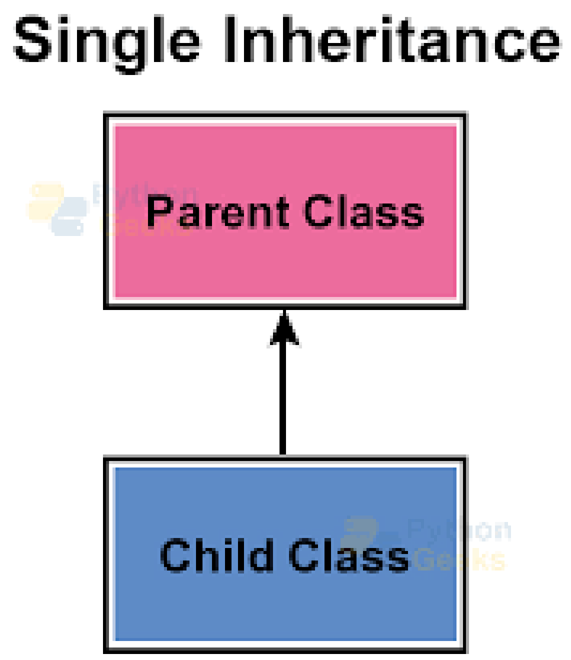
Super Function In Python

Prepared By
Khaled Samy Abdelfattah Ahmed

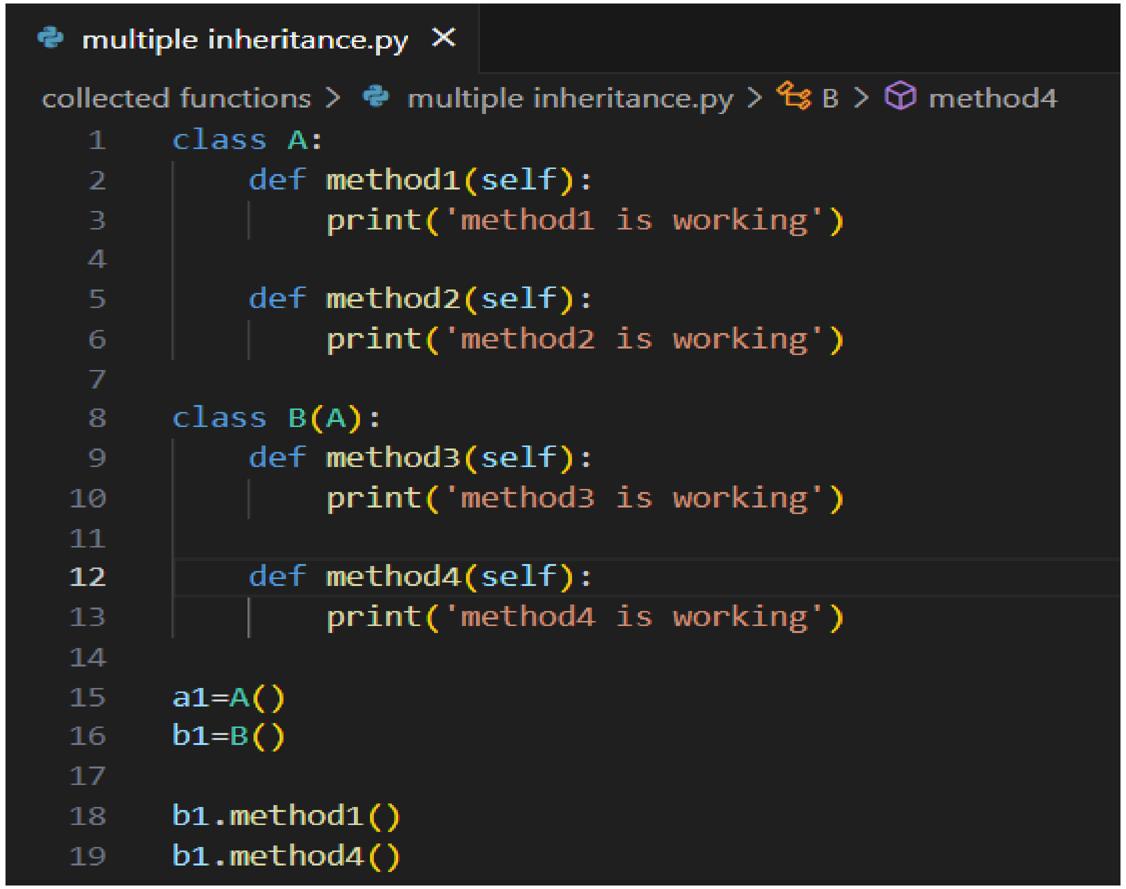
- Why and how we use Super function ?
- How super Function handle Single and Multiple Inheritance ?

