1. Define a simple Car class with attributes for make, model, and year. Create an object of this class and print its attributes.

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
A1 = Car("JANUARY", 2024, 2023 )
print(A1.make)
print(A1.model)
print(A1.year)
```

```
JANUARY
2024
2023
```

2. Create a Person class with attributes for name and age. Add a method to print a greeting message. Create an object and call the method.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

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

# Create an object of the Person class
person = Person("Navdeep singh", 18)

# Call the greet method
person.greet()
```

```
Hello, my name is Navdeep singh and I am 18 years old.
```

3. Define a Rectangle class with attributes for length and width. Add a method to calculate the area of the rectangle. Create an object and print the area.

```
class Rectangle:
     def __init__(self, Length, width):
        self.Length = Length
        self.width = width

     def area(self):
        return self.Length * self.width

rectangle = Rectangle(5, 6)
Area = rectangle.area()
print(f"the area of rectangle of =",Area )
```

```
the area of rectangle of = 30
```

4. Create a Student class with attributes for name and grades (a list of numbers). Add a method to calculate the average grade. Create an object and print the average grade.

```
class Student:
    def __init__(self, name, grades):
        self.name = name
        self.grades = grades

    def average_grade(self):
        total = sum(self.grades)
        average = total / len(self.grades)
        return average
student = Student("navi", [4, 5, 6, 8])
grade = student.average_grade()
print(f"{student.name} avearge grade is",grade)
```

```
navi avearge grade is 5.75
```

5. Define a Book class with attributes for title, author, and pages. Add a method to display the book's details. Create an object and call the method.

```
class Book:
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages

    def display(self):
        print(f"Book title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Pages: {self.pages}")
b1 = Book("Revenge of India","Navdeep singh", 150 )
print(b1.title)
print(b1.author)
print(b1.pages)
```

```
Revenge of India
Navdeep singh
150
```

6. Create a Dog class with attributes for name and breed. Add a method to make the dog bark (print a bark message). Create an object and call the method.

```
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        print(f"{self.name} is barking.")


dog = Dog("jacky", "German Shepherd")
dog.bark()
```

```
jacky is barking.
```

7. Define a BankAccount class with attributes for account number and balance. Add methods to deposit and withdraw money. Create an object and test the methods.

```
class BankAccount:
    def __init__(self, account_number, balance):
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount}. New balance is ${self.balance}.")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew ${amount}. New balance is ${self.balance}.")
        elif amount > self.balance:
            print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")


account = BankAccount("123456789", 1000)
account.deposit(500)
account.withdraw(200)
account.withdraw(2000)
account.withdraw(-100)
account.deposit(-50)
```

```
Deposited $500. New balance is $1500.
Withdrew $200. New balance is $1300.
Insufficient funds.
Withdrawal amount must be positive.
Deposit amount must be positive.
```

8. Create a Circle class with attributes for radius. Add a method to calculate the circumference. Create an object and print the circumference.

```
class Circle:
     def __init__(self,radius):
        self.radius = radius
     def calculate_circumference(self):
        return 2 * 3.14 * self.radius
c1= Circle(45)
circumference = c1.calculate_circumference()
print(f"the circumference of circle is =",circumference)
```

```
the circumference of circle is = 282.6
```

9. Define a Laptop class with attributes for brand and price. Add a method to apply a discount to the price. Create an object and test the method.

```
class  Laptop:
    def __init__(self,brand,price):
        self.brand = brand
        self.price = price

    def apply_discount(self, discount_percentage):
        discount_amount = self.price * discount_percentage / 100
        self.price -= discount_amount
        return self.price
laptop = Laptop("hp",45000)
print(f"Original price: ${laptop.price}")
laptop.apply_discount(20)
print(f"Price after 20% discount: ${laptop.price}")
```

```
Original price: $45000
Price after 20% discount: $36000.0
```

10. Create a Employee class with attributes for name and salary. Add a method to give a raise (increase the salary). Create an object and test the method.

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def give_raise(self, raise_amount):
        self.salary += raise_amount
        return self.salary
employee = Employee("Navdeep singh", 50000)
print(f"Original salary: ${employee.salary}")
employee.give_raise(10000)
```

```
Original salary: $50000
60000
```

Double-click (or enter) to edit

11. Define a Point class with attributes for x and y coordinates. Add a method to calculate the distance from another point. Create two objects and print the distance between them.

```
import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance_from(self, other_point):
        distance = math.sqrt((self.x - other_point.x) ** 2 + (self.y - other_point.y) ** 2)
        return distance
point1 = Point(1, 2)
point2 = Point(4, 6)
distance = point1.distance_from(point2)
print(f"Distance between point1 and point2: {distance}")
```

```
Distance between point1 and point2: 5.0
```