***Komal Tariq***

***BIT-23F-021***

***SECTION: “B”***

***ARTIFICIAL INTELLIGENCE***

***LAB 8***

***ASSIGNMENT***

***MISS: AQSA***

***Task 01:***

Q:  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"Car Info:\nMake: {self.make}\nModel: {self.model}\nYear: {self.year}\nMileage: {self.mileage} miles")

def drive(self, miles):

self.mileage += miles

# Create a Car object with predefined details

car = Car("Toyota", "Corolla", 2020, 15000)

# Display initial car information

car.display\_info()

# Ask the user for miles to drive

miles\_to\_drive = int(input("Enter miles to drive: "))

# Drive the car and update mileage

car.drive(miles\_to\_drive)

# Display the updated car information

car.display\_info()

**OUTPUT:**

Car Info:

Make: Toyota

Model: Corolla

Year: 2020

Mileage: 15000 miles

Enter miles to drive: 160

Car Info:

Make: Toyota

Model: Corolla

Year: 2020

Mileage: 15160 miles

***Task 02:***

Q: 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):

self.marks.extend(marks)

def average\_marks(self):

if self.marks:

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

return 0

def display\_info(self):

print(f"Name: {self.name}, Age: {self.age}, Average Marks: {self.average\_marks():.2f}")

# Example usage

name = input("Enter student's name: ")

age = int(input("Enter student's age: "))

student = Student(name, age)

marks = list(map(int, input("Enter the student's marks (separated by spaces): ").split()))

student.add\_marks(marks)

student.display\_info()

**OUTPUT:**

Enter student's name: younus

Enter student's age: 18

Enter the student's marks (separated by spaces): 30 30 40 60 60 70 80

Name: ALI, Age: 20, Average Marks: 52.86

***Task 03:***

Q: 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"Deposited ${amount}. New balance: ${self.balance}")

else:

print("Deposit amount must be positive.")

def withdraw(self, amount):

if amount > 0 and amount <= self.balance:

self.balance -= amount

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

else:

print("Invalid withdrawal amount or insufficient funds.")

def display\_info(self):

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

print(f"Current Balance: ${self.balance}")

# Example usage

account = BankAccount("John Doe", 1000)

# Display initial account info

account.display\_info()

# Ask user for deposit

deposit\_amount = float(input("Enter amount to deposit: "))

account.deposit(deposit\_amount)

# Ask user for withdrawal

withdraw\_amount = float(input("Enter amount to withdraw: "))

account.withdraw(withdraw\_amount)

# Display updated account info

account.display\_info()

**OUTPUT:**

Account Holder: John Doe

Current Balance: $1000

Enter amount to deposit: 1500

Deposited $1500.0. New balance: $2500.0

Enter amount to withdraw: 500

Withdrew $500.0. New balance: $2000.0

Account Holder: John Doe

Current Balance: $2000.0