- Single Inheritance

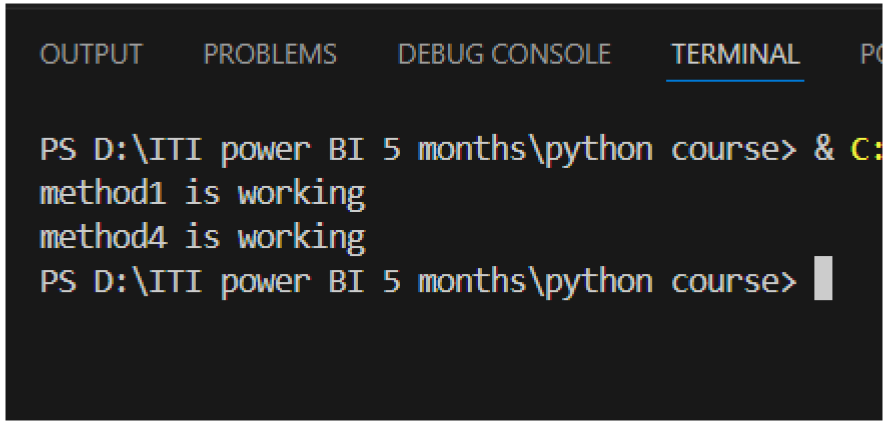


Why Super Function ?

- Child class inherit all Parent class functions.



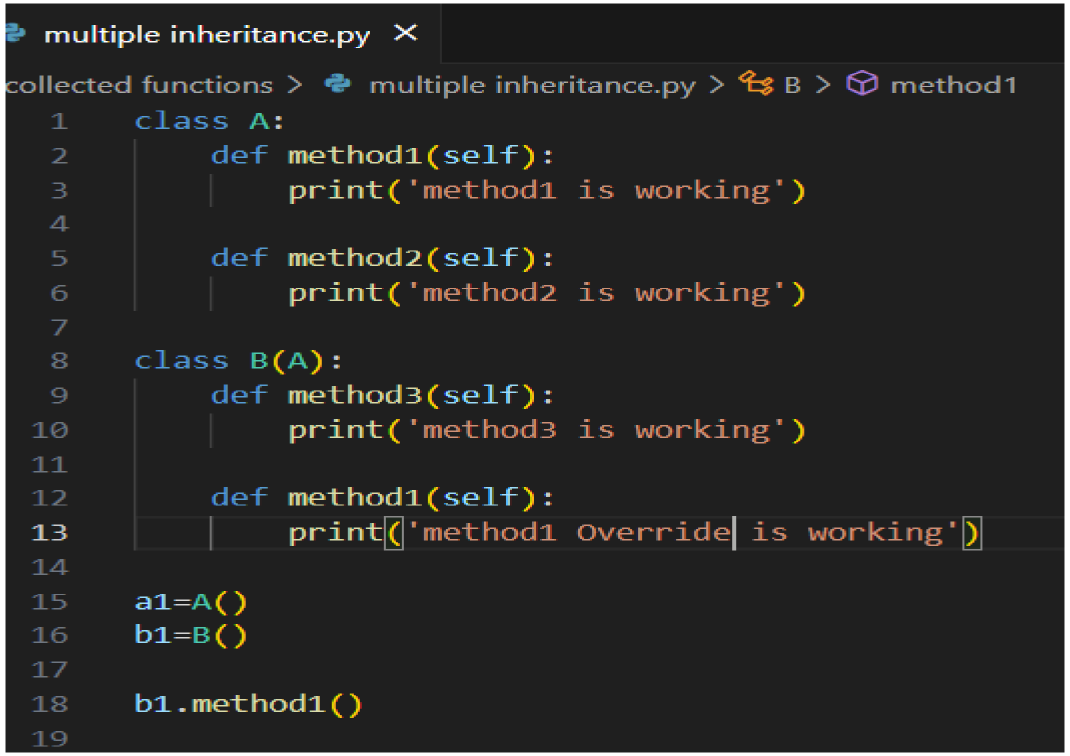
```python
class A:
    def method1(self):
        print('method1 is working')

    def method2(self):
        print('method2 is working')

class B(A):
    def method3(self):
        print('method3 is working')

    def method4(self):
        print('method4 is working')

a1=A()
b1=B()

b1.method1()
b1.method4()
```



```
PS D:\ITI power BI 5 months\python course> & C:
method1 is working
method4 is working
PS D:\ITI power BI 5 months\python course>
```

Why Super Function ?

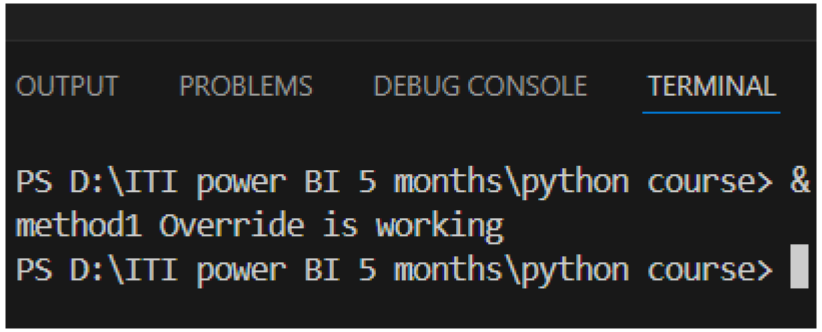
- But child class functions override the Parent class functions.



multiple inheritance.py ×

collected functions > multiple inheritance.py > B > method1

```python
class A:
    def method1(self):
        print('method1 is working')

    def method2(self):
        print('method2 is working')

class B(A):
    def method3(self):
        print('method3 is working')

    def method1(self):
        print('method1 Override is working')

a1=A()
b1=B()

b1.method1()
```

