ASSIGNMENT : 1

TITLE: INTRODUCTION TO OOPS CONCEPTS

PROGRAM CODE:

class Book:

    def \_\_init\_\_(self, book\_id, title, author):

        self.book\_id = book\_id

        self.title = title

        self.author = author

        self.is\_borrowed = False

    def \_\_str\_\_(self):

        status = "Borrowed" if self.is\_borrowed else "Available"

        return f"{self.book\_id}: '{self.title}' by {self.author} - {status}"

class Patron:

    def \_\_init\_\_(self, patron\_id, name):

        self.patron\_id = patron\_id

        self.name = name

        self.borrowed\_books = []

    def \_\_str\_\_(self):

        borrowed\_ids = [book.book\_id for book in self.borrowed\_books]

        return f"{self.patron\_id}: {self.name} (Borrowed books: {borrowed\_ids})"

class Library:

    def \_\_init\_\_(self):

        self.books = {}

        self.patrons = {}

    def add\_book(self):

        book\_id = input("Enter book ID: ")

        if book\_id in self.books:

            print(f"Book ID '{book\_id}' already exists.")

            return

        title = input("Enter book title: ")

        author = input("Enter author name: ")

        self.books[book\_id] = Book(book\_id, title, author)

        print(f"Book '{title}' added successfully.")

    def register\_patron(self):

        patron\_id = input("Enter patron ID: ")

        if patron\_id in self.patrons:

            print(f"Patron ID '{patron\_id}' already exists.")

            return

        name = input("Enter patron name: ")

        self.patrons[patron\_id] = Patron(patron\_id, name)

        print(f"Patron '{name}' registered successfully.")

    def borrow\_book(self):

        patron\_id = input("Enter patron ID: ")

        if patron\_id not in self.patrons:

            print(f"No patron found with ID '{patron\_id}'.")

            return

        book\_id = input("Enter book ID: ")

        if book\_id not in self.books:

            print(f"No book found with ID '{book\_id}'.")

            return

        patron = self.patrons[patron\_id]

        book = self.books[book\_id]

        if book.is\_borrowed:

            print(f"Book '{book.title}' is already borrowed.")

        else:

            book.is\_borrowed = True

            patron.borrowed\_books.append(book)

            print(f"Patron '{patron.name}' borrowed book '{book.title}'.")

    def return\_book(self):

        patron\_id = input("Enter patron ID: ")

        if patron\_id not in self.patrons:

            print(f"No patron found with ID '{patron\_id}'.")

            return

        book\_id = input("Enter book ID: ")

        if book\_id not in self.books:

            print(f"No book found with ID '{book\_id}'.")

            return

        patron = self.patrons[patron\_id]

        book = self.books[book\_id]

        if book in patron.borrowed\_books:

            patron.borrowed\_books.remove(book)

            book.is\_borrowed = False

            print(f"Patron '{patron.name}' returned book '{book.title}'.")

        else:

            print(f"Patron '{patron.name}' has not borrowed book '{book.title}'.")

    def list\_books(self):

        if not self.books:

            print("No books in library.")

            return

        print("\nList of Books:")

        for book in self.books.values():

            print(book)

    def list\_patrons(self):

        if not self.patrons:

            print("No patrons registered.")

            return

        print("\nList of Patrons:")

        for patron in self.patrons.values():

            print(patron)

def main():

    library = Library()

    while True:

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

        print("1. Add New Book")

        print("2. Register New Patron")

        print("3. Borrow Book")

        print("4. Return Book")

        print("5. List All Books")

        print("6. List All Patrons")

        print("7. Exit")

        choice = input("Enter your choice (1-7): ")

        if choice == "1":

            library.add\_book()

        elif choice == "2":

            library.register\_patron()

        elif choice == "3":

            library.borrow\_book()

        elif choice == "4":

            library.return\_book()

        elif choice == "5":

            library.list\_books()

        elif choice == "6":

            library.list\_patrons()

        elif choice == "7":

            print("Exiting the program. Goodbye!")

