**Inheritance in Python**

Inheritance is a fundamental concept of Object-Oriented Programming (OOP) in Python that allows a class (child class) to inherit attributes and methods from another class (parent class). This promotes code reusability and a hierarchical relationship between classes.

Python supports various types of inheritance, including **multiple inheritance** and **multilevel inheritance**.

**1. Multiple Inheritance**

**Definition**

Multiple inheritance allows a child class to inherit from more than one parent class. This means the child class will have access to the attributes and methods of all its parent classes.

**Syntax**

class Parent1:

def method1(self):

print("Method from Parent1")

class Parent2:

def method2(self):

print("Method from Parent2")

class Child(Parent1, Parent2):

def method3(self):

print("Method from Child")

# Creating an instance of Child

obj = Child()

obj.method1() # Inherited from Parent1

obj.method2() # Inherited from Parent2

obj.method3() # Defined in Child

**Output**

Method from Parent1

Method from Parent2

Method from Child

**Key Points**

* The Child class inherits from both Parent1 and Parent2.
* It can use methods from both parent classes.
* The method resolution order (MRO) determines the order in which Python looks for a method in case of method name conflicts.

**2. Multilevel Inheritance**

**Definition**

Multilevel inheritance occurs when a class is derived from another derived class, forming a chain of inheritance.

**Syntax**

class Grandparent:

def method1(self):

print("Method from Grandparent")

class Parent(Grandparent):

def method2(self):

print("Method from Parent")

class Child(Parent):

def method3(self):

print("Method from Child")

# Creating an instance of Child

obj = Child()

obj.method1() # Inherited from Grandparent

obj.method2() # Inherited from Parent

obj.method3() # Defined in Child

**Output**

Method from Grandparent

Method from Parent

Method from Child

**Key Points**

* Child inherits from Parent, and Parent inherits from Grandparent.
* The Child class has access to methods from both Parent and Grandparent.
* It forms a hierarchical structure.

**Comparison of Multiple vs Multilevel Inheritance**

| **Feature** | **Multiple Inheritance** | **Multilevel Inheritance** |
| --- | --- | --- |
| **Definition** | Child class inherits from multiple parent classes | Child class inherits from a parent class, which itself inherits from another class |
| **Complexity** | More complex due to possible conflicts | Less complex, follows a hierarchical structure |
| **Example Use** | Used when a class needs functionalities from multiple sources | Used when a class needs to extend an existing hierarchy |

**Conclusion**

* **Multiple inheritance** is useful when a class needs features from multiple parent classes, but it should be used carefully to avoid ambiguity.
* **Multilevel inheritance** is useful when building a hierarchical structure, but too many levels can make debugging difficult.