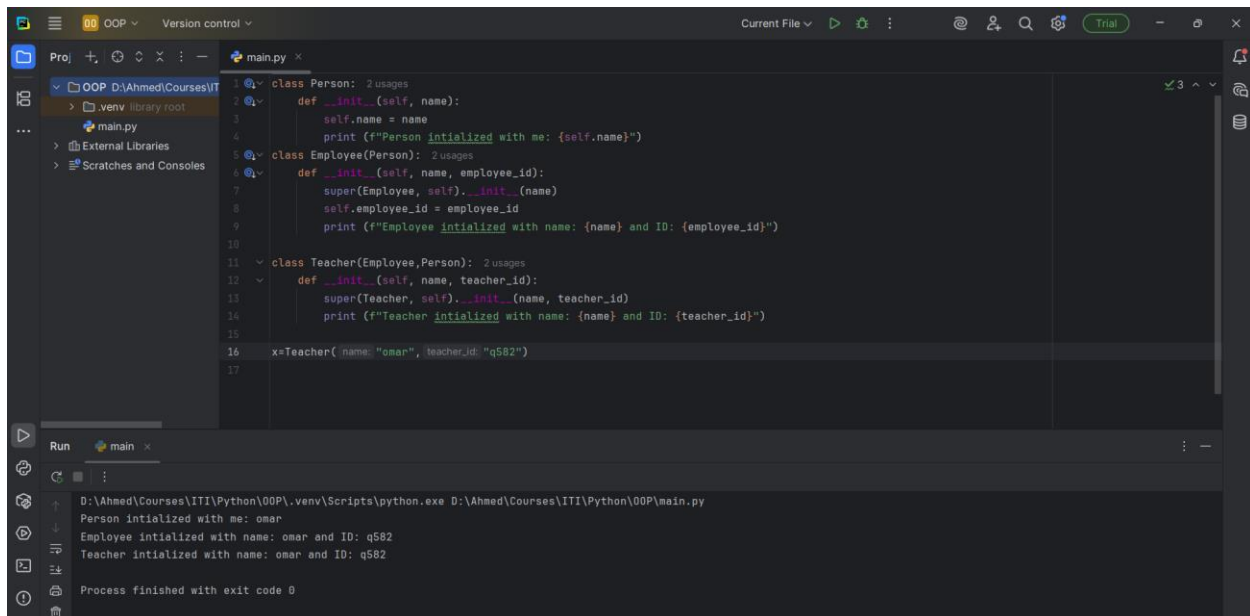


How “Super” Function handles multiple classes ?

Super function in multiple classes follows the MRO (method resolution order) which is a specific order that python decides to see which classes was inherited from in the right order as shown in the example here :

- The teacher class was inherited from the employee class which is inherited from the person class in order



The screenshot shows a Python IDE with a file named `main.py`. The code defines three classes: `Person`, `Employee`, and `Teacher`. `Person` is the base class. `Employee` inherits from `Person`. `Teacher` inherits from both `Employee` and `Person`. The `__init__` method in `Teacher` calls `super(Teacher, self).__init__(name, teacher_id)`, which then calls `super(Employee, self).__init__(name)`, which finally calls `super(Person, self).__init__(name)`. The output of the program shows the initialization order: `Person initialized with me: omar`, `Employee initialized with name: omar and ID: q582`, and `Teacher initialized with name: omar and ID: q582`. The process finished with exit code 0.

```
1 class Person: 2 usages
2     def __init__(self, name):
3         self.name = name
4         print (f"Person initialized with me: {self.name}")
5 class Employee(Person): 2 usages
6     def __init__(self, name, employee_id):
7         super(Employee, self).__init__(name)
8         self.employee_id = employee_id
9         print (f"Employee initialized with name: {name} and ID: {employee_id}")
10
11 class Teacher(Employee, Person): 2 usages
12     def __init__(self, name, teacher_id):
13         super(Teacher, self).__init__(name, teacher_id)
14         print (f"Teacher initialized with name: {name} and ID: {teacher_id}")
15
16 x=Teacher( name: "omar", teacher_id: "q582")
17
```

Run main x

D:\Ahmed\Courses\ITI\Python\OOP\venv\Scripts\python.exe D:\Ahmed\Courses\ITI\Python\OOP\main.py

Person initialized with me: omar

Employee initialized with name: omar and ID: q582

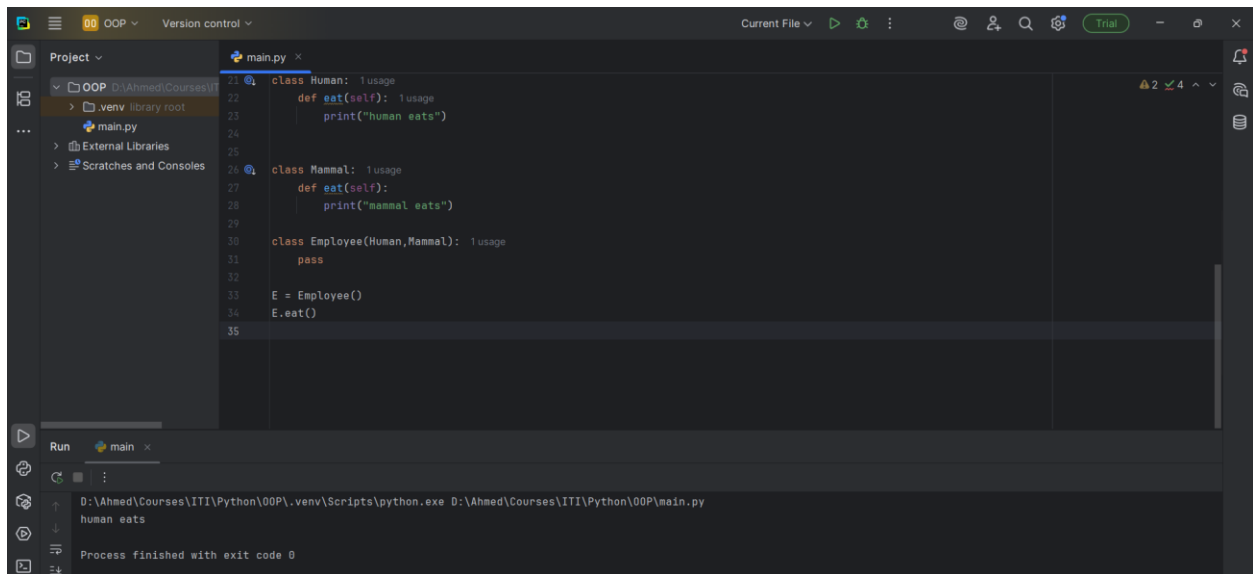
Teacher initialized with name: omar and ID: q582

Process finished with exit code 0

If Human and Mammal Have the same method like eat but with different Implementation.
When

Child[Employee] calls eat method how python handle this case.

In this case over here as well python follows the MRO so the child class Employee when it calls the method eat() it will see the methods eats in order and call the first one it reads as shown it printed "Human Eats"



```
21 class Human: 1usage
22     def eat(self): 1usage
23         print("human eats")
24
25
26 class Mammal: 1usage
27     def eat(self):
28         print("mammal eats")
29
30 class Employee(Human, Mammal): 1usage
31     pass
32
33 E = Employee()
34 E.eat()
35
```

Run main x

D:\Ahmed\Courses\ITI\Python\OOP\venv\Scripts\python.exe D:\Ahmed\Courses\ITI\Python\OOP\main.py
human eats

Process finished with exit code 0