PYTHON OBJECT-ORIENTED PROGRAMMING (OOP) – COMPLETE CONCEPTS

Introduction to Object-Oriented Programming (OOP)

Object-OrientedProgramming organizescodeinto objectscontainingattributesand methods. OOP enables modularity, reusability, and maintainable code. Python is a fully object-oriented language.

Classes and Objects – Blueprints and Instances

```
class Car:

    pass

my_car = Car()
print(my_car)
```

Attributes and Methods in Classes

```
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, I am {self.name} and I am {self.age} years old.")

p1 = Person("Ava", 20)
p1.greet()
```

Constructors – The __init__ Method

```
class Student:

    def __init__(self, name, course):
        self.name = name
        self.course = course

s = Student("Liam", "Python")
print(s.name, s.course)
```

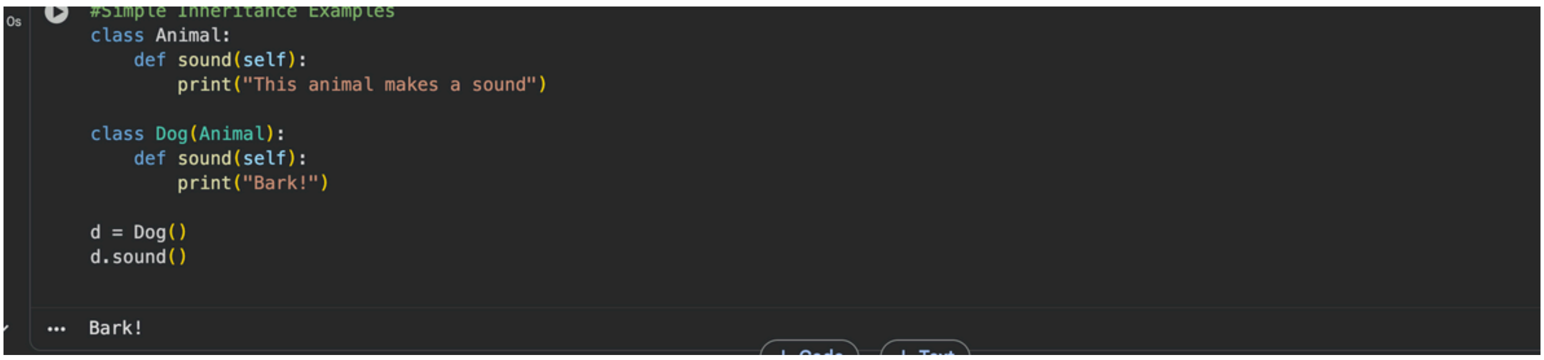
Class Variables vs Instance Variables

```
class Product:

    category = "Electronics"   # class variable

    def __init__(self, name, price):
        self.name = name       # instance variable
        self.price = price

p1 = Product("Phone", 50000)
```

```
p2 = Product("Laptop", 90000)

print(p1.category)
print(p2.category)
print(p1.name)
print(p2.name)
```

