

**Lab # 07**

**Student Name: Piyar Ali Soomro**

**Roll Number: BIT-23F-019**

**Section: A**

**Subject: Artificial Intelligence**

**Department: BS Information Technology**

**New assignment**

**LAB: 07**

**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

**Code:**

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"\n Car Information:")

        print(f" Make:    {self.make}")

        print(f" Model:   {self.model}")

        print(f" Year:    {self.year}")

        print(f" Mileage: {self.mileage} miles")

    def drive(self, miles):

        if miles > 0:

            self.mileage += miles

            print(f" The car has been driven {miles} miles. Total mileage is now {self.mileage} miles.")

        else:

            print("Miles to drive must be a positive number.")

# Create a new Car object

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

# Display car information

my\_car.display\_info()

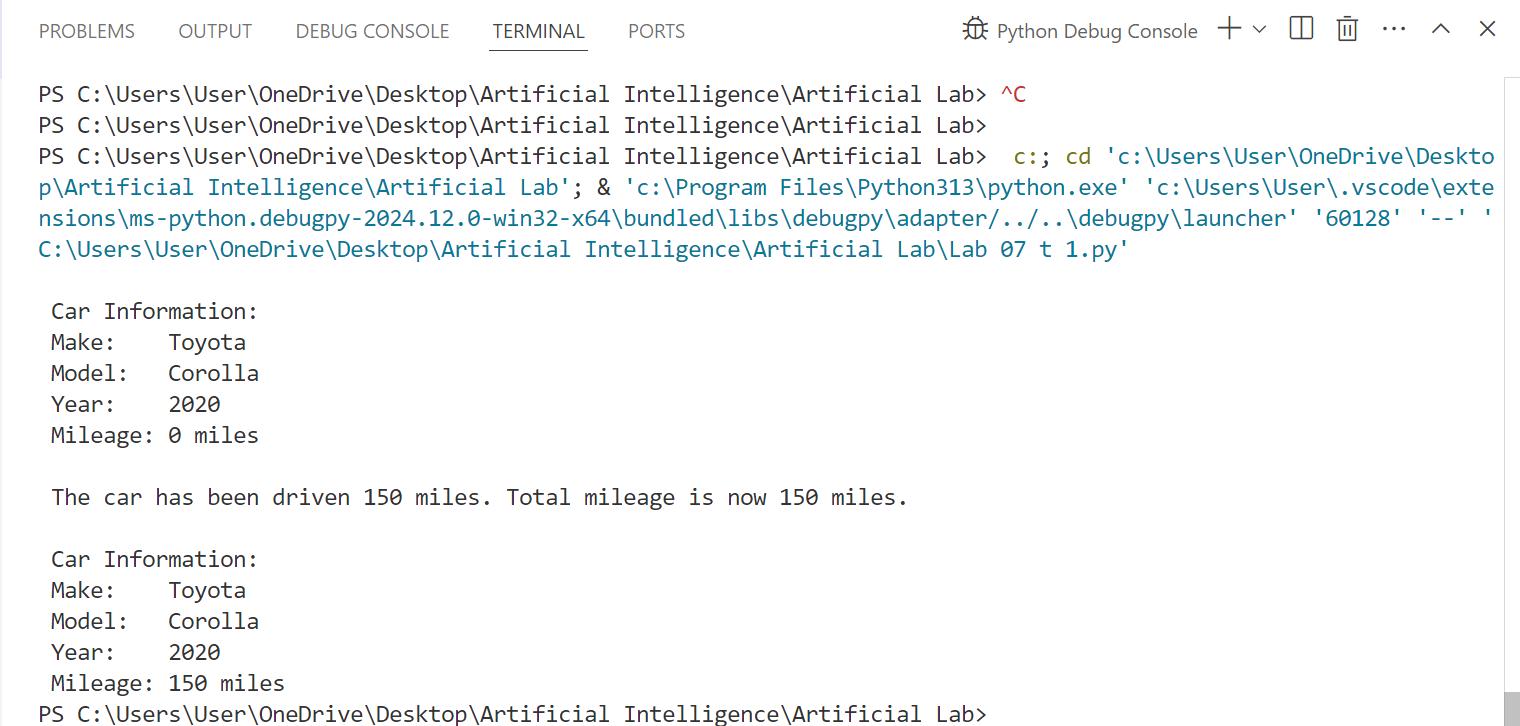
# Drive the car

my\_car.drive(150)

# Display updated car information

my\_car.display\_info()

**Output:**



**Task2**. 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).

**Code:**

class Student:

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

        self.name = name

        self.age = age

        self.marks = []

    def add\_marks(self, marks):

        if all(isinstance(mark, (int,float)) and mark >= 0 for mark in marks):

            self.marks.extend(marks)

            print(f"\n => Marks {marks} added successfully! ")

        else:

            print("All marks must be non-negative numbers.")

    def average\_marks(self):

        if self.marks:

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

        else:

            return None

    def display\_info(self):

        avg\_marks = self.average\_marks()

        print(f"\n Student Information:")

        print(f" Name: {self.name}")

        print(f" Age:  {self.age}")

        if avg\_marks is not None:

            print(f" Average Marks: {avg\_marks:.2f}")

        else:

            print(" Average Marks: No marks available.")