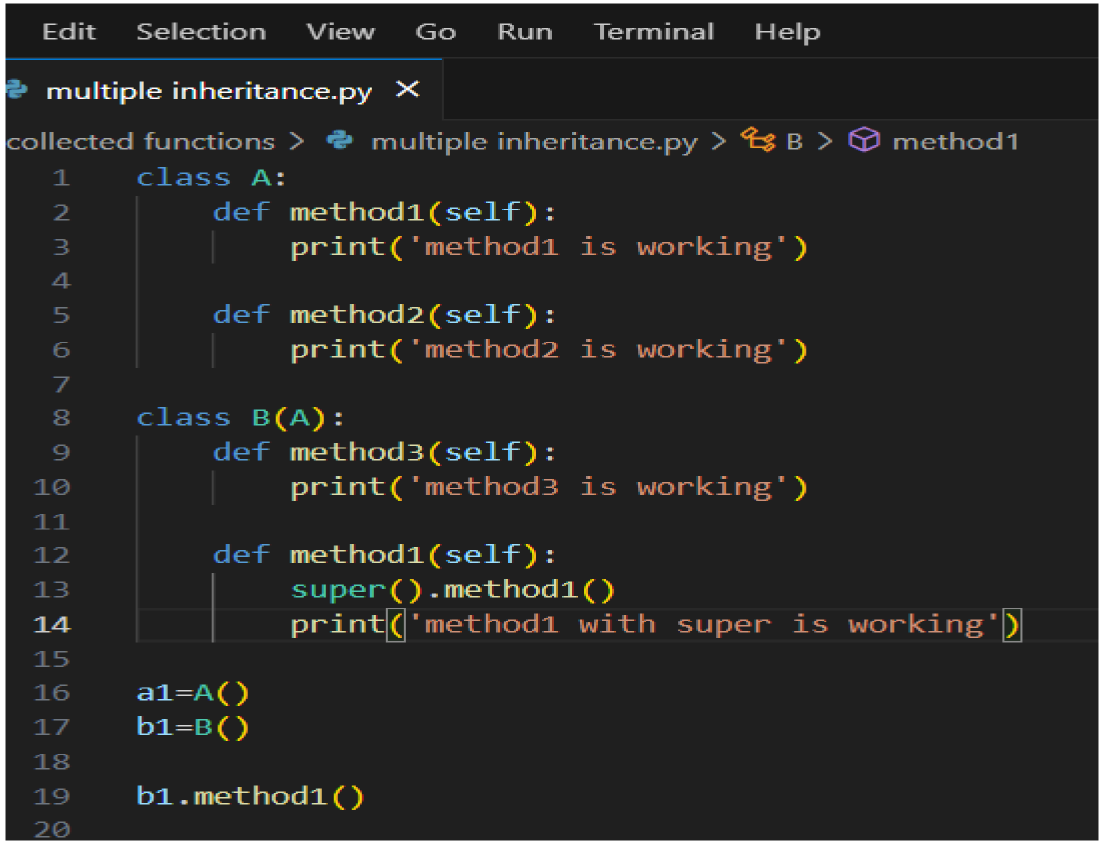


```
PS D:\ITI power BI 5 months\python course> &
method1 Override is working
PS D:\ITI power BI 5 months\python course>
```

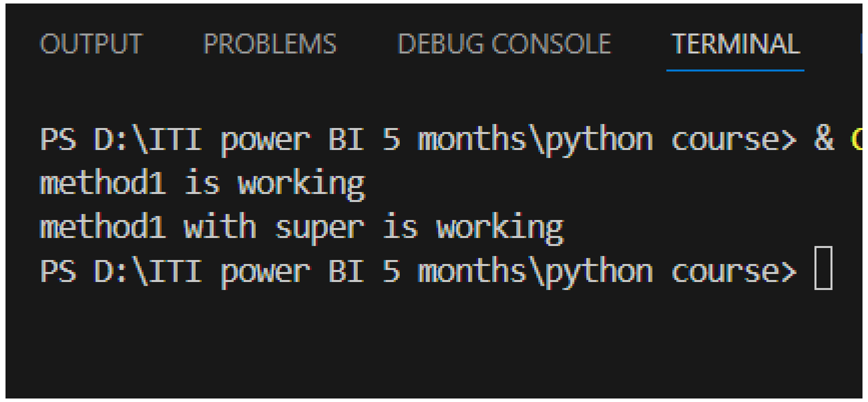
- (method1) in child class overrode (method1) in parent class.

Why Super Function ?

- Super function helps us access and add on Parent class function in Child class without copying or rewriting it.

