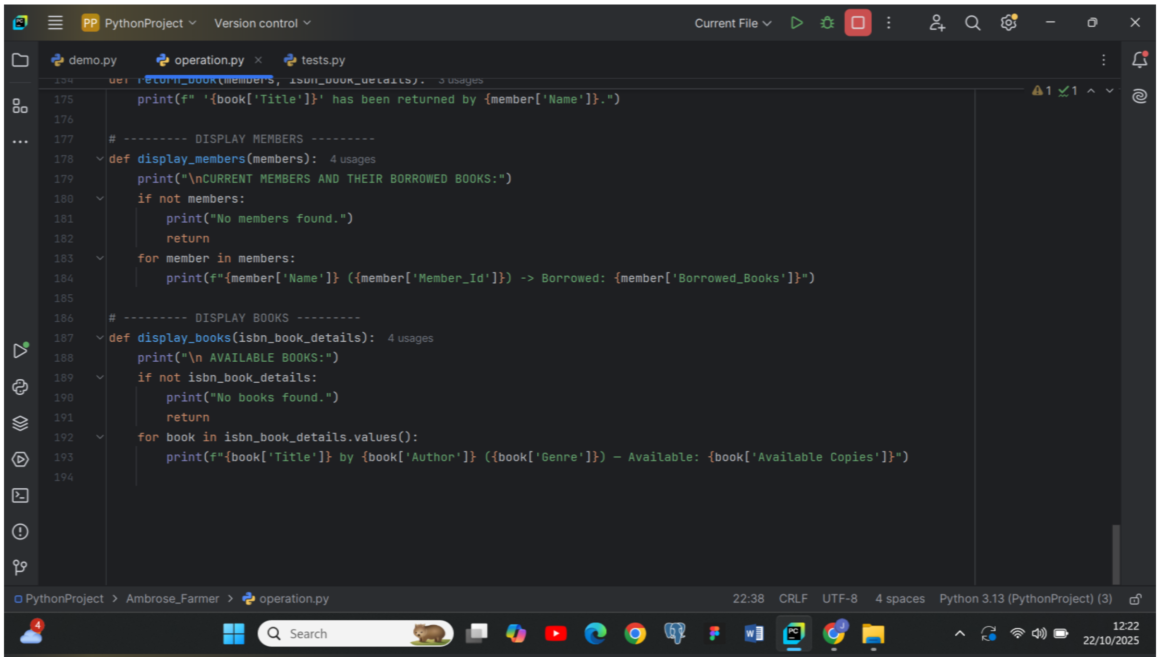
```python
def borrow_book(members, isbn_book_details):

    if book["Available Copies"] <= 0:
        print(f"Sorry, '{book['Title']}' is currently unavailable.")
        return

    if len(member["Borrowed_Books"]) >= 3:
        print("Borrowing limit reached (max 3 books).")
        return

    member["Borrowed_Books"].append(book["Title"])
    book["Available Copies"] -= 1
    print(f"'{book['Title']}' has been borrowed by {member['Name']}.")

# --------- RETURN BOOK ---------
def return_book(members, isbn_book_details):
    member_id = int(input("Enter member ID: "))
    isbn = input("Enter ISBN of the book to return: ").strip()

    member = next((m for m in members if m["Member_Id"] == member_id), None)
    if not member:
        print("Member not found.")
        return

    if isbn not in isbn_book_details:
        print("Book not found in library records.")