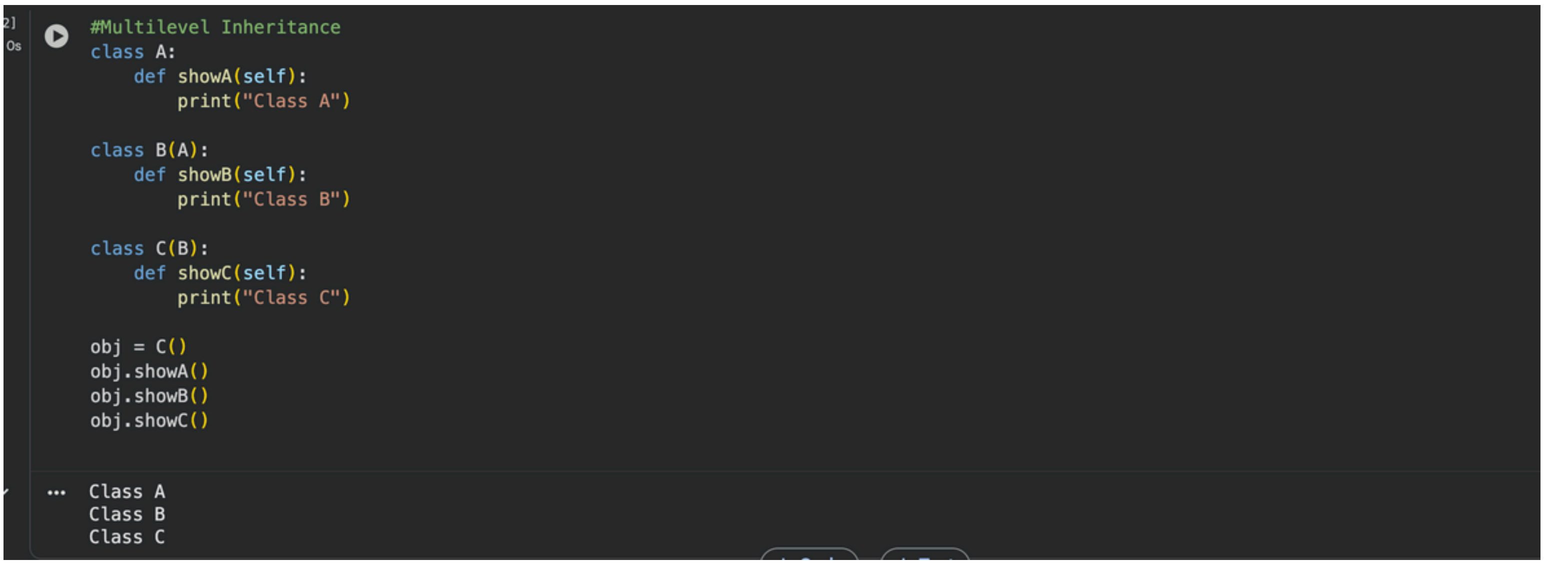
Basic Inheritance Concepts

```
class Animal:
    def speak(self):
        print("This animal makes a sound")

class Dog(Animal):
    def speak(self):
        print("Dog barks")

d = Dog()
d.speak()
```



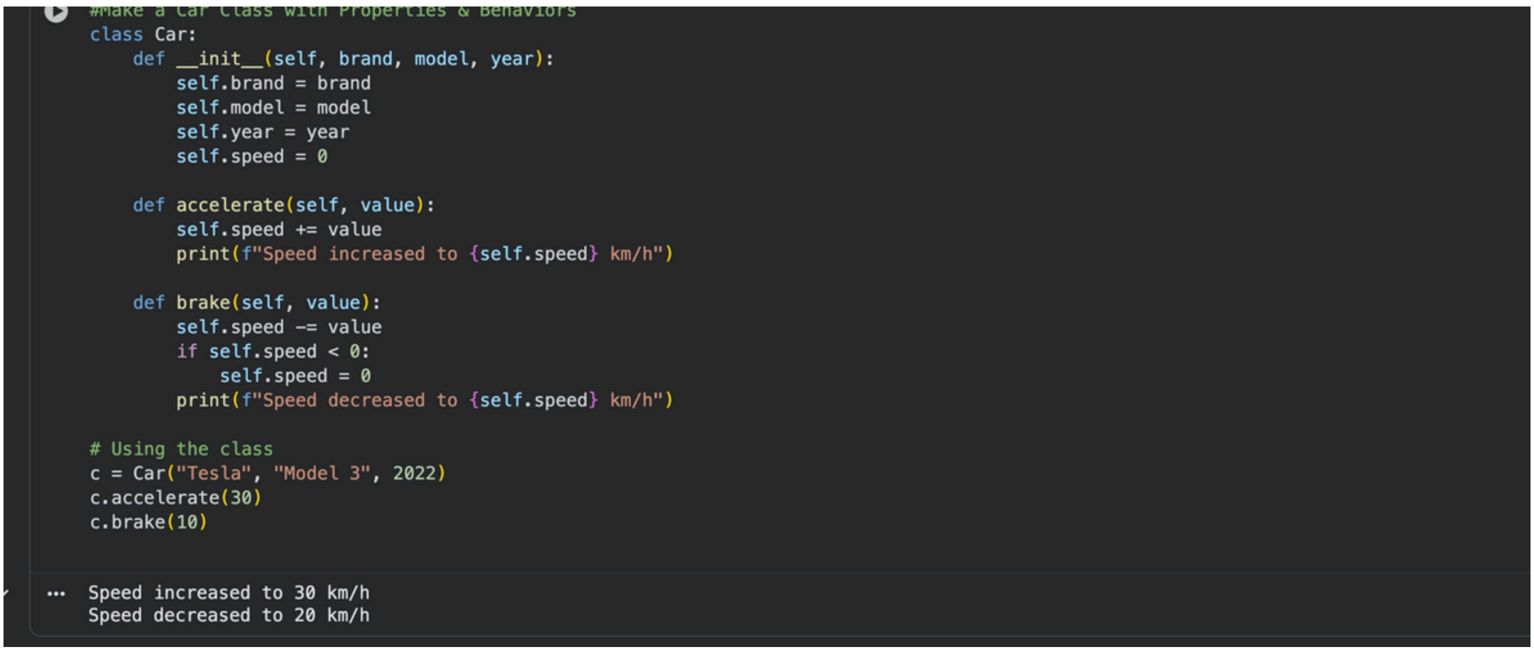
```python
#Simple Inheritance Examples
class Animal:
    def sound(self):
        print("This animal makes a sound")

class Dog(Animal):
    def sound(self):
        print("Bark!")

d = Dog()
d.sound()
```

```
Bark!
```



