Method Resolution Order (MRO)

The MRO determines the order in which Python searches for methods in a class hierarchy. In the example provided, the MRO for class C is [C, A, B, object]. When super() is called in C, it follows this order, resulting in the sequence of method calls: C -> A -> B.

Understanding MRO is essential for effectively using multiple inheritance in Python, as it ensures that method calls are predictable and consistent. This knowledge helps in designing complex class hierarchies and debugging issues related to method resolution.

Example :

```python
class Father:
    def __init__(self, name, **kwargs):
        print("Father's init called")
        self.name = name
        super().__init__(**kwargs)

class Mother:
    def __init__(self, eye_color, **kwargs):
        print("Mother's init called")
        self.eye_color = eye_color
        super().__init__(**kwargs)

class Child(Father, Mother):
    def __init__(self, name, eye_color):
        print("Child's init called")
        # Use super() to call Father's __init__ and Mother's __init__ in the correct order
        super().__init__(name=name, eye_color=eye_color)
```