Multiple Inheritance Report

# 1. How super() Function Handles 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).

1. How it works:

* MRO is a list of classes Python creates internally that defines the order in which classes are searched for methods.
* 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.
* This ensures that each parent class is called only once, even if it appears multiple times due to diamond inheritance.

Example:



Why this order?

• MRO for D = [D, B, C, A, object]  
• Python generates this order using C3 linearization (MRO).  
• Each class calls super(), which moves to the next in the MRO chain.

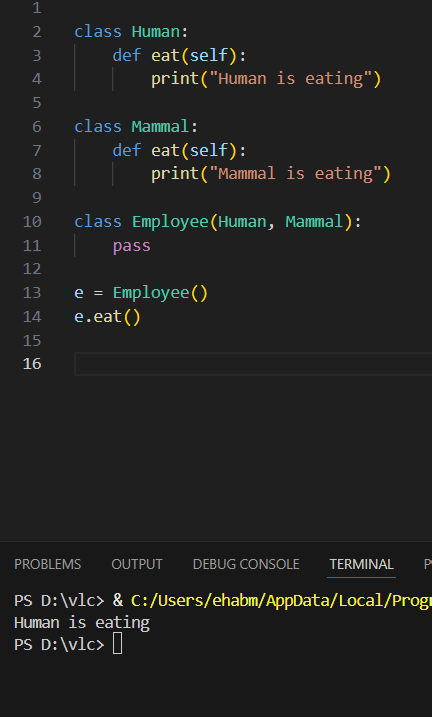
# 2. If Human and Mammal Have the Same Method (eat) with Different Implementations

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).

1. How Python handles it:

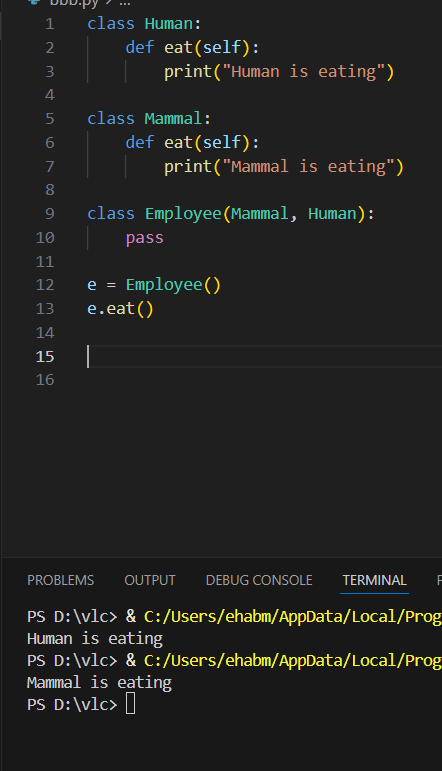
* When you call Employee().eat(), Python searches for eat() following the MRO list of the Employee class.
* It will execute the first eat() method it finds in the MRO chain.
* The order in which you define the parent classes in the Employee class affects the MRO.

Example (Employee inherits from Human, Mammal):



Explanation:  
• MRO for Employee = [Employee, Human, Mammal, object]

Example (Employee inherits from Mammal, Human):



Explanation:  
• MRO for Employee = [Employee, Mammal, Human, object]