Report

1. How super() Handles Multiple Inheritance

In Python, super() is used to call methods from a parent or sibling class without hardcoding the parent class name. It works by following the **Method Resolution Order (MRO)**, which determines the order in which base classes are searched when executing a method.

Multiple Inheritance & super()

In multiple inheritance, super() doesn't necessarily refer to the immediate parent—it refers to the **next class in the MRO**.

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
```

2. If Human and Mammal Have the Same Method (like eat) but with Different Implementations. When Child [Employee] Calls eat(), How Does Python Handle This?

Example:

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


class Mammal:
    def eat(self):
        print("Mammal is eating with mouth.")


class Employee(Human, Mammal):
    pass


e = Employee()
e.eat()
```

Output:

```
Human is eating with hands.
```

** In cases where multiple parent classes define the same method (eat), and the child class inherits from both, Python uses the MRO to decide which method to call.