```



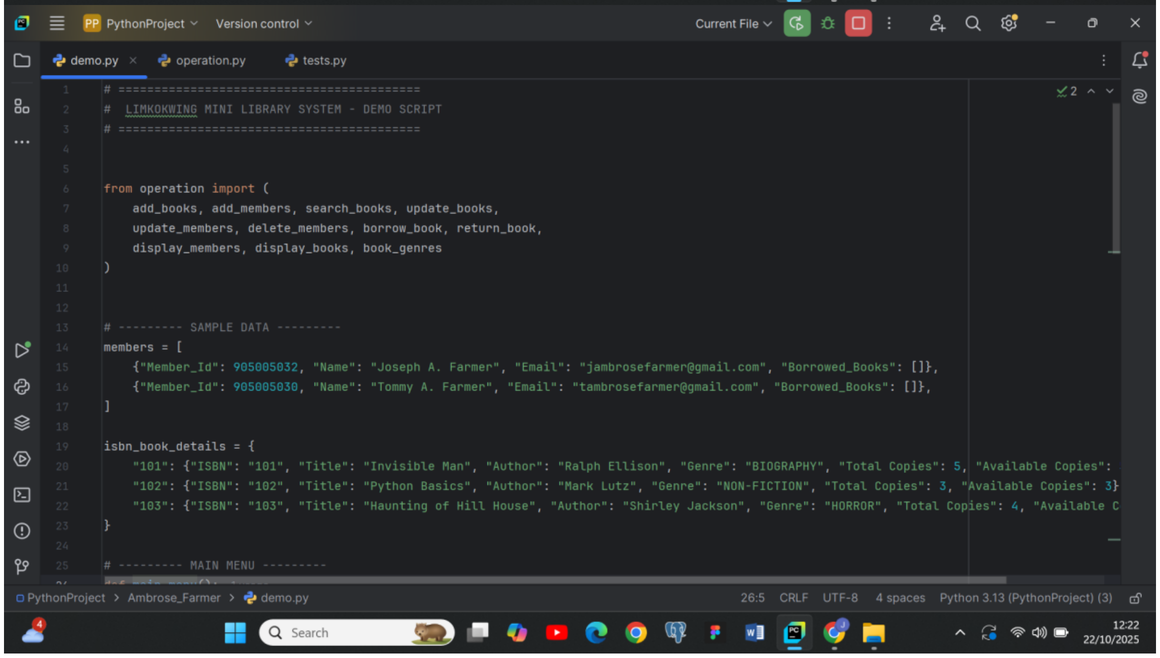
```python
def return_book(members, isbn_book_details):
        print("Book not found in library records.")
        return

    book = isbn_book_details[isbn]

    if book["Title"] not in member["Borrowed_Books"]:
        print(f"{member['Name']} did not borrow '{book['Title']}'.")
        return

    member["Borrowed_Books"].remove(book["Title"])
    book["Available Copies"] += 1
    print(f" '{book['Title']}' has been returned by {member['Name']}.")

# -------- DISPLAY MEMBERS --------
def display_members(members):
    print("\nCURRENT MEMBERS AND THEIR BORROWED BOOKS:")
    if not members:
        print("No members found.")
        return
    for member in members:
        print(f"{member['Name']} ({member['Member_Id']}) -> Borrowed: {member['Borrowed_Books']}")

# -------- DISPLAY BOOKS --------
def display_books(isbn_book_details):
    print("\n AVAILABLE BOOKS:")
```


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject › Ambrose_Farmer › operation.py
22:38 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:22
22/10/2025

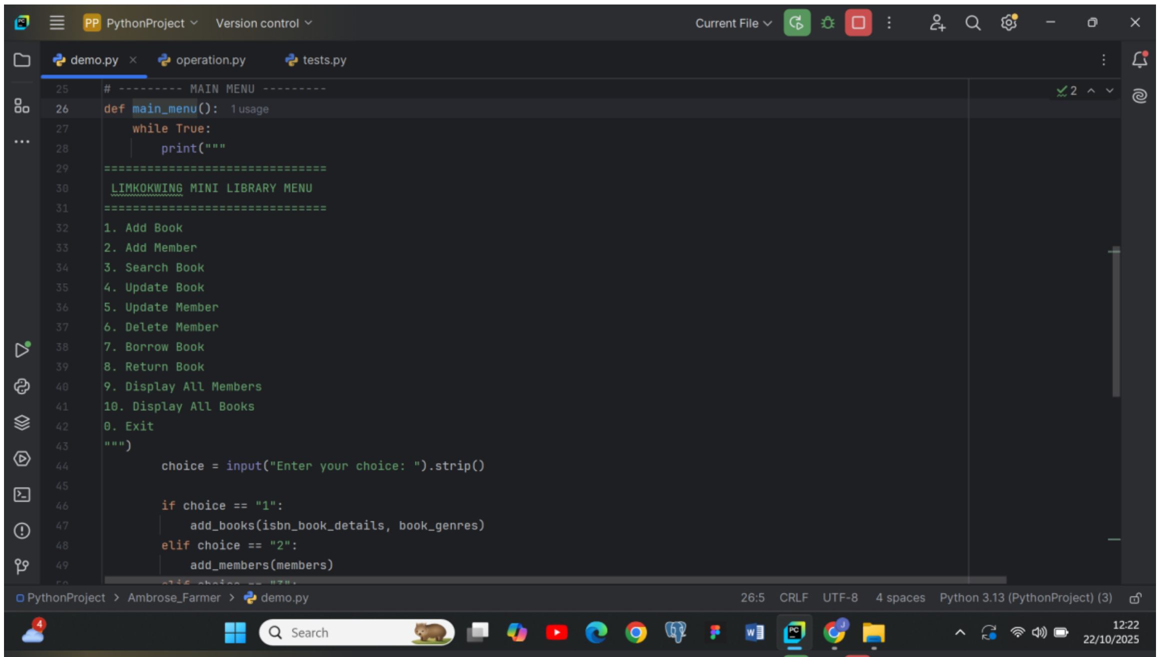
```python
def return_book(members, isbn_book_details):  3 usages
    print(f" '{book['Title']}' has been returned by {member['Name']}.")