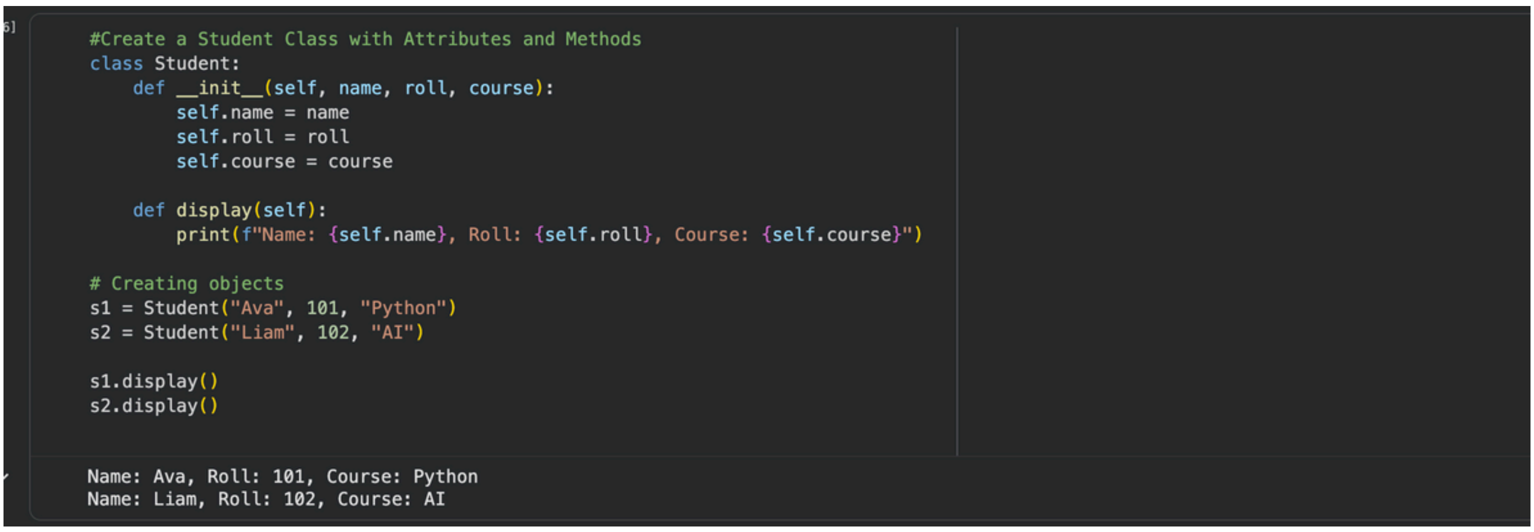
```python
#Multilevel Inheritance
class A:
    def showA(self):
        print("Class A")

class B(A):
    def showB(self):
        print("Class B")

class C(B):
    def showC(self):
        print("Class C")

obj = C()
obj.showA()
obj.showB()
obj.showC()
```

```
Class A
Class B
Class C
```



