Report Multiple Inheritance

1. How super Function handle Multiple Inheritance

The super () function in Python is used to delegate method calls to parent or sibling classes in the method resolution order (MRO). In the case of multiple inheritance, super () ensures that each class in the inheritance chain is only initialized once.

Multiple inheritance is a feature in object-oriented programming where a class can inherit from more than one parent class. In Python, this is allowed and used through this syntax:

```
class Child(Parent1, Parent2):
    pass
```

How super() works in Multiple Inheritance?

Python uses the C3 Linearization (MRO algorithm) to determine the order in which classes are initialized or methods are resolved. Super () goes through this MRO and ensures that every class in the hierarchy is called exactly once.

<u>Example: Proving super () with Multiple Inheritance</u>

```python
class A:
    def __init__(self):
        print("A initialized")
        super().__init__()
class B(A):
    def __init__(self):
        print("B initialized")
        super().__init__()
class C(A):
    def __init__(self):
        print("C initialized")
        super().__init__()
class D(B, C):
    def __init__(self):
        print("D initialized")
        super().__init__()
d = D()
```

output

```
D initialized
B initialized
C initialized
A initialized
```

Explanation:

- Class D inherits from B and C, both of which inherit from A.
- Each class calls super (). __ init __ ().
- The MRO for class D is: **D -> B -> C -> A -> object**

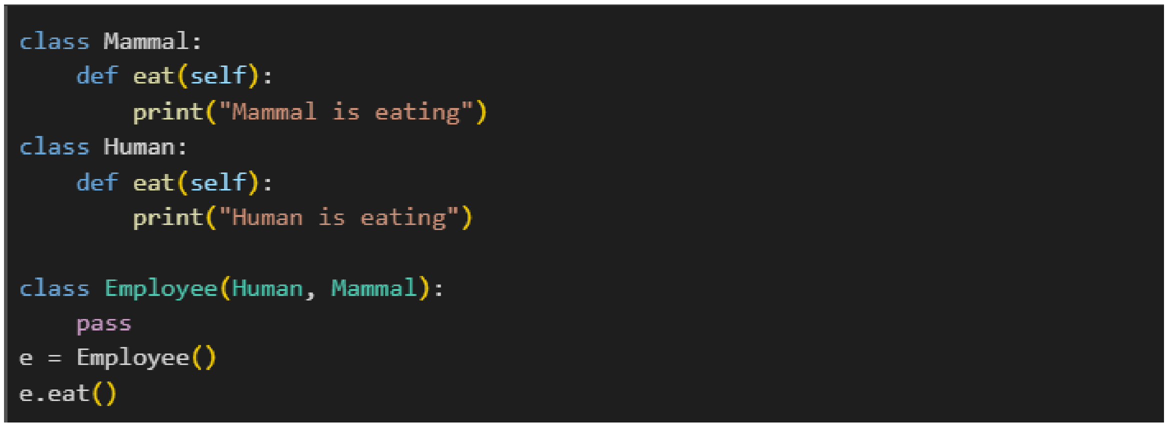
Using super () ensures that each class's __ init __ method is called only once and in the correct order as defined by MRO.

Summary:

- Super () ensures clean method delegation by following MRO.
- In multiple inheritance, it prevents duplicate method calls.
- It is safer and more maintainable than hardcoding parent class names.

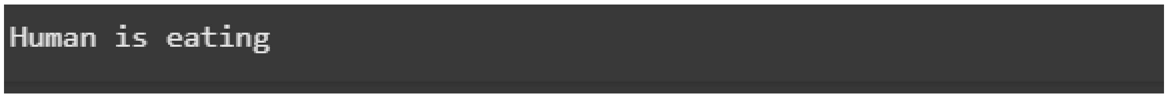
2. **If Human and Mammal Have the same method like eat but with different Implementation. When Child[Employee] calls eat method how python handle this case.**
When a child class like Employee inherits from both Human and Mammal, and both parents have a method with the same name (eat()), Python resolves the method to the first match in the MRO unless you explicitly chain methods using super ().