# -------- DISPLAY MEMBERS --------
def display_members(members):  4 usages
    print("\nCURRENT MEMBERS AND THEIR BORROWED BOOKS:")
    if not members:
        print("No members found.")
        return
    for member in members:
        print(f"{member['Name']} ({member['Member_Id']}) -> Borrowed: {member['Borrowed_Books']}")

# -------- DISPLAY BOOKS --------
def display_books(isbn_book_details):  4 usages
    print("\n AVAILABLE BOOKS:")
    if not isbn_book_details:
        print("No books found.")
        return
    for book in isbn_book_details.values():
        print(f"{book['Title']} by {book['Author']} ({book['Genre']}) - Available: {book['Available Copies']}")
```



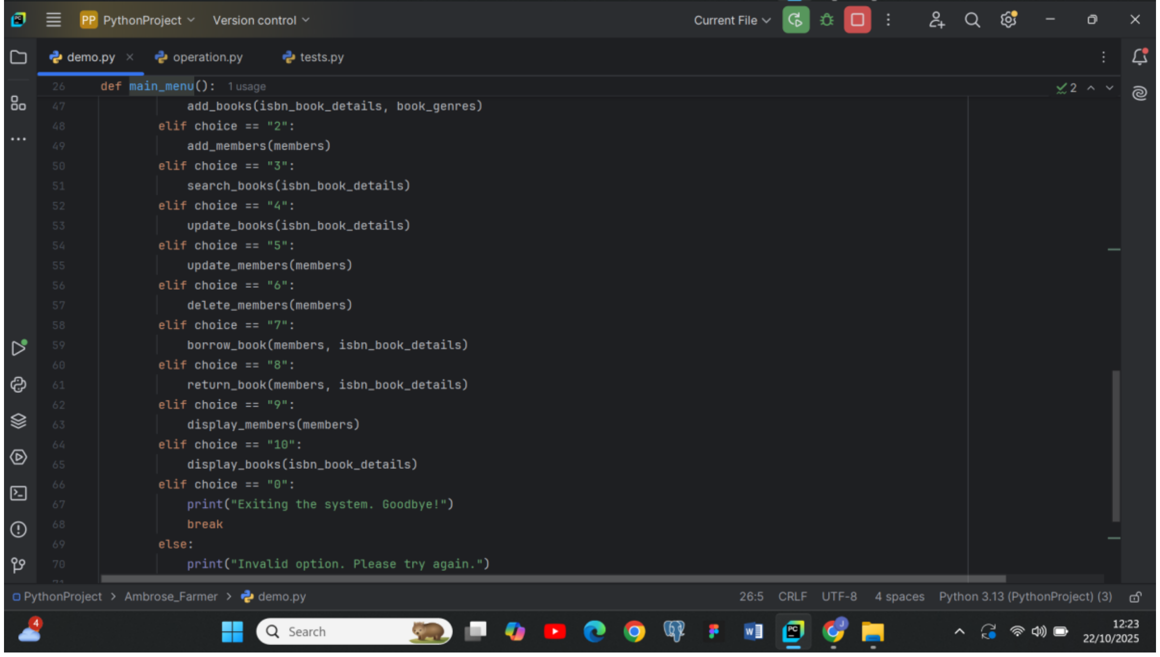
```
    print(f" '{book['Title']}' has been returned by {member['Name']}.")

