Inheritance in Python

Concept Overview

Inheritance is an **Object-Oriented Programming (OOP)** concept that allows one class (the *child* or *derived* class) to acquire the **properties and behaviors (attributes and methods)** of another class (the *parent* or *base* class).
It promotes **code reusability**, **organization**, and **extensibility**.

⚙️ Types of Inheritance in Python

1️⃣ Single Inheritance

A child class inherits from **one** parent class.

```
class Parent:
    def display(self):
        print("This is the Parent class.")


class Child(Parent):
    def show(self):
        print("This is the Child class.")


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

🧩 *Output:*

```
This is the Parent class.
This is the Child class.
```

2️⃣ Multilevel Inheritance

Inheritance occurs through **multiple levels** (like a family tree).

```
class GrandParent:
    def feature1(self):
        print("Feature 1 from GrandParent")

class Parent(GrandParent):
    def feature2(self):
        print("Feature 2 from Parent")

class Child(Parent):
    def feature3(self):
        print("Feature 3 from Child")

obj = Child()
obj.feature1()
obj.feature2()
obj.feature3()
```

3️⃣ Multiple Inheritance

A child class inherits from **two or more parent classes**.

```
class Father:
    def skill1(self):
        print("Father: Coding")

class Mother:
    def skill2(self):
        print("Mother: Designing")

class Child(Father, Mother):
    def skill3(self):
        print("Child: Both Skills")

obj = Child()
obj.skill1()
obj.skill2()
obj.skill3()
```

4️⃣ Hierarchical Inheritance

Multiple child classes inherit from the **same parent** class.

```
class Parent:
    def show(self):
        print("This is the Parent class.")

class Child1(Parent):
    def feature1(self):
        print("Feature from Child1")

class Child2(Parent):
    def feature2(self):
        print("Feature from Child2")

obj1 = Child1()
obj2 = Child2()
obj1.show()
obj2.show()
```

5️⃣ Hybrid Inheritance

A combination of **two or more inheritance types**.

```
class A:
    def showA(self):
        print("Class A")

class B(A):
    def showB(self):
        print("Class B")

class C(A):
    def showC(self):
        print("Class C")

class D(B, C):
    def showD(self):
        print("Class D")

obj = D()
obj.showA()
obj.showB()
obj.showC()
obj.showD()
```

🧠 Using the super() Function

The super() function allows you to **access methods or properties** of the **parent class** without explicitly naming it.
It's mainly used in **method overriding** to extend or modify the parent's behavior.

```
class Parent:
    def greet(self):
        print("Hello from Parent")


class Child(Parent):
    def greet(self):
        super().greet()  # Call the parent method
        print("Hello from Child")


obj = Child()
obj.greet()
```

🧩 *Output:*

```
Hello from Parent
Hello from Child
```

🏁 **Key Takeaways**

- ✅ Inheritance enables **code reuse** and **simplifies maintenance**.
- 🧩 The super() function helps you **reuse parent methods** efficiently.
- ⚠️ In multiple inheritance, Python uses the **MRO (Method Resolution Order)** to determine which parent's method runs first.