**Multiple Inheritance**

* **Definition**

In Python, multiple inheritance means that a class can inherit attributes and methods from more than one parent class.  
This allows the child class to use features from multiple sources at the same time.

class A:

def method\_a(self):

print("Method from A")

class B:

def method\_b(self):

print("Method from B")

class C(A, B):

pass

obj = C()

obj.method\_a() # Method from A

obj.method\_b() # Method from B

* **What happens if the child and the parent have the same method?**

If the child class defines a method with the same name as in the parent, the child’s version overrides the parent’s version.

class A:

def greet(self):

print("Hello from A")

class B(A):

def greet(self):

print("Hello from B")

obj = B()

obj.greet() # Hello from B

* **What happens if two parents have the same method?**

Python uses **Method Resolution Order (MRO)** to decide which parent’s method to call. MRO follows the order in which the parents are listed when defining the class (left to right).

class A:

def greet(self):

print("Hello from A")

class B:

def greet(self):

print("Hello from B")

class C(A, B):

pass

obj = C()

obj.greet() # Hello from A

* **What happens if two parents share the same parent?**

Python still uses MRO to make sure the shared parent’s methods are not called more than once. This avoids duplication problems.

class Grandparent:

def greet(self):

print("Hello from Grandparent")

class Parent1(Grandparent):

pass

class Parent2(Grandparent):

pass

class Child(Parent1, Parent2):

pass

obj = Child()

obj.greet() #Hello from Grandparent