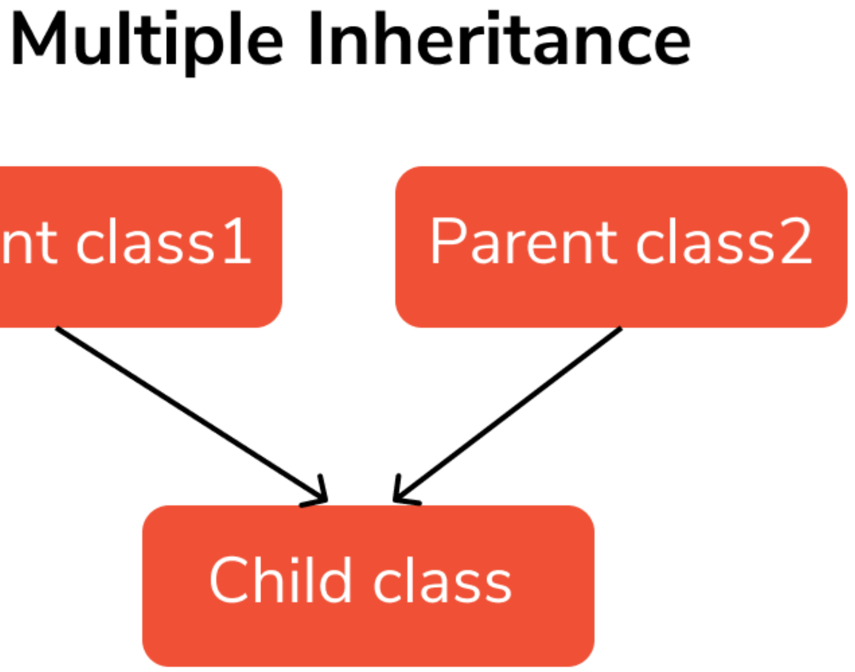

Edit Selection View Go Run Terminal Help
multiple inheritance.py
collected functions > multiple inheritance.py > B > method1

```python
class A:
    def method1(self):
        print('method1 is working')

    def method2(self):
        print('method2 is working')

class B(A):
    def method3(self):
        print('method3 is working')

    def method1(self):
        super().method1()
        print('method1 with super is working')

a1=A()
b1=B()

b1.method1()
```



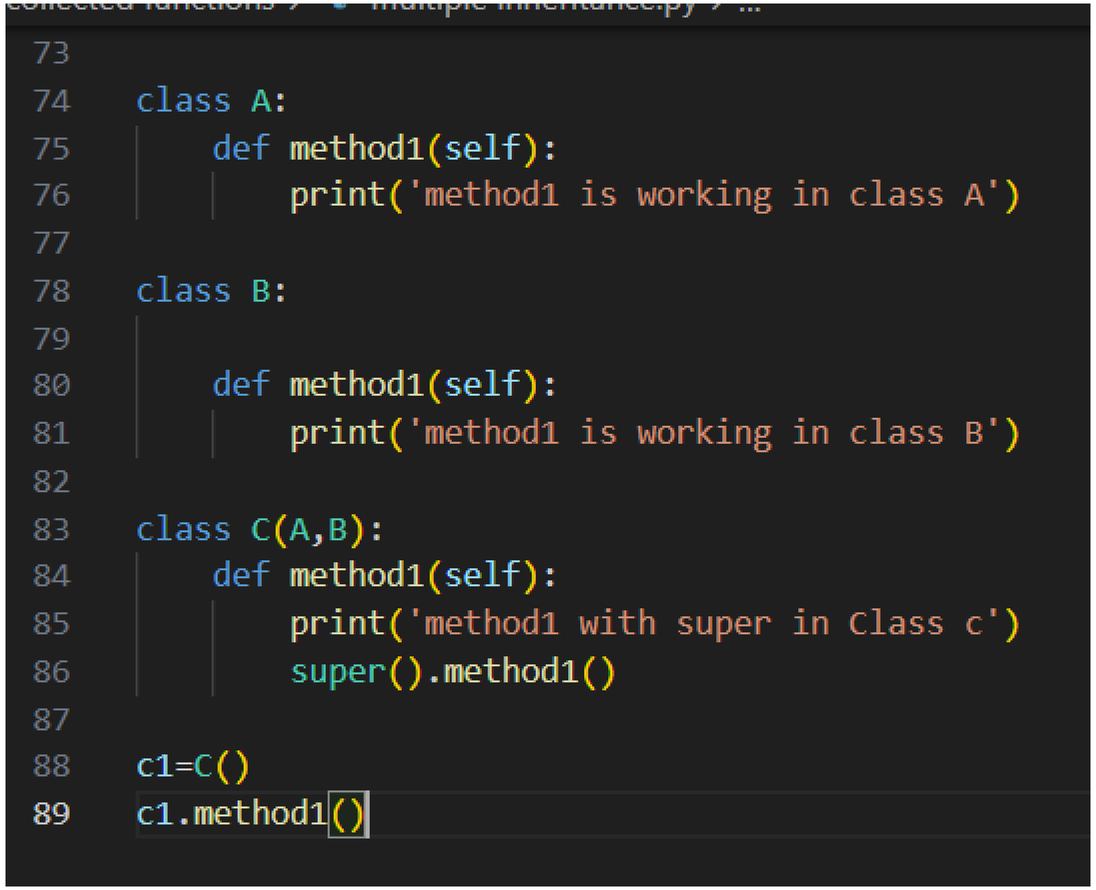
```
PS D:\ITI power BI 5 months\python course> & C
method1 is working
method1 with super is working
PS D:\ITI power BI 5 months\python course>
```