# --------- DISPLAY MEMBERS ---------
def display_members(members):  4 usages
    print("\nCURRENT MEMBERS AND THEIR BORROWED BOOKS:")
    if not members:
        print("No members found.")
        return
    for member in members:
        print(f"{member['Name']} ({member['Member_Id']}) -> Borrowed: {member['Borrowed_Books']}")

# --------- DISPLAY BOOKS ---------
def display_books(isbn_book_details):  4 usages
    print("\n AVAILABLE BOOKS:")
    if not isbn_book_details:
        print("No books found.")
        return
    for book in isbn_book_details.values():
        print(f"{book['Title']} by {book['Author']} ({book['Genre']}) — Available: {book['Available Copies']}")
```



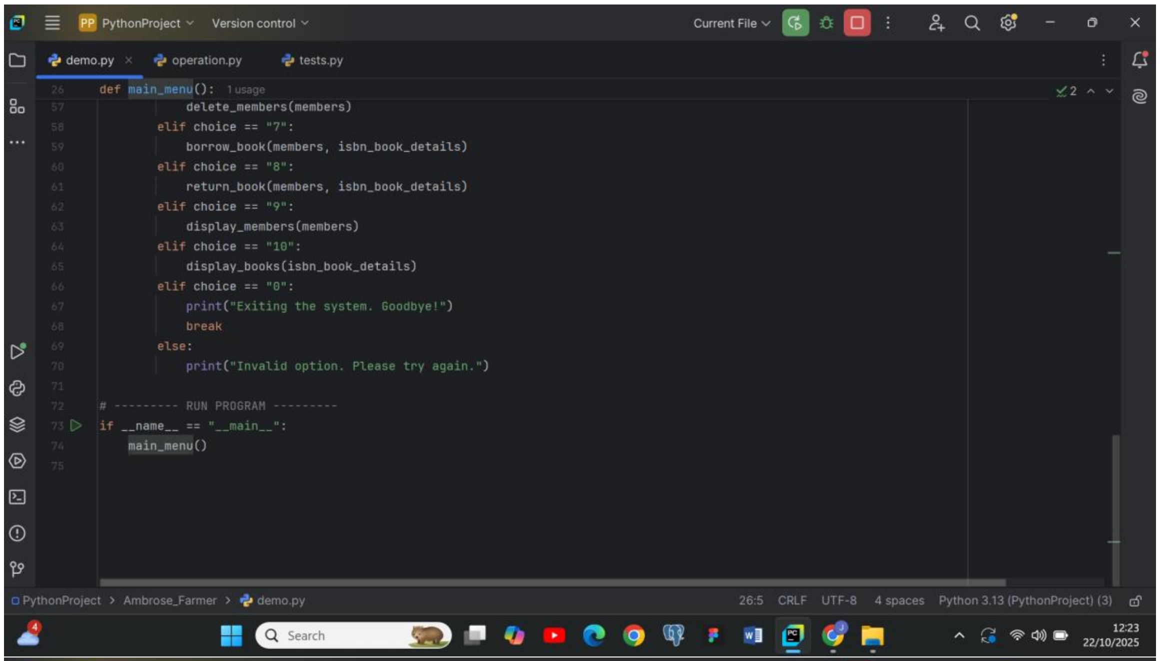
```
# ========================================
#  LIMKOKWING MINI LIBRARY SYSTEM - DEMO SCRIPT
# ========================================


from operation import (
    add_books, add_members, search_books, update_books,
    update_members, delete_members, borrow_book, return_book,
    display_members, display_books, book_genres
)


# --------- SAMPLE DATA ---------
members = [
    {"Member_Id": 905005032, "Name": "Joseph A. Farmer", "Email": "jambrosefarmer@gmail.com", "Borrowed_Books": []},
    {"Member_Id": 905005030, "Name": "Tommy A. Farmer", "Email": "tambrosefarmer@gmail.com", "Borrowed_Books": []},
]

isbn_book_details = {
    "101": {"ISBN": "101", "Title": "Invisible Man", "Author": "Ralph Ellison", "Genre": "BIOGRAPHY", "Total Copies": 5, "Available Copies":
    "102": {"ISBN": "102", "Title": "Python Basics", "Author": "Mark Lutz", "Genre": "NON-FICTION", "Total Copies": 3, "Available Copies": 3}
    "103": {"ISBN": "103", "Title": "Haunting of Hill House", "Author": "Shirley Jackson", "Genre": "HORROR", "Total Copies": 4, "Available C
}

# --------- MAIN MENU ---------
```