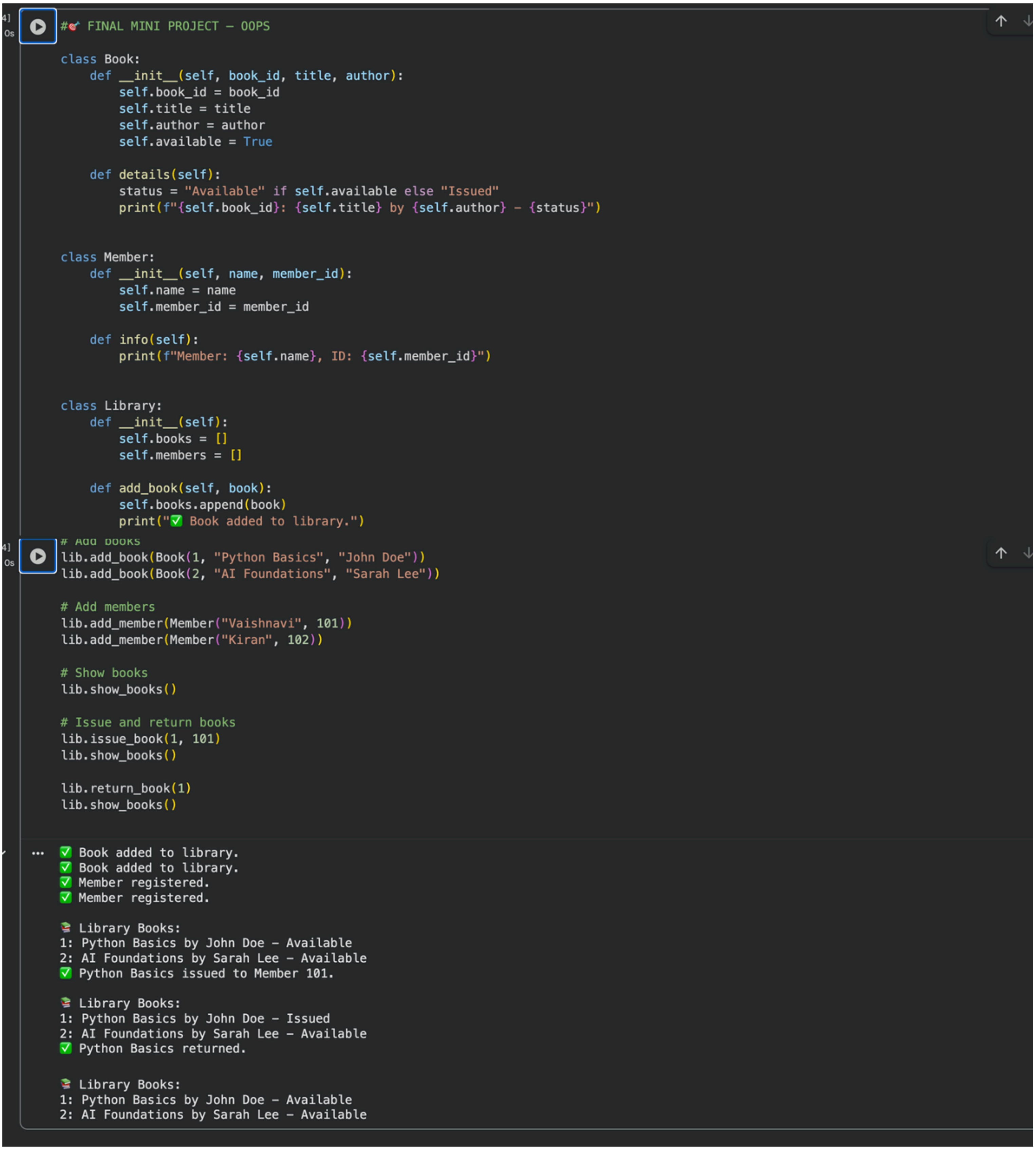
```python
#Make a Car Class with Properties & Behaviors
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year
        self.speed = 0

    def accelerate(self, value):
        self.speed += value
        print(f"Speed increased to {self.speed} km/h")

    def brake(self, value):
        self.speed -= value
        if self.speed < 0:
            self.speed = 0
        print(f"Speed decreased to {self.speed} km/h")

# Using the class
c = Car("Tesla", "Model 3", 2022)
c.accelerate(30)
c.brake(10)
```

```
Speed increased to 30 km/h
Speed decreased to 20 km/h
```



```python
#Create a Student Class with Attributes and Methods
class Student:
    def __init__(self, name, roll, course):
        self.name = name
        self.roll = roll
        self.course = course

    def display(self):
        print(f"Name: {self.name}, Roll: {self.roll}, Course: {self.course}")

# Creating objects
s1 = Student("Ava", 101, "Python")
s2 = Student("Liam", 102, "AI")

s1.display()
s2.display()
```