- Multiple Inheritance

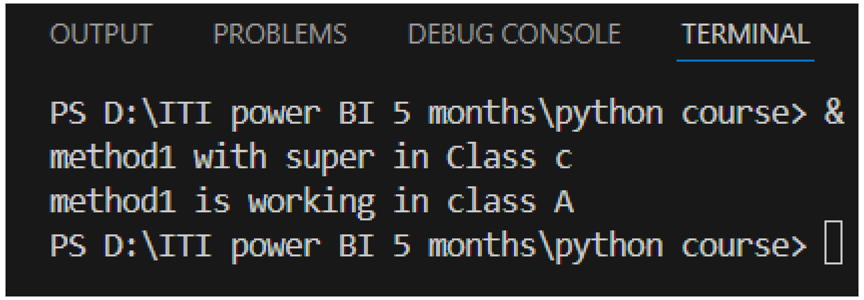


MRO – Method Resolution Order

- MRO defines the order in which Python looks for a method or attribute when using super() in a class hierarchy.



```python
class A:
    def method1(self):
        print('method1 is working in class A')

class B:

    def method1(self):
        print('method1 is working in class B')

class C(A,B):
    def method1(self):
        print('method1 with super in Class c')
        super().method1()

c1=C()
c1.method1()
```



```
PS D:\ITI power BI 5 months\python course> &
method1 with super in Class c
method1 is working in class A
PS D:\ITI power BI 5 months\python course>
```

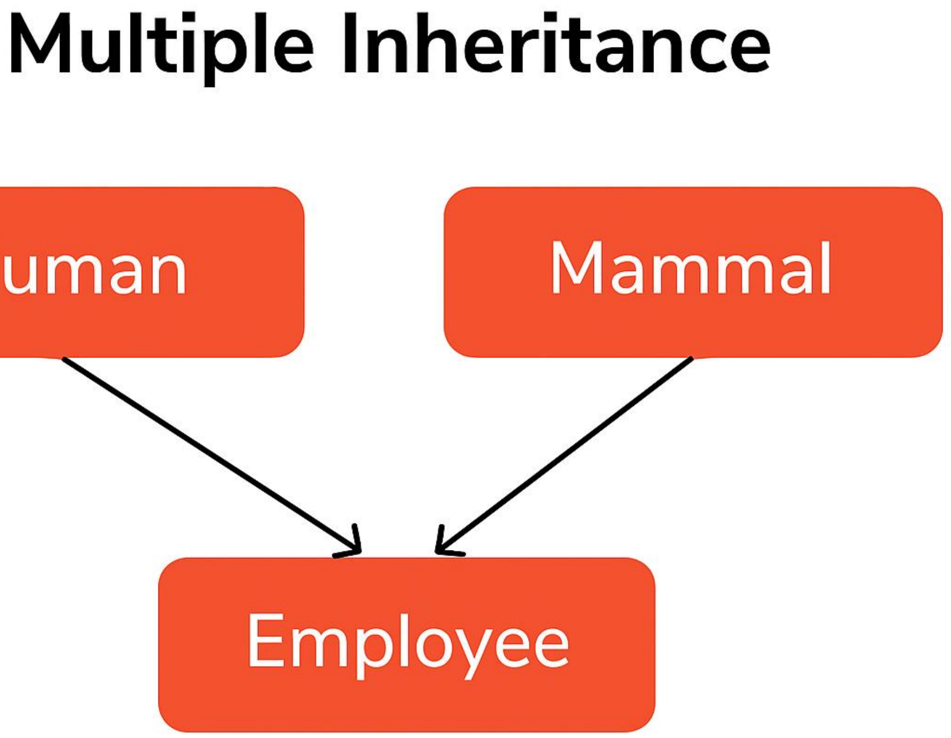
Output illustration

- When we call (method1) on C class object it first go to execute the function version in the class C itself and when encountered Super function it goes to export the function version in Parent class.

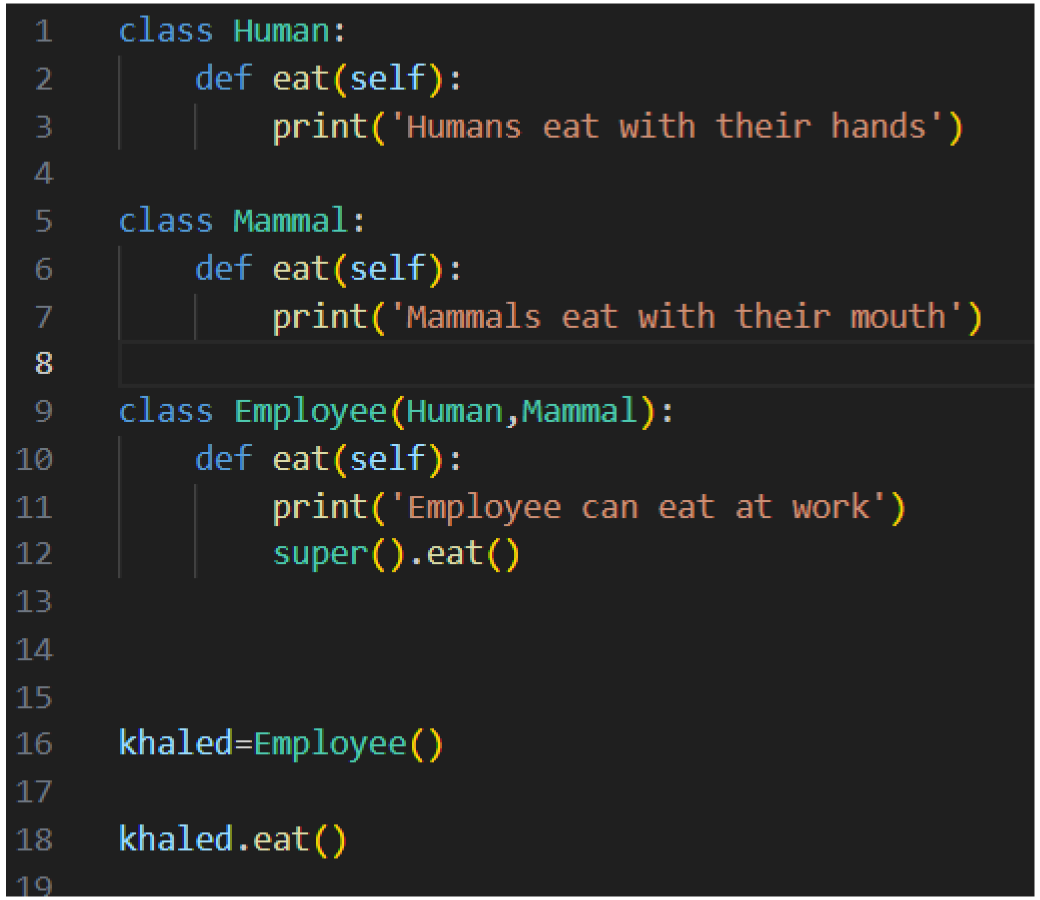
- MRO tells the Super function to go to the version in the first passed Parent class in the Child class definition.
  example : C(A,B)   The MRO orders super function to go to A then B