            break

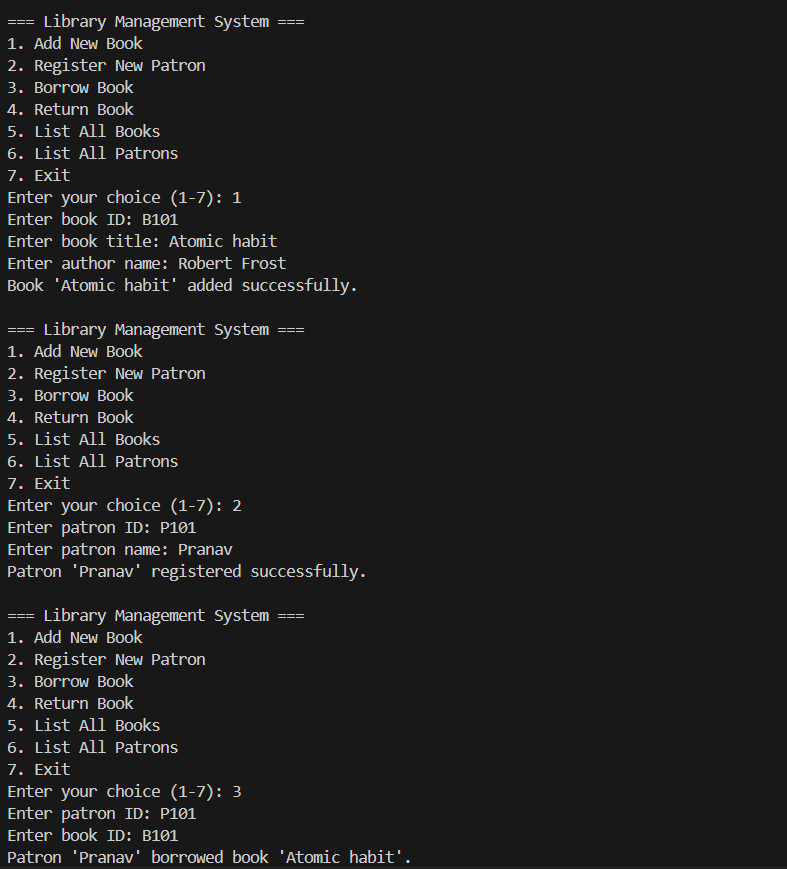
        else:

            print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT:



BUCKET LIST EXPERIMENT TITLE:

4. Library Book Borrowing

Scenario: Model a library system where members can borrow books. Each

book has a title, author, and availability status. A member should be able to

borrow/return books and view their borrowed list.

Rubrics:

● Use two classes: Book and Member.

● Use composition (Member has Book).

● Track book status (borrowed/available).

● Implement borrowing/returning logic.

PROGRAM CODE:

class Book:

    def \_\_init\_\_(self, title, author):

        self.title = title

        self.author = author

        self.is\_borrowed = False

    def \_\_str\_\_(self):

        status = "Borrowed" if self.is\_borrowed else "Available"

        return f"'{self.title}' by {self.author} - {status}"

class Member:

    def \_\_init\_\_(self, name):

        self.name = name

        self.borrowed\_books = []  # Composition: Member has Book(s)

    def borrow\_book(self, book):

        if book.is\_borrowed:

            print(f"Sorry, '{book.title}' is already borrowed.")

        else:

            book.is\_borrowed = True

            self.borrowed\_books.append(book)

            print(f"{self.name} successfully borrowed '{book.title}'.")

    def return\_book(self, book):

        if book in self.borrowed\_books:

            book.is\_borrowed = False

            self.borrowed\_books.remove(book)

            print(f"{self.name} successfully returned '{book.title}'.")

        else:

            print(f"{self.name} did not borrow '{book.title}'.")

    def view\_borrowed\_books(self):

        if not self.borrowed\_books:

            print(f"{self.name} has not borrowed any books.")

        else:

            print(f"{self.name}'s borrowed books:")

            for book in self.borrowed\_books:

                print(f"- {book.title} by {book.author}")

def main():

    books = []   # list of all Book objects

    members = [] # list of all Member objects

    while True:

        print("\n=== Library Menu ===")

        print("1. Add new book")

        print("2. Register new member")

        print("3. Borrow a book")

        print("4. Return a book")

        print("5. View all books")

        print("6. View member's borrowed books")

        print("7. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":

            title = input("Enter book title: ")

            author = input("Enter book author: ")

            books.append(Book(title, author))

            print(f"Book '{title}' by {author} added successfully.")