# Create a Student object

student = Student(name="Piyar Ali Soomro", age=22)

# Display initial student information

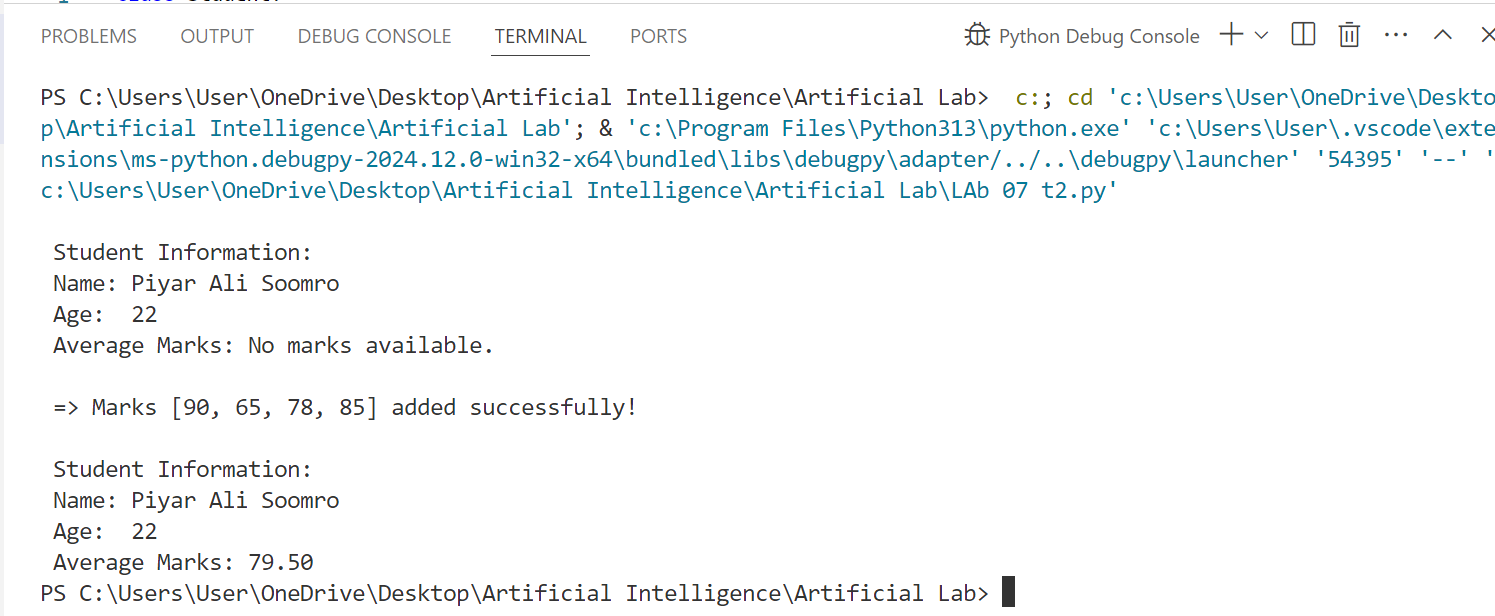
student.display\_info()

# Add marks to the student

student.add\_marks([90, 65, 78, 85 ])

# Display updated student information

student.display\_info()

**Output:**

**Task3.** 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.

**Code:**

class BankAccount:

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

        self.account\_holder = account\_holder

        self.balance = balance

    def deposit(self, amount):

        if amount > 0:

            self.balance += amount

            print(f" Rs = {amount:.2f} deposited successfully. New balance: Rs = {self.balance:.2f}")

        else:

            print("Deposit amount must be positive.")

    def withdraw(self, amount):

        if amount > 0:

            if amount <= self.balance:

                self.balance -= amount

                print(f" Rs = {amount:.2f} withdrawn successfully. New balance: Rs = {self.balance:.2f}")

            else:

                print(f" Insufficient funds. Your current balance is Rs = {self.balance:.2f}.")

        else:

            print("Withdrawal amount must be positive.")

    def display\_balance(self):

        print(f"\n Account Holder: {self.account\_holder}")

        print(f" Current Balance: Rs = {self.balance:.2f}")

# Create a bank account

account = BankAccount(account\_holder=" Piyar Ali Soomro", balance=1000)

# Display initial balance

account.display\_balance()

# Deposit money

account.deposit(786)

# Withdraw money

account.withdraw(987)

# Attempt to withdraw more than the balance

account.withdraw(1200)

# Display final balance

account.display\_balance()

**Output:**