- As there is no further Super function in the A version of (method1) function the function executes and the MRO chain breaks and stops without going to the function versions in other Multiple Parent classes.

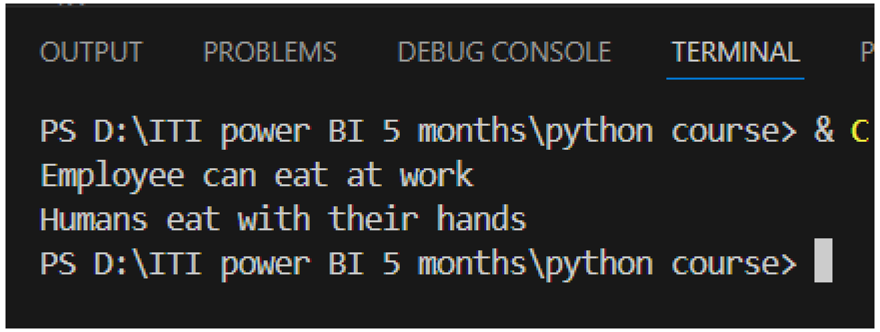
Employee Class



- How Super function in Employee Class will handle the Multiple inheritance from Human and Mammal Classes ?



```python
class Human:
    def eat(self):
        print('Humans eat with their hands')

class Mammal:
    def eat(self):
        print('Mammals eat with their mouth')

class Employee(Human,Mammal):
    def eat(self):
        print('Employee can eat at work')
        super().eat()



khaled=Employee()

khaled.eat()
```



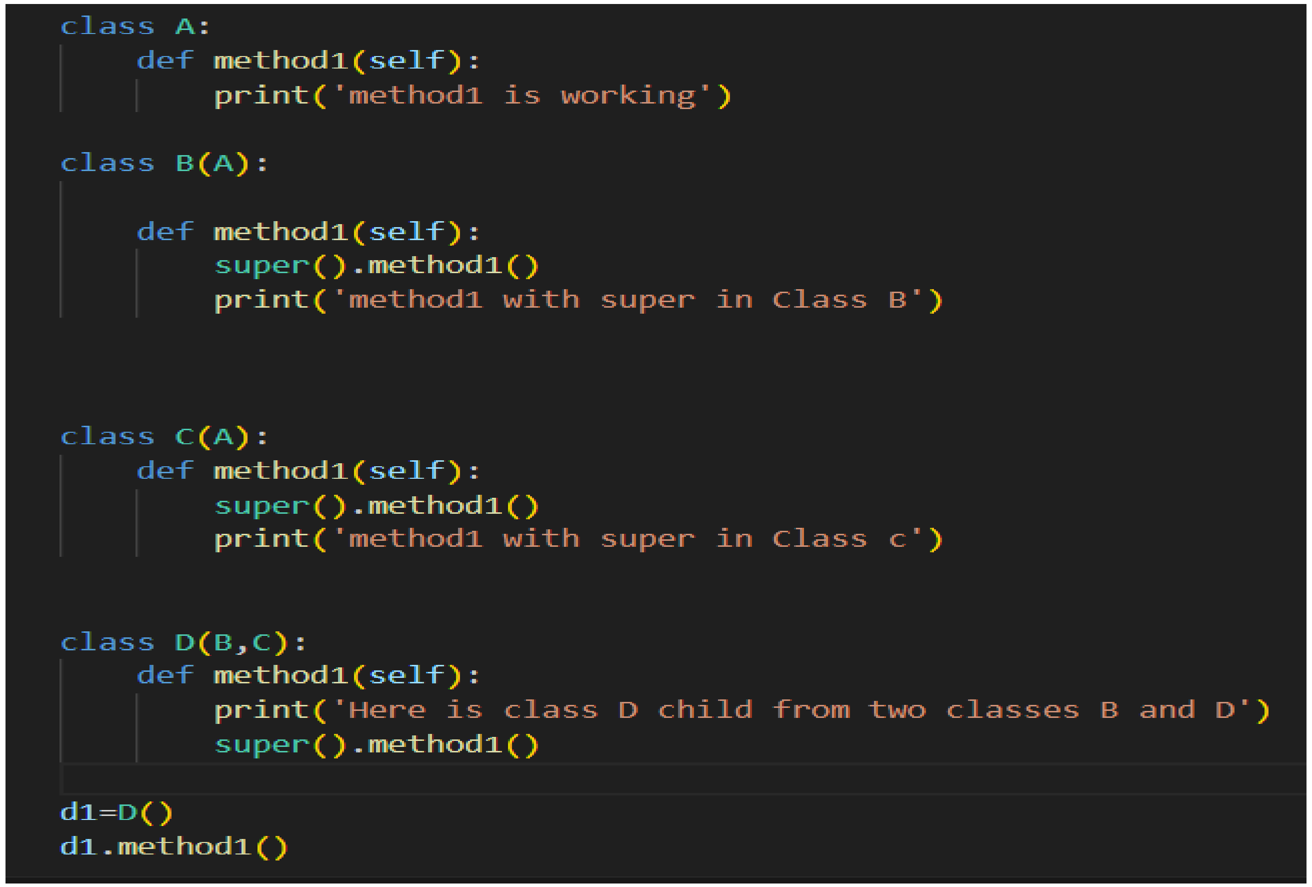
```
PS D:\ITI power BI 5 months\python course> & C
Employee can eat at work
Humans eat with their hands
PS D:\ITI power BI 5 months\python course>
```

