Inheritance Report

# Method resolution order using C3 linearization [1](#_refrences)

In Python, super() is used to call the next method in the hierarchy line, following the method resolution order (MRO). The MRO determines the order in which base classes are searched when executing a method. Python uses the C3 linearization or C3 superclass linearization algorithm to compute the MRO in a way that preserves the left-to-right ordering specified in the class definition and is deterministic.

class A:

    def action(self): print("A")

class B(A):

    def action(self):

        super().action()

        print("B")

class C(A):

    def action(self):

        super().action()

        print("C")

class D(B, C):

    def action(self):

        super().action()

        print("D")

D().action()

*# A*

*# C*

*# B*

*# D*

Using super() in a multiple inheritance scenario ensures that all base classes are properly initialized and that the method overriding chains properly delegate calls up the inheritance chain. Here are some cases where you might want to use super():

* In the constructor (\_\_init\_\_) methods to ensure all parent classes are initialized.
* When overriding methods to extend or modify the behavior of inherited methods, ensuring every class in the hierarchy gets a chance to run its logic.

# Human and mammal (MRO)

If we have this:

When we try:

We get “Human” as the response since python follows C3 linearization to decide which method to use.

We also see a class’s MRO:



# refrences

1. https://codemia.io/knowledge-hub/path/how\_does\_pythons\_super\_work\_with\_multiple\_inheritance