<u>Example 1: Without super()</u>



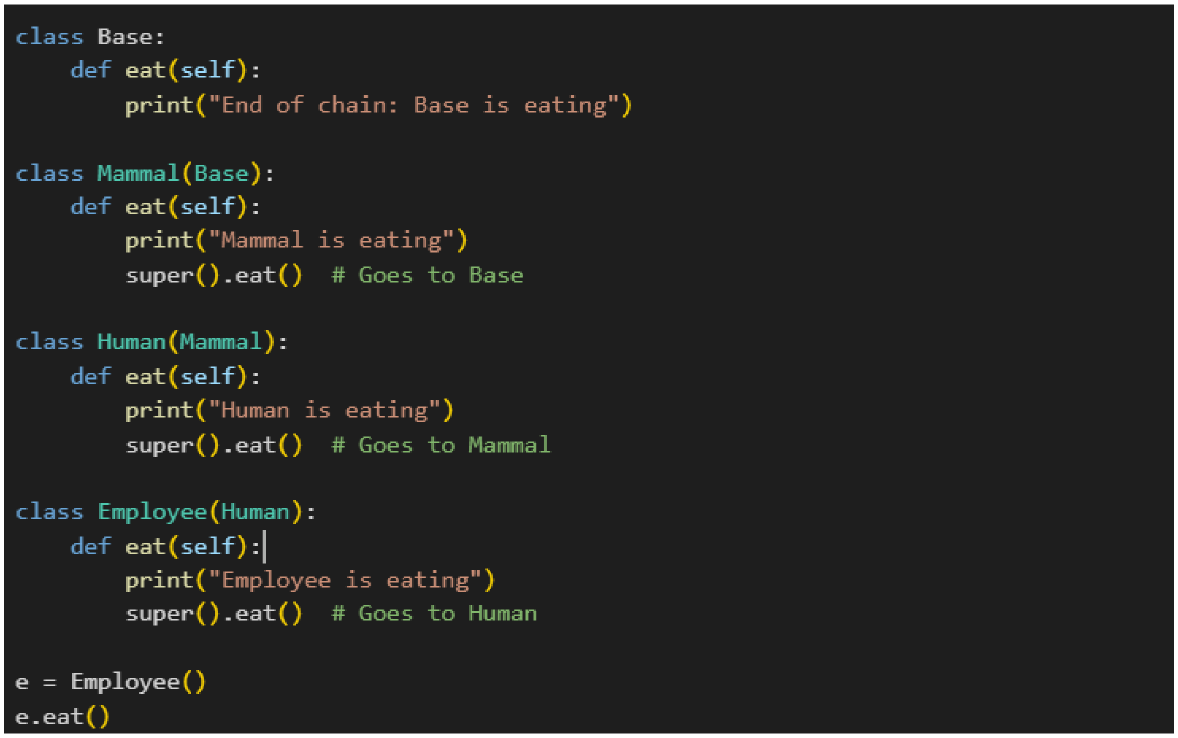
```python
class Mammal:
    def eat(self):
        print("Mammal is eating")
class Human:
    def eat(self):
        print("Human is eating")

class Employee(Human, Mammal):
    pass
e = Employee()
e.eat()
```

Output



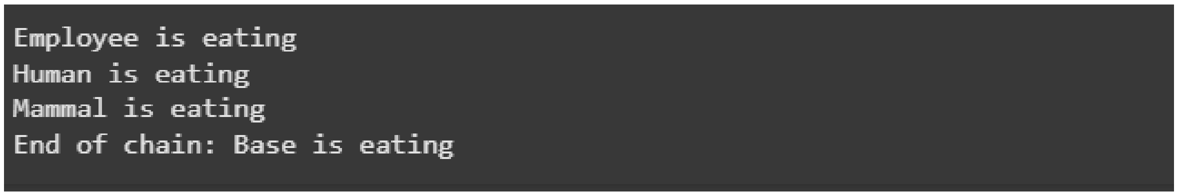
```
Human is eating
```

Example 2: With super() to chain both



```python
class Base:
    def eat(self):
        print("End of chain: Base is eating")

class Mammal(Base):
    def eat(self):
        print("Mammal is eating")
        super().eat()  # Goes to Base

class Human(Mammal):
    def eat(self):
        print("Human is eating")
        super().eat()  # Goes to Mammal

class Employee(Human):
    def eat(self):
        print("Employee is eating")
        super().eat()  # Goes to Human

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

Output



```
Employee is eating
Human is eating
Mammal is eating
End of chain: Base is eating
```

Conclusion:

- Using super() in all classes allows cooperative multiple inheritance, where each method in the hierarchy is called exactly once, in the correct order.
- Without super(), only the first matching method in the MRO is executed.
- This avoids duplication and improves code maintainability

Finally:

Concept	**Description**
super()	Delegates method calls following MRO, ensures each method is called once.
Multiple Inheritance	Handled safely via MRO (C3 Linearization).
Method Conflict (Same Method)	Resolved by choosing the first method in the MRO unless super() is used.
Cooperative Inheritance	Achieved when all classes use super().