Practice Exercises



**Introduction to Object – Oriented Programming (OOP) in Python**

**INDIVIDUAL EXERCISES ( with instructor)**

*Exercise 1: Build a Simple Library System*

* We start by defining three classes: Book, Member, and Library.
* The **Book** class represents individual books with attributes such as title, author, and availability status.
* The **Member** class models library members with attributes like name, member ID, and a list of borrowed books.
* The **Library** class manages the library's collection of books and members.
* Implement Class Methods:
* In the **Book** class, we have an **\_\_init\_\_** method to initialize book attributes, and a **\_\_str\_\_** method to provide a string representation of the book object.
* Similarly, the Member class has an **\_\_init\_\_** method to initialize member attributes, and a **\_\_str\_\_** method for string representation.
* The Library class has methods to add books to the collection (add\_book), register members (register\_member), lend books to members (lend\_book), and handle book returns (return\_book).
* Test the Library System:
* We create instances of the **Book, Member,** and **Library** classes.
* Books and members are added to the library using the add\_book and register\_member methods.
* We test the functionality of lending books by calling the lend\_book method and returning books using the return\_book method.
* The output of each operation is printed to the console to provide feedback on the actions taken.
* Now, let's execute each step of the code:

*class Book:*

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

*self.title = title*

*self.author = author*

*self.available = True*

*def \_\_str\_\_(self):*

*return f"{self.title} by {self.author}"*

*class Member:*

*def \_\_init\_\_(self, name, member\_id):*

*self.name = name*

*self.member\_id = member\_id*

*self.borrowed\_books = []*

*def \_\_str\_\_(self):*

*return f"Member: {self.name} (ID: {self.member\_id})"*

*class Library:*

*def \_\_init\_\_(self):*

*self.books = []*

*self.members = []*

*def add\_book(self, book):*

*self.books.append(book)*

*def register\_member(self, member):*

*self.members.append(member)*

*def lend\_book(self, book, member):*

*if book.available:*

*book.available = False*

*member.borrowed\_books.append(book)*

*print(f"{book.title} has been lent to {member.name}")*

*else:*

*print(f"Sorry, {book.title} is not available for lending")*

*def return\_book(self, book, member):*

*if book in member.borrowed\_books:*

*book.available = True*

*member.borrowed\_books.remove(book)*

*print(f"{book.title} has been returned by {member.name}")*

*else:*

*print(f"{member.name} did not borrow {book.title}")*

*# Example usage:*

*# Create books*

*book1 = Book("Harry Potter and the Sorcerer's Stone", "J.K. Rowling")*

*book2 = Book("To Kill a Mockingbird", "Harper Lee")*

*# Create members*

*member1 = Member("John", 101)*

*member2 = Member("Alice", 102)*

*# Create library*

*library = Library()*

*# Add books to the library*

*library.add\_book(book1)*

*library.add\_book(book2)*

*# Register members*

*library.register\_member(member1)*

*library.register\_member(member2)*

*# Test lending and returning books*

*library.lend\_book(book1, member1)*

*library.lend\_book(book2, member2)*

*library.return\_book(book1, member1)*

*library.return\_book(book2, member1) # This should fail as book2 is not borrowed by member1*