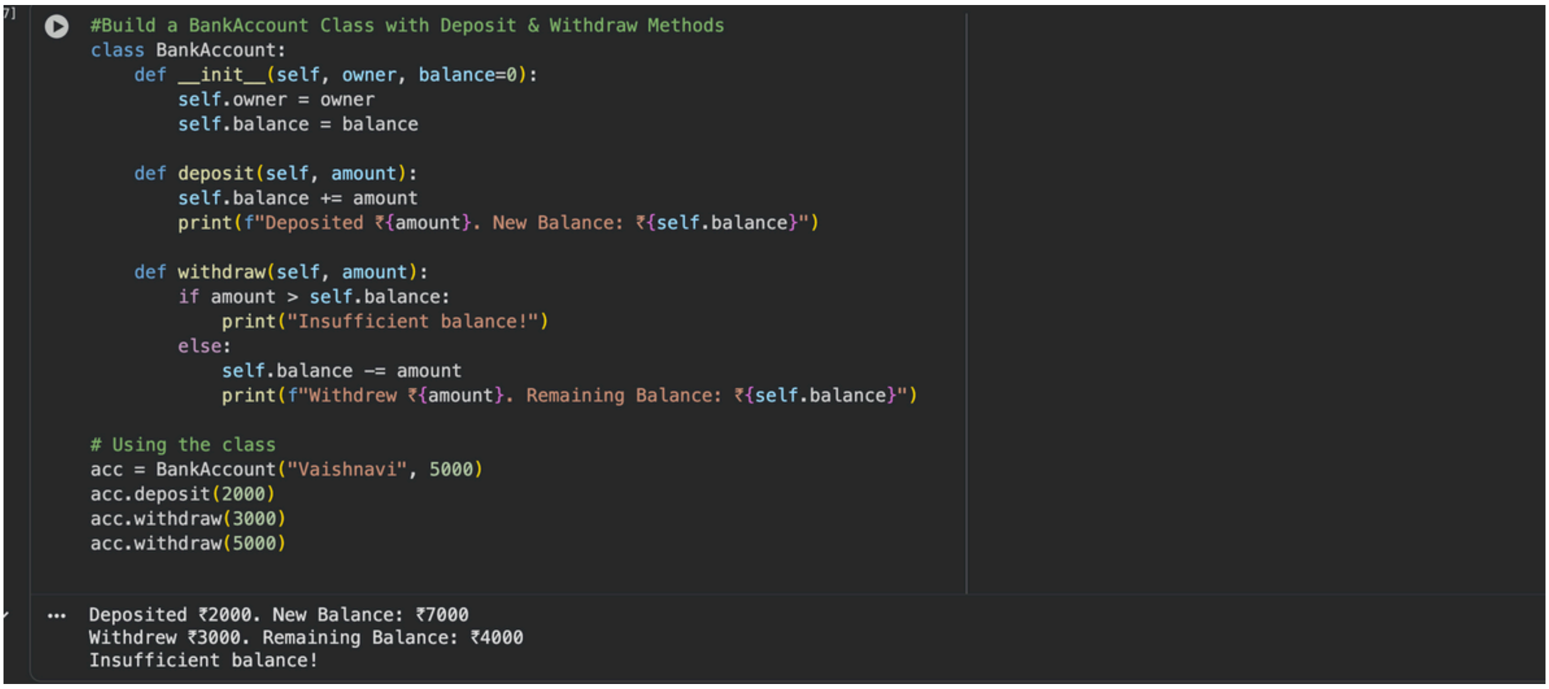
```
Name: Ava, Roll: 101, Course: Python
Name: Liam, Roll: 102, Course: AI
```



```python
#🎯 FINAL MINI PROJECT – OOPS

class Book:
    def __init__(self, book_id, title, author):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.available = True

    def details(self):
        status = "Available" if self.available else "Issued"
        print(f"{self.book_id}: {self.title} by {self.author} – {status}")


class Member:
    def __init__(self, name, member_id):
        self.name = name
        self.member_id = member_id

    def info(self):
        print(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)
        print("✅ Book added to library.")
```

