**NAME VIKRAM KUMAR**

**ID BIT-23F-026**

**SUBJECT ARTIFICAL INTELEGENCE**

**LAB #8**

**SECTION 3A**

**DATE 11-12-2024**

**LAB TASK 1**

Task1. Write a Python class named Car that represents a car. The class should have the following attributes:

make: the car's make (e.g., "Toyota")

model: the car's model (e.g., "Corolla")

year: the car's manufacturing year (e.g., 2020)

mileage: the number of miles driven by the car.

The class should have the following methods:

\_init\_(self): Constructor to initialize the car's attributes.

display\_info(): Displays the car's information (make, model, year, mileage).

drive(miles): Increases the mileage by the specified number of miles

**class** Car:

**def** \_\_init\_\_(self, make, model, year, mileage=0):

        self.make = make

        self.model = model

        self.year = year

        self.mileage = mileage

**def** display\_info(self):

        print(**f**"Car Information:\nMake: {self.make}\nModel: {self.model}\nYear: {self.year}\nMileage: {self.mileage} miles")

**def** drive(self, miles):

        self.mileage += miles

        print(**f**"The car has been driven for {miles} miles. New mileage: {self.mileage} miles")

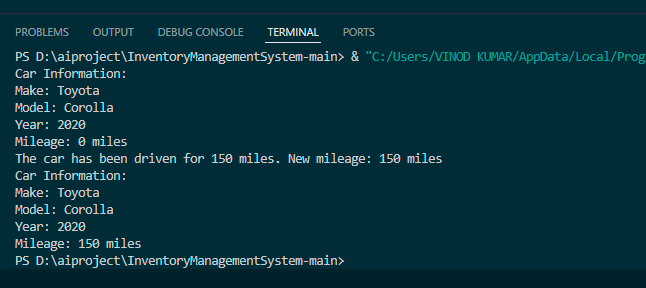
my\_car = Car("Toyota", "Corolla", 2020)

my\_car.display\_info()

my\_car.drive(150)

my\_car.display\_info()

**OUTPUT**



**LAB TASK 2**

Write a Python class named Student that represents a student. The class should have the following attributes:

name: the student's name.

age: the student's age.

marks: a list of the student's marks.

The class should have the following methods:

\_init\_(self): Constructor to initialize the student's attributes.

add\_marks(self, marks): Adds a list of marks to the student's marks list.

average\_marks(self): Calculates and returns the average of the student's marks.

display\_info(self): Displays the student's information (name, age, average marks).

**class** Student:

**def** \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

        self.marks = []

**def** add\_marks(self, marks):

        self.marks.extend(marks)

**def** average\_marks(self):

        if len(self.marks) == 0:

            return 0

        return sum(self.marks) / len(self.marks)

**def** display\_info(self):

        avg\_marks = self.average\_marks()

        print(**f**"Student Information:\nName: {self.name}\nAge: {self.age}\nAverage Marks: {avg\_marks**:.2f**}")

student1 = Student("VIKRAM", 20)

student1.add\_marks([85, 90, 78, 92])

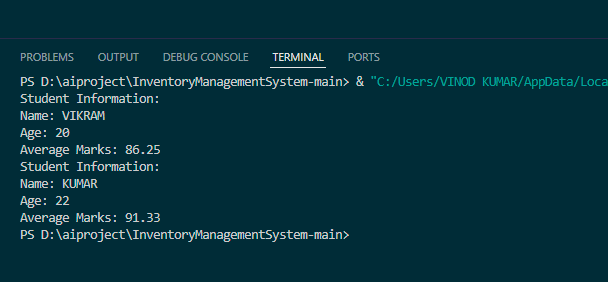
student1.display\_info()

student2 = Student("KUMAR", 22)

student2.add\_marks([88, 95, 91])

student2.display\_info()

**OUTPUT**



**LAB TASK 3**

Write a Python class named BankAccount that represents a bank account. The class should have the following attributes:

account\_holder: the name of the account holder.

balance: the balance of the account.

The class should have the following methods:

\_init\_(self): Constructor to initialize the account holder's name and balance.

deposit(self, amount): Deposits an amount into the account.

withdraw(self, amount): Withdraws an amount from the account if there are sufficient funds.

display\_balance(self): Displays the current balance of the account.

**class** BankAccount:

**def** \_\_init\_\_(self, account\_holder, balance=0):

*# Constructor to initialize the account holder's name and balance*

        self.account\_holder = account\_holder

        self.balance = balance

**def** deposit(self, amount):

*# Deposits an amount into the account*

        if amount > 0:

            self.balance += amount

            print(**f**"Deposited {amount}. New balance: {self.balance}")

        else:

            print("Deposit amount must be greater than zero.")

**def** withdraw(self, amount):

*# Withdraws an amount from the account if there are sufficient funds*

        if amount <= self.balance:

            self.balance -= amount

            print(**f**"Withdrew {amount}. New balance: {self.balance}")

        else:

            print("Insufficient funds for withdrawal.")

**def** display\_balance(self):

*# Displays the current balance of the account*

        print(**f**"Account holder: {self.account\_holder}\nCurrent balance: {self.balance}")

*# Example usage:*

account1 = BankAccount("VIKRAM", 500)

account1.display\_balance()

account1.deposit(200)

account1.withdraw(100)

account1.display\_balance()

account2 = BankAccount("KUMAR")

account2.deposit(1000)

account2.withdraw(1200)

account2.display\_balance()

**OUTPUT**