        elif choice == "2":

            name = input("Enter member's name: ")

            members.append(Member(name))

            print(f"Member '{name}' registered successfully.")

        elif choice == "3":

            if not members:

                print("No members registered yet.")

                continue

            if not books:

                print("No books in the library yet.")

                continue

            print("\nSelect member:")

            for idx, member in enumerate(members, start=1):

                print(f"{idx}. {member.name}")

            try:

                member\_idx = int(input("Enter member number: ")) - 1

                member = members[member\_idx]

            except (ValueError, IndexError):

                print("Invalid member selection.")

                continue

            print("\nSelect book to borrow:")

            for idx, book in enumerate(books, start=1):

                print(f"{idx}. {book}")

            try:

                book\_idx = int(input("Enter book number: ")) - 1

                book = books[book\_idx]

                member.borrow\_book(book)

            except (ValueError, IndexError):

                print("Invalid book selection.")

        elif choice == "4":

            if not members:

                print("No members registered yet.")

                continue

            print("\nSelect member:")

            for idx, member in enumerate(members, start=1):

                print(f"{idx}. {member.name}")

            try:

                member\_idx = int(input("Enter member number: ")) - 1

                member = members[member\_idx]

            except (ValueError, IndexError):

                print("Invalid member selection.")

                continue

            if not member.borrowed\_books:

                print(f"{member.name} has no borrowed books.")

                continue

            print("\nSelect book to return:")

            for idx, book in enumerate(member.borrowed\_books, start=1):

                print(f"{idx}. {book.title} by {book.author}")

            try:

                book\_idx = int(input("Enter book number: ")) - 1

                book = member.borrowed\_books[book\_idx]

                member.return\_book(book)

            except (ValueError, IndexError):

                print("Invalid selection.")

        elif choice == "5":

            if not books:

                print("No books in the library yet.")

            else:

                print("\nAll Books:")

                for idx, book in enumerate(books, start=1):

                    print(f"{idx}. {book}")

        elif choice == "6":

            if not members:

                print("No members registered yet.")

                continue

            print("\nSelect member to view borrowed books:")

            for idx, member in enumerate(members, start=1):

                print(f"{idx}. {member.name}")

            try:

                member\_idx = int(input("Enter member number: ")) - 1

                member = members[member\_idx]

                member.view\_borrowed\_books()

            except (ValueError, IndexError):

                print("Invalid member selection.")

        elif choice == "7":

            print("Exiting the program. Goodbye!")

            break

        else:

            print("Invalid choice. Please enter a number from 1 to 7.")

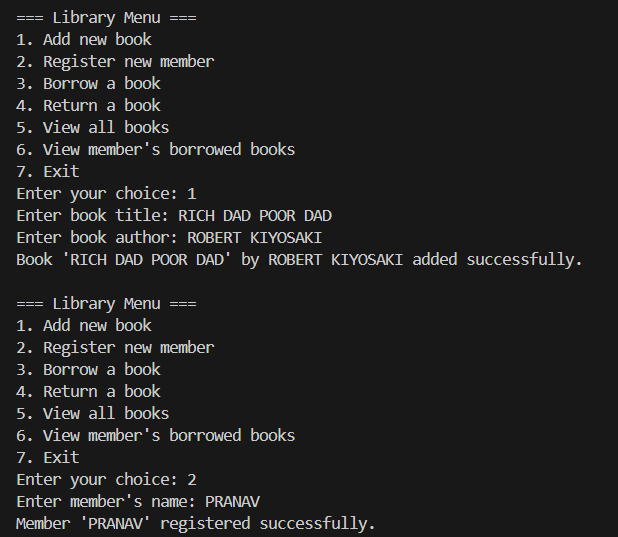
if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT:

A screenshot of a computer program

AI-generated content may be incorrect.



A paper with writing on it

AI-generated content may be incorrect.

A piece of paper with writing on it

AI-generated content may be incorrect.