Output illustration

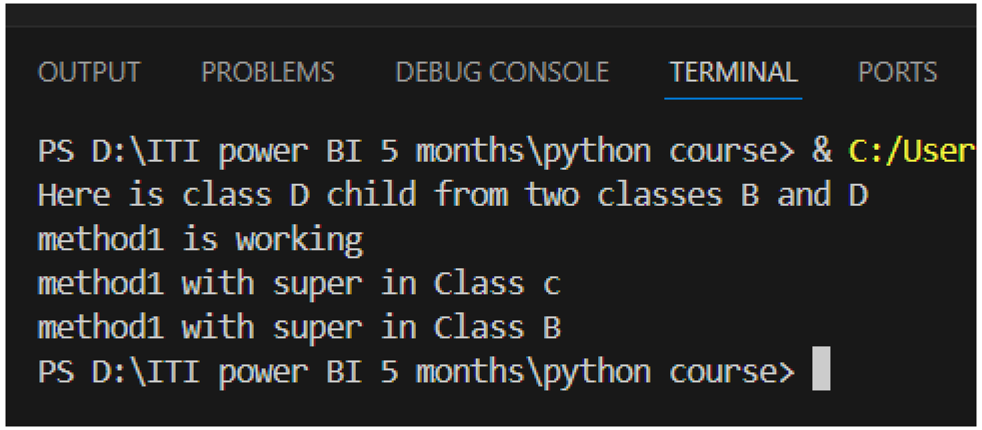
- Only the Class Human version of function (eat) executes because the class Human is the first to come in passed Parent classes inside the Class Employee definition and there is no further Super function in Class Human.

MRO – Method Resolution Order

- What happens when the two Multiple Parent Classes have their shared Parent class ?



```python
class A:
    def method1(self):
        print('method1 is working')

class B(A):

    def method1(self):
        super().method1()
        print('method1 with super in Class B')



class C(A):
    def method1(self):
        super().method1()
        print('method1 with super in Class c')


class D(B,C):
    def method1(self):
        print('Here is class D child from two classes B and D')
        super().method1()

d1=D()
d1.method1()
```



```
PS D:\ITI power BI 5 months\python course> & C:/User
Here is class D child from two classes B and D
method1 is working
method1 with super in Class c
method1 with super in Class B
PS D:\ITI power BI 5 months\python course>
```

Output illustration

- When we call (method1) on D class object it first go to execute the function version in the class D itself and when encountered Super function it goes to export the function version in Parent class.

- But we have Multiple Inheritance ! Here comes the MRO ordering role .

- MRO orders the next super function to the Parent class who comes first in the Child class definition.

  example : D(B,C)   The MRO orders super function to go to B then C

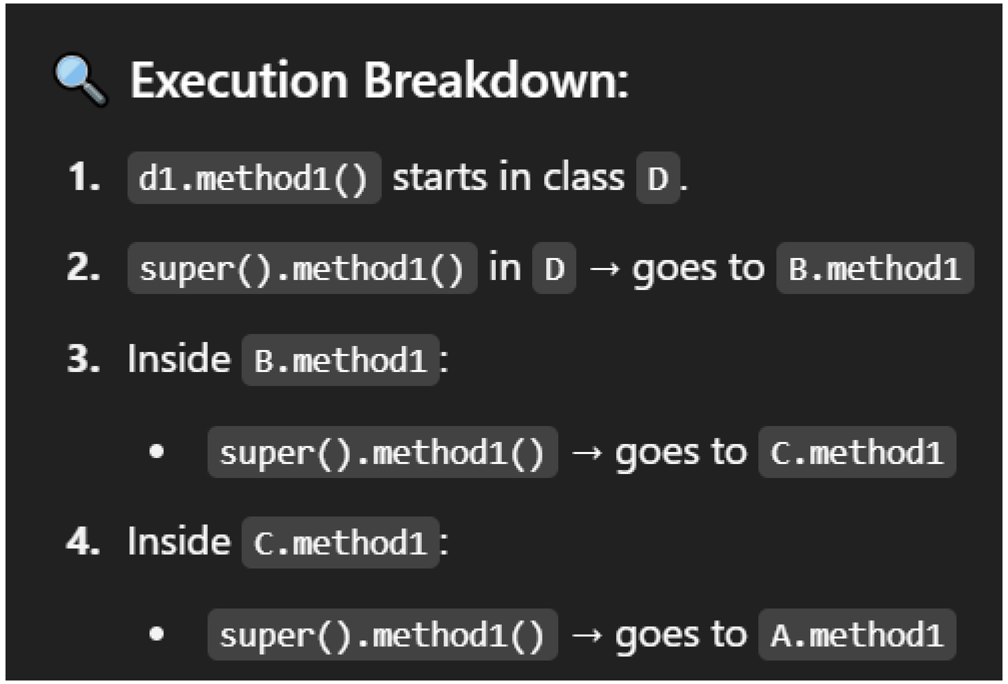
- After checking on all function versions in the Multiple Parent classes, The MRO orders super function to go to their shared Parent Class and executes it first then go executing backward the way in the same order but reversed.

