Multiple Inheritance

1-How super Function handle Multiple Inheritance?

- Super function is used to give you access to methods from a parent or sibling class, which is especially helpful in multiple inheritance scenarios.
- Super function plays an important role in multiple inheritance and helps drive the flow of the code execution. It helps in managing or determining the control of from where I can draw the values of my desired functions and variables.
- Super returns the parent, but in the case of multiple inheritance, it returns the first parent.
- Python uses something called the MRO (Method Resolution Order) to determine which parent class's method should be called when using super.
- Super returns a proxy object that delegates method calls to the next class in the MRO. It's often used in class methods to call a method from a parent or sibling class.
- Use super consistently in all classes involved in multiple inheritance to ensure each class gets called exactly once in MRO order.

Example

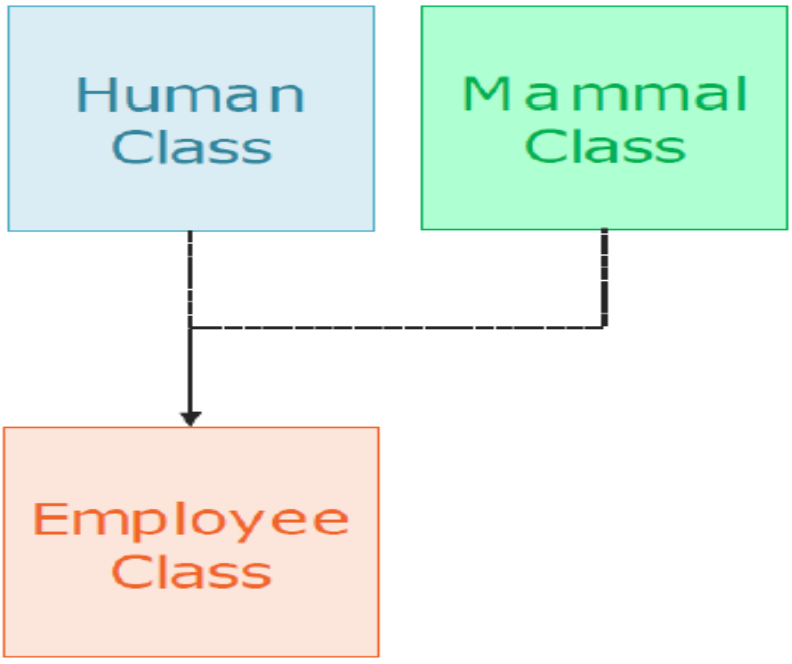
```
73
74  class A:
75      def __init__(self):
76          print("Initializing A")
77
78  class B(A):
79      def __init__(self):
80          super().__init__()
81          print("Initializing B")
82
83  class C(A):
84      def __init__(self):
85          super().__init__()
86          print("Initializing C")
87
88  class D(B, C):
89      def __init__(self):
90          super().__init__()
91          print("Initializing D")
92
93  d = D()
```

In this example, D inherits from both B and C, which themselves inherit from A. The super function in each class's constructor ensures that the initializers of the parent classes are called in the proper order (A, B, C). This maintains consistency and avoids potential conflicts in method resolution.

The Result:

```
Initializing A
Initializing C
Initializing B
Initializing D
PS C:\Users\youse>
```

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.



```python
class Human:
    def eat(self):
        print("Human eats")

class Mammal:
    def eat(self):
        print("Mammal eats ")

class Employee(Human, Mammal):
    pass

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

Python looks for the eat() method in the Method Resolution Order (MRO) of the Employee class.

So, if Employee inherits from Human first, then Mammal, the MRO is going to implement function eat that is in human class

And if Employee inherits from Mammal first, then Human, so it will execute the eat function from Mammal class.

Python looks for the method in the class itself, and if not found, it searches its parents in the order they appear in the class definition.

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

So it looks in: Employee, then Human, then Mammal

Once it finds eat() in Human, it executes the eat function and stops searching .