Multiple Inheritance in Python

Prepared by: Yousef Hamdy
Date: 2025/8/14

Inheritance is a fundamental concept in object-oriented programming (OOP) that allows a class to acquire attributes and methods from another class. **Multiple inheritance** is a specific form where a child class inherits from more than one parent class.
This report explains multiple inheritance in Python, the behavior when methods are overridden, and how Python handles method resolution when multiple parent classes are involved.

---

1. What is Multiple Inheritance?

Definition:
Multiple inheritance occurs when a class derives from two or more parent classes. This allows the child class to combine functionalities from multiple sources.

Example:

```python
class Parent1:
    def greet(self):
        print("Hello from Parent1")

class Parent2:
    def farewell(self):
        print("Goodbye from Parent2")

class Child(Parent1, Parent2):
    pass

c = Child()
c.greet()     # Inherited from Parent1
c.farewell()  # Inherited from Parent2
```

2. What Happens if the Child and Parent Have the Same Method?

If a child class defines a method with the same name as one in its parent, the **child's version overrides the parent's version** when called on the child object. This is known as **method overriding**.

Example:

Result: Python always uses the **most specific** version of the method (closest to the object in the

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

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

c = Child()
c.greet()  # Output: Hello from Child
```

inheritance hierarchy).

---

3. What Happens if Two Parents Have the Same Method?

When two or more parent classes define the same method name, Python uses the **Method Resolution Order (MRO)** to determine which method to call.
The MRO is a predefined order in which Python searches classes for a method.

Example:

```
class Parent1:
    def greet(self):
        print("Hello from Parent1")

class Parent2:
    def greet(self):
        print("Hello from Parent2")

class Child(Parent1, Parent2):
    pass

c = Child()
c.greet()  # Output: Hello from Parent1
```

Explanation:
In the above case, Python searches in the following order (which can be seen using Child.mro()):
Child → Parent1 → Parent2 → object
Since Parent1 appears first in the inheritance list, its method is used.

---

4. What Happens if Two Parents Have the Same Parent?

If both parent classes inherit from the same grandparent class, Python ensures the **grandparent's methods are not duplicated in the search path** due to the **C3 linearization algorithm**.

Example:

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

class Parent1(Grandparent):
    pass

class Parent2(Grandparent):
    pass

class Child(Parent1, Parent2):
    pass

c = Child()
c.greet()  # Output: Hello from Grandparent
print(Child.mro())
```

Result & MRO Order:
[Child, Parent1, Parent2, Grandparent, object]
Python calls the method from Grandparent only once, even though it is inherited through two separate paths.

---

5. Conclusion

- **Multiple inheritance** allows a child class to inherit from more than one parent, enabling the combination of different functionalities.

- If the child and parent share a method name, the child's method overrides the parent's version.
- If two parents share the same method, Python follows the **MRO** to decide which one to execute, prioritizing the first parent listed.
- If two parents share the same ancestor, Python ensures that the ancestor is only searched once, maintaining a consistent and efficient lookup order.