Output illustration

- We might be confused why the super function in class B version of (method1) didn’t trigger and go up to the class A version of (method1) before the version in class C !!?

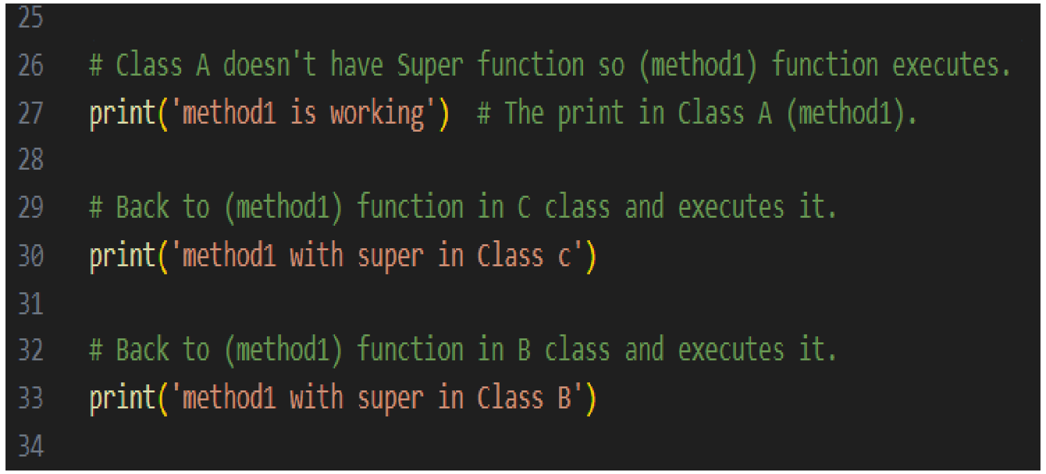
- That’s because how MRO works as it orders the super functions in a unique way so Multiple Parent classes comes before the up-level Parent classes but The first parent executes first then executes backward.
- The Multiple Parent classes are ordered as they come in the Child class definition.

Output illustration

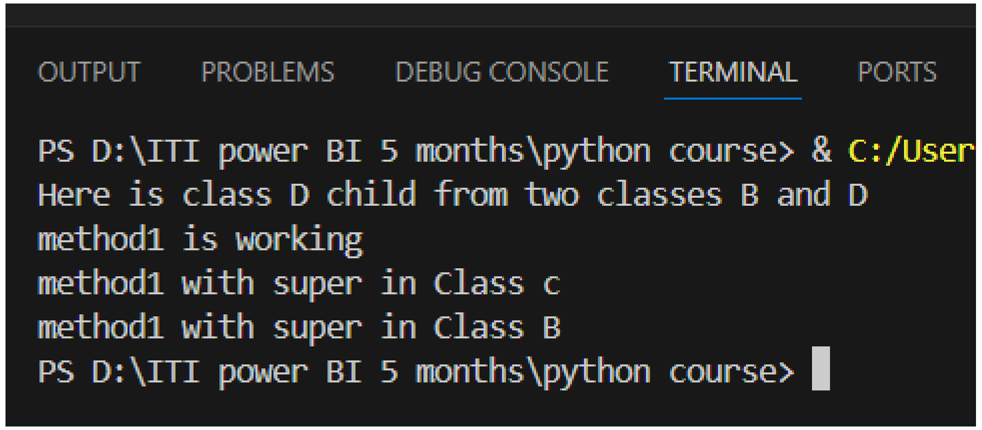


🔍 **Execution Breakdown:**

1. `d1.method1()` starts in class `D`.
2. `super().method1()` in `D` → goes to `B.method1`
3. Inside `B.method1`:
   - `super().method1()` → goes to `C.method1`
4. Inside `C.method1`:
   - `super().method1()` → goes to `A.method1`



```python

# Class A doesn't have Super function so (method1) function executes.
print('method1 is working')  # The print in Class A (method1).

# Back to (method1) function in C class and executes it.
print('method1 with super in Class c')

# Back to (method1) function in B class and executes it.
print('method1 with super in Class B')

```



```
PS D:\ITI power BI 5 months\python course> & C:/User
Here is class D child from two classes B and D
method1 is working
method1 with super in Class c
method1 with super in Class B
PS D:\ITI power BI 5 months\python course>
```