Simple library system:-

→ One main User

→ add Book
→ Print library books

Class Book:

Def __init__(Self, name, id, total_quantity)

Self.name = name
Self.id = id
Self.total = total
Self.total_quantity = total_quantity
Self.total_Borrowed = 0
def Borrow(self) → Bool: → To See if We can Borrow or Not
if Self.total_quantity - Self.total_Borrowed = 0:
Self.Borrowed +1 Return false ← if دا مفيش كتب
Return True ← لو في

def Return_Copy(Self):

assert Self.total_Borrowed >0 error لو اصغر من صفر هيرمي
Self.total_Borrowed -= 1 → ننقص عدد ال Borrowed

def __repr__(self): → To Print the objects of class Book

return f'Book Name: {Self.name} _id: {Self.id} - total quantity: {self.total_quantity} - ' / f' total Borrowed {Self.total Borrowed}

Class User:

```
def __init__ (Self, name, id):
    Self.name = name
    Self.id = id
    Self.Borrowed_Books = []

def Borrow_Book (Self, Book):
    Self.Borrowed_Books.append(Book) → add Book to Borrowed_Books

def Is_Borrowed (Self, Book):
    For Book in Self.Borrowed_Books:
        if book.id == Book.id:
            Return True
    Return False
```

→ Check if The Book is Borrowed or Not By Checking id

```
def Simple_repr (Self, Isdetailed = false)
    {
```

```
Class Backend_manager:

    def __init__(self):
        books = []
        users = []

def add_Book(self, name, id, total_quantity):
    self.books.append(Book(name, id, total_quantity))
                    ↳ Ref to Book Class

def get_Book_with_Prefix(self, Prefix):

    Return [book for book in Books if Prefix in Book]
                    ↳ Return list if Prefix is part of
                        a Book Name.

def add_User(self, name, id): ⟶ added struct User to
    self.users.append(User(name, id)) list Users

def Get_user_by_name(Sef, Username):
    For user in self.users:
        If username == user.name:
            Return user
    Return False

def Get_book_by_name(self, book_name):
        For book in self.Books:
            If book_name == book.name:
                Return Book.
        Return False
```

```
def Borrow_Book(self, username, Booknamed):

    book = self.get_book_by_Name(Bookname)
    User = Self.get_User_by.name(Username)

    if Book is None and User is None:
        Return.
    if Book.Borrow():  → Can we Borrow ?
        User.Borrow_Book(book)
                    ↳ add it to User
```

```
def Return_Book(Self, User_name, Book_name):
    book = Self.get_book_by_name(Book_name)
    User = Self.get_User_by_User(User_name)

    if book is None and User is None:
        Return                                  Check if
    if User.is Borrowed(book):                  Book Borrowed
    book.Return_Copy()                          or Not then
        User.return_copy(book)                  Return it.
    else:
      Return f'Book isn't Borrowed'
```

```
def get_Users_borrowed_Books(self, book_name):

    book = Self.get_book_by_name(Book_name)

    Return [User.name for User in self.Users
            if book
```