1. Class Practice questions

Q.1 Create a class Person with attributes name and age. Add a method show_details() that prints both.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def show_details(self):
        print(f"Name: {self.name}, Age: {self.age}")

p1 = Person("Neeraj", 21)
p1.show_details()

# output
Name: Neeraj, Age: 21
```

Q.2 Create a class Circle with attribute radius and methods:
area() → calculate circle area
circumference() → calculate circle circumference

```
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return math.pi * self.radius * self.radius
    def circumference(self):
        return 2 * math.pi * self.radius

c = Circle(5)
print("Area:", c.area())
print("Circumference:", c.circumference())

# Output
Area: 78.53981633974483
Circumference: 31.41592653589793
```

Q.3 Create a class **Student** with a class variable **school_name** and instance variables **name** and **roll**. Initialize them in constructor and create a method **show()** that displays details.

```python
class Student:
    school_name = "SHSS"   # class variable
    def __init__(self, name, roll):
        self.name = name
        self.roll = roll
    def show(self):
        print(f"{self.name} ({self.roll}) - {Student.school_name}")


# Test
s1 = Student("Neeraj", 101)
s2 = Student("Aman", 102)

s1.show()
s2.show()

# Output
Neeraj (101) - SHSS
Aman (102) - SHSS
```

Q4. Create a class BankAccount with methods:
**deposit(amount) **→ add money
**withdraw(amount) **→ subtract money if balance is sufficient
**check_balance() **→ print balance

```python
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited: {amount}, New Balance: {self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds!")
        else:
            self.balance -= amount
            print(f"Withdrawn: {amount}, Remaining Balance: {self.balance}")

    def check_balance(self):
        print(f"Current Balance: {self.balance}")
```

```
acc = BankAccount("Neeraj", 500)
acc.deposit(200)
acc.withdraw(100)
acc.check_balance()

# Output
Deposited: 200, New Balance: 700
Withdrawn: 100, Remaining Balance: 600
Current Balance: 600
```

Q.5 Create a class Employee with attributes **name and salary**. Add a method **get_tax()** that calculates **10%** of the salary as tax.

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def get_tax(self):
        tax = self.salary * 0.10
        return tax

e1 = Employee("Neeraj", 50000)
print(f"Employee: {e1.name}, Tax: {e1.get_tax()}")

# Output
Employee: Neeraj, Tax: 5000.0
```

Q.6 Inheritance Practice:
Class **Animal** → method **speak()**
Class **Dog** (inherits Animal) → override speak() to print "Woof!"
Class **Cat** (inherits Animal) → override speak() to print "Meow!"

```
class Animal:
    def speak(self):
        print("I am an animal")
class Dog(Animal):
    def speak(self):
        print("Woof!")
class Cat(Animal):
    def speak(self):
        print("Meow!")

a = Animal()
d = Dog()
c = Cat()
```

```
a.speak()
d.speak()
c.speak()

# Output
I am an animal
Woof!
Meow!
```

Q.7 Create a class ShoppingCart with methods:
add_item(item, price) → add items to cart
remove_item(item) → remove item from cart
get_total() → return total price of items
Use a dictionary to store items and prices.

```
class ShoppingCart:
    def __init__(self):
        self.items = {}

    def add_item(self, item, price):
        self.items[item] = price
        print(f"Added {item} - {price}")

    def remove_item(self, item):
        if item in self.items:
            del self.items[item]
            print(f"Removed {item}")
        else:
            print(f"{item} not found in cart")

    def get_total(self):
        return sum(self.items.values())

cart = ShoppingCart()
cart.add_item("Shoes", 1200)
cart.add_item("Shirt", 800)
cart.remove_item("Shirt")
print("Total:", cart.get_total())

# Output
Added Shoes - 1200
Added Shirt - 800
```

```
Removed Shirt
Total: 1200
```

Q.8 Create a class Library that maintains a list of books. Methods:
add_book(title)
remove_book(title)
display_books() → show all books

```python
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, title):
        self.books.append(title)
        print(f"Book '{title}' added.")

    def remove_book(self, title):
        if title in self.books:
            self.books.remove(title)
            print(f"Book '{title}' removed.")
        else:
            print(f"Book '{title}' not found.")

    def display_books(self):
        print("Books in library:", self.books)

lib = Library()
lib.add_book("Python Basics")
lib.add_book("Data Structures")
lib.display_books()
lib.remove_book("Python Basics")
lib.display_books()

# Output
Book 'Python Basics' added.
Book 'Data Structures' added.
Books in library: ['Python Basics', 'Data Structures']
Book 'Python Basics' removed.
Books in library: ['Data Structures']
```

Q.9 Create a class Math with:
Static method factorial(n)
Class method square(cls, x)
Test both methods.

```python
class Math:
    @staticmethod
    def factorial(n):
        result = 1
        for i in range(1, n+1):
            result *= i
        return result
    @classmethod
    def square(cls, x):
        return x * x

print("Factorial(5):", Math.factorial(5))
print("Square(4):", Math.square(4))

# Output
Factorial(5): 120
Square(4): 16
```

Q.10 Multiple Inheritance Practice:
Class Teacher → method teach()
Class Student → method study()
Class TeachingAssistant inherits from both Teacher and Student, and should be able to call both teach() and study().

```python
class Teacher:
    def teach(self):
        print("Teaching...")

class Student:
    def study(self):
        print("Studying...")

class TeachingAssistant(Teacher, Student):
    def assist(self):
        print("Assisting teacher and helping students.")


# Test
ta = TeachingAssistant()
ta.teach()
ta.study()
ta.assist()

# Output
```

```
Teaching...
Studying...
Assisting teacher and helping students.
```