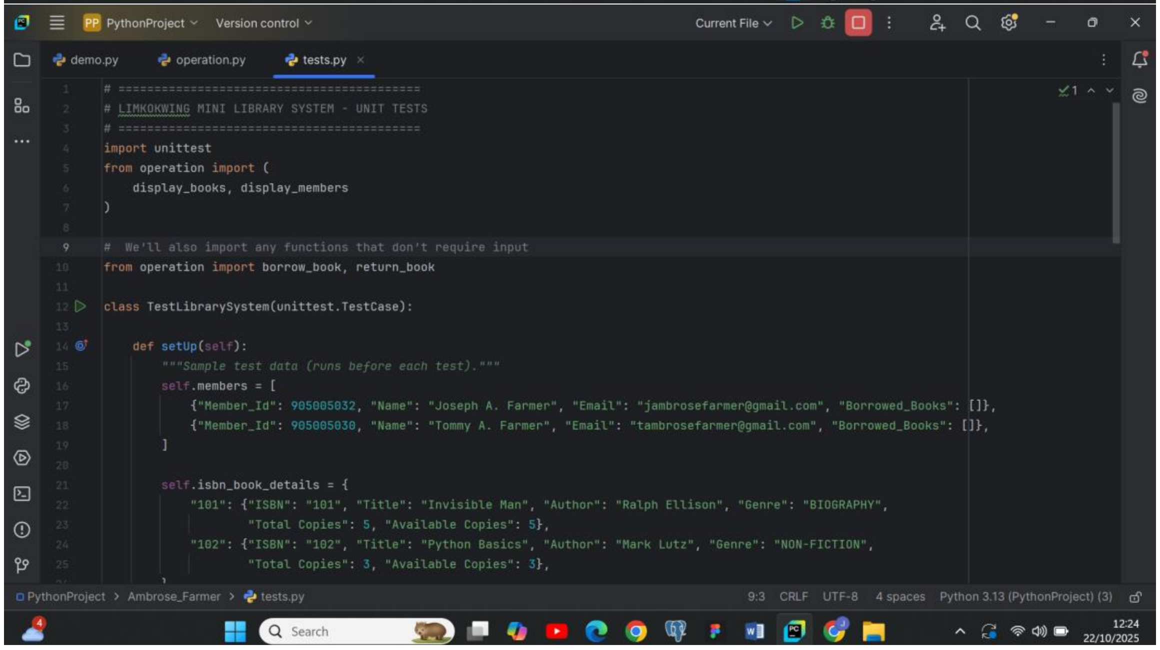
```python
# --------- MAIN MENU ---------
def main_menu():
    while True:
        print("""
==============================
 LIMKOKWING MINI LIBRARY MENU
==============================
1. Add Book
2. Add Member
3. Search Book
4. Update Book
5. Update Member
6. Delete Member
7. Borrow Book
8. Return Book
9. Display All Members
10. Display All Books
0. Exit
""")
        choice = input("Enter your choice: ").strip()

        if choice == "1":
            add_books(isbn_book_details, book_genres)
        elif choice == "2":
            add_members(members)
```



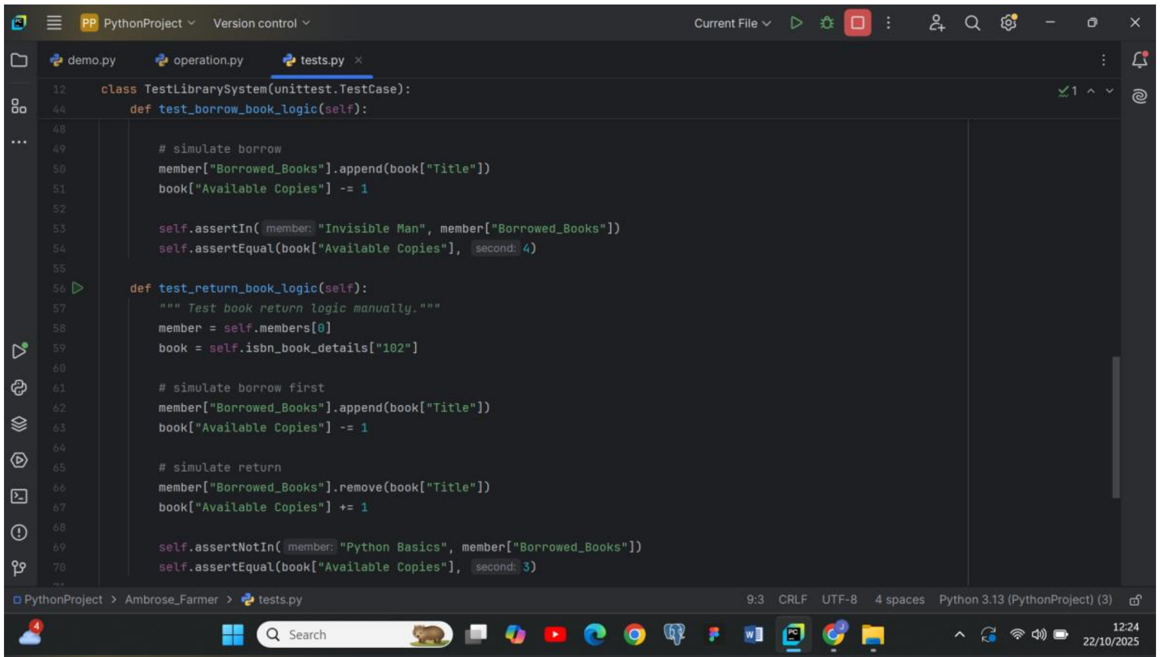
```python
def main_menu():
            add_books(isbn_book_details, book_genres)
        elif choice == "2":
            add_members(members)
        elif choice == "3":
            search_books(isbn_book_details)
        elif choice == "4":
            update_books(isbn_book_details)
        elif choice == "5":
            update_members(members)
        elif choice == "6":
            delete_members(members)
        elif choice == "7":
            borrow_book(members, isbn_book_details)
        elif choice == "8":
            return_book(members, isbn_book_details)
        elif choice == "9":
            display_members(members)
        elif choice == "10":
            display_books(isbn_book_details)
        elif choice == "0":
            print("Exiting the system. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")
```


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject > Ambrose_Farmer > demo.py
26:5 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:23
22/10/2025

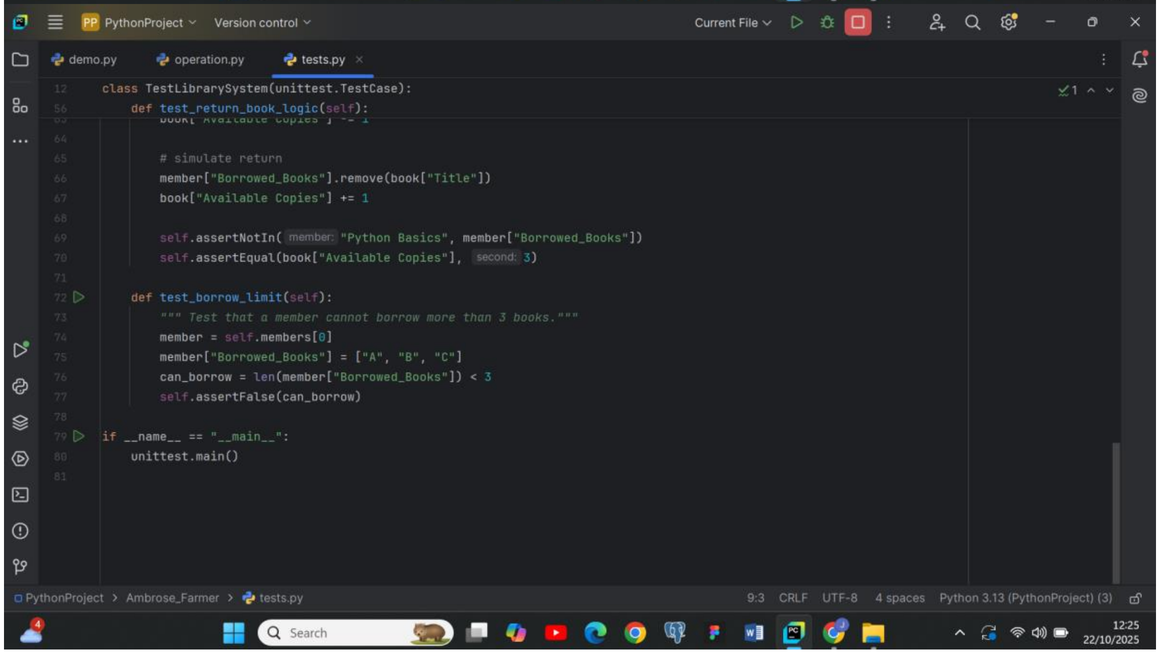
```python
def main_menu():  1 usage
            delete_members(members)
        elif choice == "7":
