Multiple Inheritance Report

1. How super Function handle Multiple Inheritance

When you use super() in Python with multiple inheritance, it doesn't just call the parent class directly; it follows something called the MRO (Method Resolution Order).

How it works?

1) **MRO is a list** of classes Python creates internally that defines the order in which classes are searched for methods.
2) When you call super(), Python:
    - Looks at the **next class in the MRO list** after the current class.
    - Calls the method from that next class.
3) This ensures that **each parent class is called only once**, even if it appears multiple times due to diamond inheritance.

Example:

F: > Data Engieer iti > python > repport.py > ...

```python
class A:
    def __init__(self):
        print("A init")
        super().__init__()

class B:
    def __init__(self):
        print("B init")
        super().__init__()

class C(A, B):
    def __init__(self):
        print("C init")
        super().__init__()

obj = C()
```

Output

```
C:\Users\alaao>C:/Users/alaao/
C init
A init
B init
```

Why this order?

- C → A → B → **object**
- Python generates this order using C3 linearization (MRO).
- Each class calls super(), which moves to the next in the MRO chain.

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 Human and Mammal both define a method eat() with different implementations, and Employee (the child class) inherits from both, Python decides which eat() to call using the Method Resolution Order (MRO).

How Python handles it ?

1. When you call Employee().eat(), Python searches for eat() following the MRO list of the Employee class.
2. It will execute the first eat() method it finds in the MRO chain.
3. The order in which you define the parent classes in the Employee class affects the MRO.

Example

```
1   class Mammal:
2       def eat(self):
3           print("Mammal is eating")
4
5   class Human:
6       def eat(self):
7           print("Human is eating")
8
9   class Employee(Human, Mammal):
10      pass
11
12  emp = Employee()
13  emp.eat()
```

Output

```
C:\Users\alaao>C:/Users/alaao/AppData/Local
Human is eating
```

Explanation:

- MRO for Employee = [Employee, Human, Mammal, object]
- Python finds eat() in Human first, so it calls that.

Example

```
class Employee(Mammal, Human):
    pass

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

Output

```
Mammal is eating
```

Explanation:

- MRO for Employee = [Employee, Mammal, Human, object]
- Now Mammal comes first, so Python calls that.