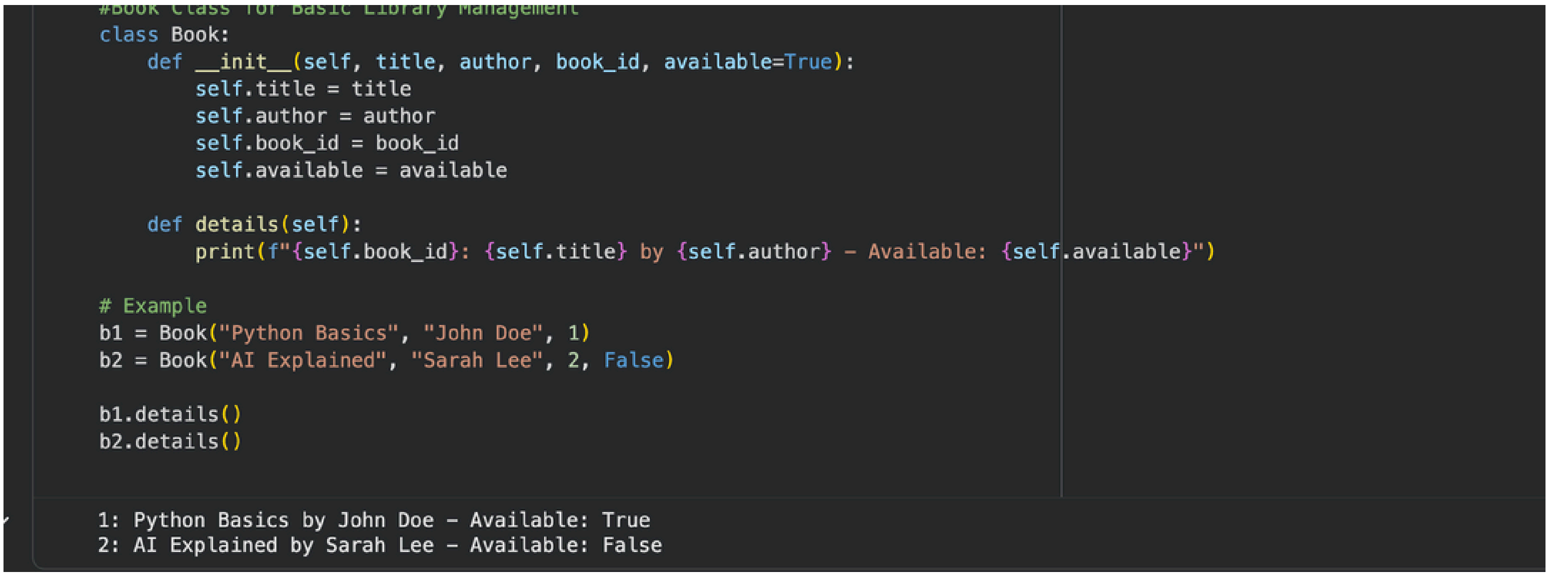
```python
# Add books
lib.add_book(Book(1, "Python Basics", "John Doe"))
lib.add_book(Book(2, "AI Foundations", "Sarah Lee"))

# Add members
lib.add_member(Member("Vaishnavi", 101))
lib.add_member(Member("Kiran", 102))

# Show books
lib.show_books()

# Issue and return books
lib.issue_book(1, 101)
lib.show_books()

lib.return_book(1)
lib.show_books()
```

```
✅ Book added to library.
✅ Book added to library.
✅ Member registered.
✅ Member registered.

📚 Library Books:
1: Python Basics by John Doe – Available
2: AI Foundations by Sarah Lee – Available
✅ Python Basics issued to Member 101.

📚 Library Books:
1: Python Basics by John Doe – Issued
2: AI Foundations by Sarah Lee – Available
✅ Python Basics returned.

📚 Library Books:
1: Python Basics by John Doe – Available
2: AI Foundations by Sarah Lee – Available
```