            borrow_book(members, isbn_book_details)
        elif choice == "8":
            return_book(members, isbn_book_details)
        elif choice == "9":
            display_members(members)
        elif choice == "10":
            display_books(isbn_book_details)
        elif choice == "0":
            print("Exiting the system. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")

# --------- RUN PROGRAM ---------
if __name__ == "__main__":
    main_menu()
```


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject > Ambrose_Farmer > tests.py
9:3 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
Search
12:24
22/10/2025

```python
# ==========================================
# LIMKOKWING MINI LIBRARY SYSTEM - UNIT TESTS
# ==========================================
import unittest
from operation import (
    display_books, display_members
)

#  We'll also import any functions that don't require input
from operation import borrow_book, return_book

class TestLibrarySystem(unittest.TestCase):

    def setUp(self):
        """Sample test data (runs before each test)."""
        self.members = [
            {"Member_Id": 905005032, "Name": "Joseph A. Farmer", "Email": "jambrosefarmer@gmail.com", "Borrowed_Books": []},
            {"Member_Id": 905005030, "Name": "Tommy A. Farmer", "Email": "tambrosefarmer@gmail.com", "Borrowed_Books": []},
        ]

        self.isbn_book_details = {
            "101": {"ISBN": "101", "Title": "Invisible Man", "Author": "Ralph Ellison", "Genre": "BIOGRAPHY",
                    "Total Copies": 5, "Available Copies": 5},
            "102": {"ISBN": "102", "Title": "Python Basics", "Author": "Mark Lutz", "Genre": "NON-FICTION",
                    "Total Copies": 3, "Available Copies": 3},
```


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject > Ambrose_Farmer > tests.py
