7. Inheritance

❖Single, Multilevel, Multiple, Hierarchical, and Hybrid inheritance in Python.

1. Single Inheritance :

Single inheritance means one child class inherits from one parent class.

Syntax:

```
class A:
    def fun1(self):
        print("Hello from A!")
class B(A):  # B inherits from A
    def fun2(self):
        print("Hello from B!")


obj = B()
obj.fun1()
obj.fun2()
```

2. Multilevel Inheritance :

Multilevel inheritance means a class inherits from a child class which is itself derived from another class.

Syntax:

```
class A:
    def fun1(self):
        print("Level 1")
class B(A):  # B inherits A
    def fun2(self):
        print("Level 2")
class C(B):  # C inherits B (and A through B)
    def fun3(self):
        print("Level 3")


obj = C()
obj.fun1()
obj.fun2()
obj.fun3()
```

3. Multiple Inheritance :

Multiple inheritance means one class inherits from more than one class.

Syntax:

```
class A:

    def fun1(self):

        print("From A")


    def fun2(self):

        print("From A - 2")


class B:

    def fun3(self):

        print("From B")


class C(A, B):  # Inherits from A and B

    def fun4(self):

        print("From C")


obj = C()
```

```
obj.fun1()
obj.fun2()
obj.fun3()
obj.fun4()
```

4. Hierarchical Inheritance :

Hierarchical inheritance means multiple child classes inherit from a single parent class.

Syntax:

```
class A:
    def fun1(self):
        print("Parent A")

class B(A):  # Child 1
    def fun2(self):
        print("Child B")

class C(A):  # Child 2
```

```
    def fun3(self):
        print("Child C")

obj1 = B()
obj2 = C()

obj1.fun1()
obj1.fun2()

obj2.fun1()
obj2.fun3()
```

5.Hybrid Inheritance :

Hybrid inheritance combines more than one type of inheritance.

Syntax:

```
class A:
    def fun1(self):
        print("From A")


class B(A):

    def fun2(self):

        print("From B")


class C:

    def fun3(self):

        print("From C")


class D(B, C):  # Hybrid: multilevel (A→B→D) + multiple (C)

    def fun4(self):

        print("From D")


obj = D()

obj.fun1()

obj.fun2()

obj.fun3()

obj.fun4()
```

❖Using the super() function to access properties of the parent class.

What super() Does :

- **Creates a proxy** to the next class in the **Method Resolution Order (MRO)**—not always the direct parent.
- This allows your subclass to call or delegate execution upwards—useful for extending or combining behavior.

Why Use super()?

- In single inheritance, it saves you from hard-coding the parent class name.
- In **multiple inheritance**, it ensures each class in the MRO is called **once and in correct order**, avoiding duplicated or missing calls.

Syntax:

```
class A:
    def __init__(self):
        self.msg = "Hello from A"

    def fun1(self):
        print(self.msg)

class B(A):
    def __init__(self):
        super().__init__()
        self.msg = "Hello from B"

    def fun1(self):
        super().fun1()
        print("Hello 2 from B")

obj = B()
obj.fun1()
```