Q1. Explain Class and Object with respect to Object-Oriented Programming. Give a suitable example.

Class

A **class** is a blueprint or template used to create objects. It defines the properties (data members) and behaviors (methods) that the objects created from it will have.

Object

An **object** is an instance of a class. It represents a real-world entity and can access the variables and methods defined in the class.

Example:

```
class Student:
    def display(self):
        print("This is a student class")

obj = Student()   # object creation
obj.display()
```

Q2. Name the four pillars of OOPs.

The four main pillars of Object-Oriented Programming are:

1. **Encapsulation**
2. **Inheritance**
3. **Polymorphism**
4. **Abstraction**

Q3. Explain why the `__init__()` function is used. Give a suitable example.

The `__init__()` function is a **constructor** in Python.
It is automatically called when an object of a class is created and is used to **initialize (assign values to) object variables**.

Example:

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def show(self):
        print(self.name, self.salary)

emp = Employee("Rahul", 50000)
emp.show()
```

Q4. Why is `self` used in OOPs?

`self` is a reference to the **current object** of the class.

Uses of `self`:

- To access instance variables
- To call instance methods within the class
- To differentiate between instance variables and local variables

Example:

```
class Car:
    def set_speed(self, speed):
        self.speed = speed
    def show_speed(self):
        print(self.speed)

c = Car()
c.set_speed(80)
c.show_speed()
```

Q5. What is inheritance? Give an example for each type of inheritance.

Inheritance

Inheritance allows a class (child class) to acquire the properties and methods of another class (parent class).
It promotes **code reusability**.

Types of Inheritance with Examples

1. Single Inheritance

```
class Parent:
    def show(self):
        print("Parent class")

class Child(Parent):
    pass

obj = Child()
obj.show()
```

---

2. Multiple Inheritance

```
class Father:
    def f(self):
        print("Father")
```

```python
class Mother:
    def m(self):
        print("Mother")

class Child(Father, Mother):
    pass

obj = Child()
obj.f()
obj.m()
```

---

3. Multilevel Inheritance

```python
class Grandparent:
    def g(self):
        print("Grandparent")

class Parent(Grandparent):
    def p(self):
        print("Parent")

class Child(Parent):
    pass

obj = Child()
obj.g()
obj.p()
```

---

4. Hierarchical Inheritance

```python
class Parent:
    def show(self):
        print("Parent")

class Child1(Parent):
    pass

class Child2(Parent):
    pass
```

---

5. Hybrid Inheritance

```
class A:
    pass

class B(A):
    pass

class C(A):
    pass

class D(B, C):
    pass
```