9:3 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
12:24
22/10/2025

```python
class TestLibrarySystem(unittest.TestCase):
    def test_borrow_book_logic(self):

        # simulate borrow
        member["Borrowed_Books"].append(book["Title"])
        book["Available Copies"] -= 1

        self.assertIn("Invisible Man", member["Borrowed_Books"])
        self.assertEqual(book["Available Copies"], 4)

    def test_return_book_logic(self):
        """ Test book return logic manually."""
        member = self.members[0]
        book = self.isbn_book_details["102"]

        # simulate borrow first
        member["Borrowed_Books"].append(book["Title"])
        book["Available Copies"] -= 1

        # simulate return
        member["Borrowed_Books"].remove(book["Title"])
        book["Available Copies"] += 1

        self.assertNotIn("Python Basics", member["Borrowed_Books"])
        self.assertEqual(book["Available Copies"], 3)
```


PythonProject
Version control
Current File
demo.py
operation.py
tests.py
PythonProject > Ambrose_Farmer > tests.py
9:3 CRLF UTF-8 4 spaces Python 3.13 (PythonProject) (3)
12:25
22/10/2025

```python
class TestLibrarySystem(unittest.TestCase):
    def test_return_book_logic(self):

        # simulate return
        member["Borrowed_Books"].remove(book["Title"])
        book["Available Copies"] += 1

        self.assertNotIn("Python Basics", member["Borrowed_Books"])
        self.assertEqual(book["Available Copies"], 3)

    def test_borrow_limit(self):
        """ Test that a member cannot borrow more than 3 books."""
        member = self.members[0]
        member["Borrowed_Books"] = ["A", "B", "C"]
        can_borrow = len(member["Borrowed_Books"]) < 3
        self.assertFalse(can_borrow)

if __name__ == "__main__":
    unittest.main()
```