Multiple Inheritance

وراثت چندگانه در پایتون

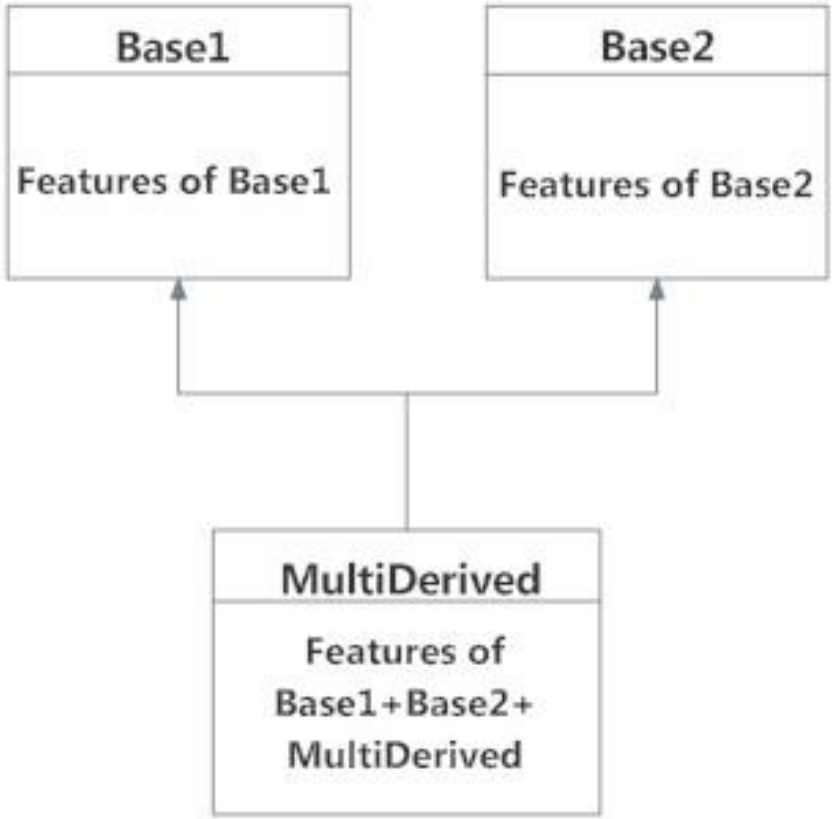
- یک کلاس در پایتون می تواند از بیش از یک کلاس base مشتق شود. به این مقوله **وراثت چندگانه** گفته می شود.
- در وراثت چندگانه، کلاس فرزند همه ویژگی های کلاس های پایه را به ارث می برد.

ساختار وراثت چندگانه در پایتون

```
class Base1:
    pass

class Base2:
    pass

class MultiDerived(Base1, Base2):
    pass
```



وراثت چندسطحی (multilevel) در پایتون

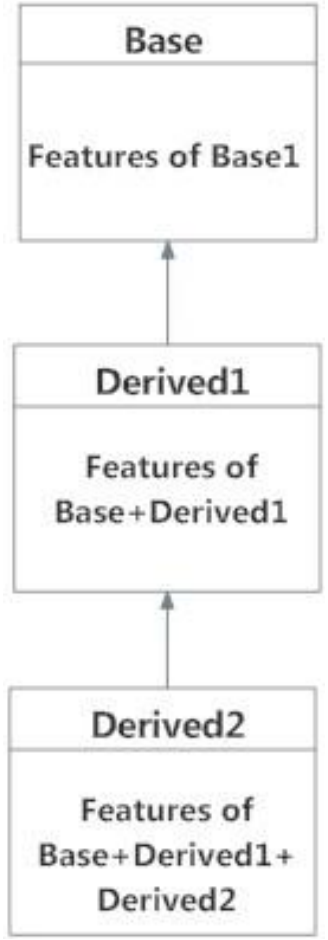
- یک کلاس می تواند از یک کلاس مشتق شده دیگر در پایتون ارث بری کند. به این ساختار **وراثت چندسطحی** گفته می شود.
- در وراثت چندسطحی، ویژگی های کلاس پایه و مشتق شده توسط کلاس جدید به ارث برده می شود.

ساختار وراثت چندگانه در پایتون

```
class Base:
    pass

class Derived1(Base):
    pass

class Derived2(Derived1):
    pass
```



Method Resolution Order

- هر کلاسی در پایتون از کلاس **object** مشتق می شود.(رایج ترین کلاس پایه در پایتون) در نتیجه همه کلاس های دیگر از کلاس های built-in گرفته تا کلاس هایی که توسط کاربران تعریف می شوند، کلاس مشتق شده محسوب شده و همه اشیا آنها نمونه ای از کلاس object محسوب می شوند.
- مثال

```
# Output: True
print(issubclass(list,object))

# Output: True
print(isinstance(5.5,object))

# Output: True
print(isinstance("Hello",object))
```

Method Resolution Order - ادامه

- در وراثت چندگانه، هر صفتی ابتدا در کلاس جاری جستجو می شود و سپس اگر نبود، در کلاس پدرش مورد جستجو قرار می گیرد.
- پس در مثال وراثت چندگانه، ترتیب بصورت MultiDerieved، Base1، Base2 و object می باشد.
- به مجموعه قواعدی که برای پیدا کردن این ترتیب مورد استفاده قرار می گیرند، MRO گفته می شود.
- برای دیدن MRO به دو صورت زیر می توان عمل کرد:

```
>>> MultiDerived.__mro__
(<class '__main__.MultiDerived'>,
 <class '__main__.Base1'>,
 <class '__main__.Base2'>,
 <class 'object'>)

>>> MultiDerived.mro()
[<class '__main__.MultiDerived'>,
 <class '__main__.Base1'>,
 <class '__main__.Base2'>,
 <class 'object'>]
```

مثال - Method Resolution Order

```
class X: pass
class Y: pass
class Z: pass

class A(X,Y): pass
class B(Y,Z): pass

class M(B,A,Z): pass

# Output:
# [<class '__main__.M'>, <class '__main__.B'>,
# <class '__main__.A'>, <class '__main__.X'>,
# <class '__main__.Y'>, <class '__main__.Z'>,
# <class 'object'>]

print(M.mro())
```