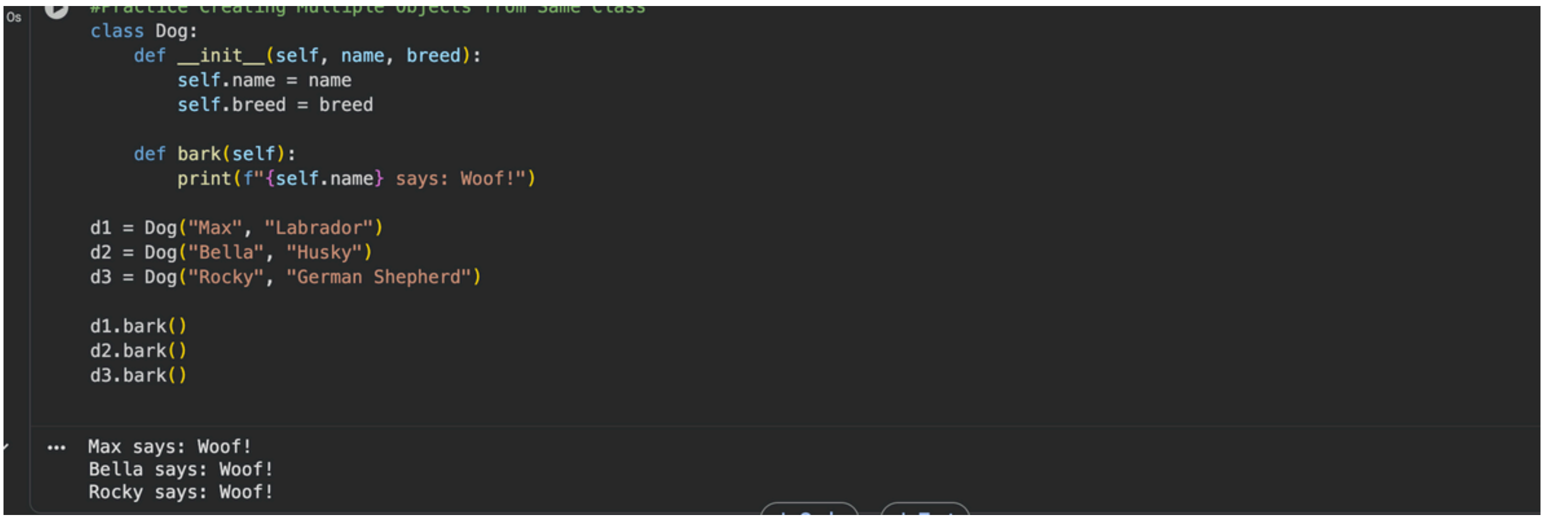
```python
#Build a BankAccount Class with Deposit & Withdraw Methods
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited ₹{amount}. New Balance: ₹{self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"Withdrew ₹{amount}. Remaining Balance: ₹{self.balance}")

# Using the class
acc = BankAccount("Vaishnavi", 5000)
acc.deposit(2000)
acc.withdraw(3000)
acc.withdraw(5000)
```

```
Deposited ₹2000. New Balance: ₹7000
Withdrew ₹3000. Remaining Balance: ₹4000
Insufficient balance!
```



```python
#Book Class for Basic Library Management
class Book:
    def __init__(self, title, author, book_id, available=True):
        self.title = title
        self.author = author
        self.book_id = book_id
        self.available = available

    def details(self):
        print(f"{self.book_id}: {self.title} by {self.author} - Available: {self.available}")

# Example
b1 = Book("Python Basics", "John Doe", 1)
b2 = Book("AI Explained", "Sarah Lee", 2, False)

b1.details()
b2.details()
```

```
1: Python Basics by John Doe - Available: True
2: AI Explained by Sarah Lee - Available: False
```



```python
#Practice Creating Multiple Objects from Same Class
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        print(f"{self.name} says: Woof!")

d1 = Dog("Max", "Labrador")
d2 = Dog("Bella", "Husky")
d3 = Dog("Rocky", "German Shepherd")

d1.bark()
d2.bark()
d3.bark()
```

```
Max says: Woof!
Bella says: Woof!
Rocky says: Woof!
```