

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.r=make
        self.model=model
        self.year=year

obj1=Car("swift","zdi+",2020)
print(obj1.r)
print(obj1.model)
print(obj1.year)
```

```
→ swift
   zdi+
   2020
```

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 greeting_function(self):
        print("Hey, you are welcome!")

# Create an object of the Person class
obj = Person("Nitin", 85)

# Call the greeting function
obj.greeting_function()

# Print the name attribute of the object
print(obj.name)
```

```
→ Hey, you are welcome!
   Nitin
```

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.lenght=length
        self.width=width
    def printing(self):
        print("Area=",self.lenght*self.width)

obj=Rectangle(6,8)
obj.printing()
```

```
→ Area= 48
```

. 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,grade):
        self.name=name
        self.grade=grade
    def average(self):

        print(sum(self.grade)/len(self.grade))

obj=student("nitin",grade=[5,4])
obj.average()
```

```
→ 4.5
```

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 book_details(self):
        print("title name=",self.title)
        print("author name=",self.author)
        print("pages name=",self.pages)
obj=book("freedom","raja singh",298)
obj.book_details()
```

```
→ title name= freedom
author name= raja singh
pages name= 298
```

Double-click (or enter) to edit

. 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,acc_no,bal):
        self.acc_no=acc_no
        self.bal=bal
    def deposit(self,amount):
        self.bal+=amount
    def withdraw(self,amount):
        if self.bal<amount:
            print("no sufficient amount")
        else:
            self.bal-=amount
    def balance(self):
        print("total balace=",self.bal)
obj=bankaccount(9,50)
obj.deposit(20)
obj.balance()
obj.withdraw(70)
obj.balance()
obj.withdraw(20)
```

```
→ total balace= 70
total balace= 0
no sufficient amount
```

Double-click (or enter) to edit

```
import math

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

    def distance(self, other_point):
        print( math.sqrt((point1.x - point2.x) ** 2 + (self.point1 - point2.y) ** 2))

point1 = Point(0, 0)
point2 = Point(3, 4)
```

```
class Calendar:
    def __init__(self, day, month, year):
        self.day = day
        self.month = month
        self.year = year

    def display_date(self):
        print(f"{self.day}/{self.month}/{self.year}")

calendar = Calendar(12, 6, 2024)
calendar.display_date()
```

```
→ 12/6/2024
```

Double-click (or enter) to edit

```

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.")

    def compare_age(self, other_person):
        if self.age > other_person.age:
            return f"{other_person.name} is younger than me."
        elif self.age < other_person.age:
            return f"{other_person.name} is older than me."
        else:
            return f"{other_person.name} is the same age as me."

# Creating objects
p1 = Person('Alice', 30)
p2 = Person('Bob', 25)
p3 = Person('Charlie', 30)

# Comparing ages
print(p1.compare_age(p2)) # Output: "Bob is younger than me."
print(p1.compare_age(p3)) # Output: "Charlie is the same age as me."
print(p2.compare_age(p3)) # Output: "Charlie is older than me."

```

Bob is younger than me.
 Charlie is the same age as me.
 Charlie is older than me.

Double-click (or enter) to edit

```

class OnesThreesNines:
    def __init__(self, num):
        self.num = num
        self.nines = self.num // 9
        self.threes = self.num // 3
        self.ones = self.num

# Example usage
n1 = OnesThreesNines(5)
print(n1.nines) # Output: 0
print(n1.ones)  # Output: 5
print(n1.threes) # Output: 1

n2 = OnesThreesNines(10)
print(n2.nines) # Output: 1
print(n2.ones)  # Output: 10
print(n2.threes) # Output: 3

```

0
 5
 1
 1
 10
 3

```

class Calculator:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            return "Error: Division by zero is not allowed."
        return a / b

# Example usage
calc = Calculator()
print(calc.add(10, 5))      # Output: 15
print(calc.subtract(10, 5)) # Output: 5
print(calc.multiply(10, 5)) # Output: 50
print(calc.divide(10, 5))   # Output: 2.0
print(calc.divide(10, 0))   # Output: Error: Division by zero is not allowed.

```

15
 5
 50

```
2.0
Error: Division by zero is not allowed.
```

```
class User:
    # Class attribute to keep track of the number of instances
    user_count = 0

    def __init__(self, name):
        self.name = name
        User.user_count += 1 # Increment the count whenever a new instance is created

# Example usage
user1 = User("Alice")
user2 = User("Bob")
user3 = User("Charlie")

# Access the user_count directly through the class name
print(User.user_count) # Output: 3
```

 3


```
class BasicPlan:
    can_stream = True
    can_download = True
    has_SD = True
    has_HD = False
    has_UHD = False
    num_of_devices = 1
    price = "$8.99"

    @classmethod
    def display_info(cls):
        print(f"{cls.__name__} Plan:")
        print(f"Can stream: {cls.can_stream}")
        print(f"Can download: {cls.can_download}")
        print(f"Has SD: {cls.has_SD}")
        print(f"Has HD: {cls.has_HD}")
        print(f"Has UHD: {cls.has_UHD}")
        print(f"Number of devices: {cls.num_of_devices}")
        print(f"Price: {cls.price}")
        print()

class StandardPlan(BasicPlan):
    has_HD = True
    num_of_devices = 2
    price = "$12.99"

class PremiumPlan(StandardPlan):
    has_UHD = True
    num_of_devices = 4
    price = "$15.99"

# Example usage
BasicPlan.display_info()
StandardPlan.display_info()
PremiumPlan.display_info()
```

 BasicPlan Plan:
Can stream: True
Can download: True
Has SD: True
Has HD: False
Has UHD: False
Number of devices: 1
Price: \$8.99

StandardPlan Plan:
Can stream: True
Can download: True
Has SD: True
Has HD: True
Has UHD: False
Number of devices: 2
Price: \$12.99

PremiumPlan Plan:
Can stream: True
Can download: True
Has SD: True
Has HD: True
Has UHD: True
Number of devices: 4
Price: \$15.99

