Report: Multiple Inheritance in Python

1. How super() handles Multiple Inheritance

In Python, super() is used to call a method from a parent class. When there is multiple inheritance, Python follows the Method Resolution Order (MRO). MRO defines which parent class is called first when super() is used. In multiple inheritance, super() does not just call one fixed parent, but follows the MRO chain. This means all parent classes can get called in a predictable order if super() is used correctly.

Example:

```
class A:
    def show(self):
        print("A")

class B(A):
    def show(self):
        print("B")
        super().show()

class C(A):
    def show(self):
        print("C")
        super().show()

class D(B, C):
    def show(self):
        print("D")
        super().show()

d = D()
d.show()
```

Output:

```
D
B
C
A
```

Explanation: The method resolution order is D -> B -> C -> A. super() follows this order automatically.

2. If Human and Mammal have the same method (eat)

If both parent classes have a method with the same name, Python uses the MRO to decide which one runs. For example:

```
class Human:
    def eat(self):
        print("Human is eating with hands")

class Mammal:
    def eat(self):
        print("Mammal is eating in general")


class Employee(Human, Mammal):
    def work(self):
        print("Employee is working")


emp = Employee()
emp.eat()
```

Output:

```
Human is eating with hands
```

Explanation: Employee inherits Human first, then Mammal, so Python uses the eat method from Human.

If you use super():

```
class Human:
    def eat(self):
        print("Human is eating with hands")


class Mammal:
    def eat(self):
        print("Mammal is eating in general")
```

```
class Employee(Human, Mammal):
    def eat(self):
        print("Employee wants to eat:")
        super().eat()


emp = Employee()
emp.eat()
```

Output:

```
Employee wants to eat:
Human is eating with hands
```

Opinion

In my opinion, super() is a powerful tool in multiple inheritance if used properly with MRO. It helps call parent methods in the right order without repeating code. However, it can be confusing if classes are not designed well. Using clear inheritance order and super() keeps the code clean and predictable.

Conclusion

super() follows MRO and calls methods in the correct order. When multiple parents have the same method, Python uses the MRO to choose which one runs first. This makes multiple inheritance safer and more predictable.