**Super( ) Python Report**

**1. How super() Handles Multiple Inheritance in Python**

The super() function is used to call methods from a parent class. In the case of multiple inheritance. When used in a class, super() doesn’t simply refer to the immediate parent — it refers to the next nearest class. This makes super() ensure that all classes in a hierarchy get a chance to execute their methods.

**Example: Multiple Inheritance**

**A black screen with green text

AI-generated content may be incorrect.**

**Output:**

**A black screen with orange text

AI-generated content may be incorrect.**

**Explanation:**

* The method resolution order for class D is D → B → C → A → object.
* Each super().show() call moves to the next class.
* This ensures that each class gets one opportunity to handle the method, without duplication or skipping.

**2. Same Method in Inheritance with Multiple Parents (Overriding):**

Suppose we have two parent classes: Human and Mammal, both define a method named eat(), but with different implementations. A child class Employee inherits from both.

A computer screen shot of text

AI-generated content may be incorrect.

**Output:**

A black background with orange text

AI-generated content may be incorrect.

**Explanation:**

Python uses Method Resolution Order (MRO) to decide which method to call. The order is based on the class definition:

So, when emp.eat() is called:

* Python checks Employee → Human → Mammal → object.
* Since Human has eat(), it is invoked, and Mammal's method is ignored unless explicitly called.

**Using super( ) in Inheritance with Multiple Parents (Overriding):**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Output:**

A black screen with orange text

AI-generated content may be incorrect.

**Explanation:**

* Employee.eat() calls super().eat() → goes to Human.eat()
* This follows: Employee → Human → object