فى الدرس دا هنتكلم عن ال **Multiple inheritance** و **Method override**.

اول حاجه ال method override. كنا اتكلمنا فى الدرس اللى فات عن حوار ال override لل attribute والدرس دا عن ال method وبكل بساطه علشان اعمل override ل method على ال method بتاع ال class اللى وارث منه بكتب اسم الفانكشن زي م هي وبخليها تنفذ اللى انا عايزه وهي هتمسح اللى ورثته وتستخدم الجديد.

دلوقتي مبدأ عمل ال Multiple inheritance هو اني بخلي class يورث اكتر من class مش واحد بس فبيورث ال methods بتاع الاتنين ويقدر يورث اي Attribute انا احددها من الاتنين.

بكل بساطه علشان class يورث مثلا من اتنين class تانيين هحط ال classes اللى عايز اورث منها بين اقواس ال class اللى هيورث، يعني كدا MyClass3(MyClass1,MyClass2)، كدا MyClass3 بيورث MyClass1 و MyClass2.

طيب ال class دلوقتي ورث اتنين فانكشن __init__ هيشغل انهي واحده لما اجي اعمل منه object ؟ هيستخدم اللى محطوطه فى الترتيب الاول بين الاقواس بتاعته، يعني انا حاطط MyClass1 الاول فى الاقواس فكدا اللى هيشتغل ال __init__ بتاع MyClass1.

علشان اعرف ايه ترتيب ال classes اللى بيورث منها class معين ممكن استخدم فانكشن mro() يعنى كدا **print(MyClass3.mro())**، كدا هيطبعلى ترتيب ال classes.

طريقه تانيه بيتم فيها مبدأ ال Multiple inheritance وهي ان لما class تاني يورث class اول وبعد كدا يجي class تالت يورث التاني فبكدا هيورث الأول والتاني، يعني لو class2(class1) و class3(class2) كدا class3 ورث class1 و class2.

بستخدم **pass** جوا ال class علشان انشأه من غير م اعمله اى initialization، يعني بنشأ class فاضي وعلشان ميدنيش error بكتبله pass.

```
# ------------------------------------------------------------
# -- Object Oriented Programming => Multiple Inheritance --
# ------------------------------------------------------------

class BaseOne:

    def __init__(self):

        print("Base One")

    def func_one(self):

        print("One")

class BaseTwo:

    def __init__(self):

        print("Base Two")

    def func_two(self):

        print("Two")
```

```python
class Derived(BaseOne, BaseTwo):

    pass

my_var = Derived()

# print(Derived.mro())

print(my_var.func_one)
print(my_var.func_two)

my_var.func_one()
my_var.func_two()

class Base:

    pass

class DerivedOne(Base):

    pass

class DerivedTwo(DerivedOne):

    pass
```

CODE

```
Base One
[<class '__main__.Derived'>, <class '__main__.BaseOne'>, <class '__main__.BaseTwo'>, <class 'object'>]
<bound method BaseOne.func_one of <__main__.Derived object at 0x000001C175FF5FD0>>
<bound method BaseTwo.func_two of <__main__.Derived object at 0x000001C175FF5FD0>>
One
Two
```

OUTPUT