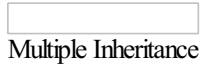


Python Multiple Inheritance

A [class](#) can be derived from more than one superclass in Python. This is called multiple [inheritance](#).

For example, a class `Bat` is derived from superclasses `Mammal` and `WingedAnimal`. It makes sense because bat is a mammal as well as a winged animal.



Python Multiple Inheritance Syntax

```
class SuperClass1:
    # features of SuperClass1

class SuperClass2:
    # features of SuperClass2

class MultiDerived(SuperClass1, SuperClass2):
    # features of SuperClass1 + SuperClass2 + MultiDerived class
```

Here, the `MultiDerived` class is derived from `SuperClass1` and `SuperClass2` classes.

Example: Python Multiple Inheritance

```
class Mammal:
    def mammal_info(self):
        print("Mammals can give direct birth.")

class WingedAnimal:
    def winged_animal_info(self):
        print("Winged animals can flap.")

class Bat(Mammal, WingedAnimal):
    pass

# create an object of Bat class
b1 = Bat()

b1.mammal_info()
b1.winged_animal_info()
```

Output

```
Mammals can give direct birth.
Winged animals can flap.
```

In the above example, the `Bat` class is derived from two super classes: *Mammal* and *WingedAnimal*. Notice the statements,

```
b1 = Bat()
b1.mammal_info()
b1.winged_animal_info()
```

Here, we are using `b1` (object of *Bat*) to access `mammal_info()` and `winged_animal_info()` methods of the *Mammal* and the *WingedAnimal* class respectively.

Python Multilevel Inheritance

In Python, not only can we derive a class from the superclass but you can also derive a class from the derived class. This form of inheritance is known as **multilevel inheritance**.

Here's the syntax of the multilevel inheritance,

```
class SuperClass:
    # Super class code here

class DerivedClass1(SuperClass):
    # Derived class 1 code here

class DerivedClass2(DerivedClass1):
    # Derived class 2 code here
```

Here, the *DerivedClass1* class is derived from the *SuperClass* class, and the *DerivedClass2* class is derived from the *DerivedClass1* class.



Example: Python Multilevel Inheritance

```
class SuperClass:

    def super_method(self):
        print("Super Class method called")

# define class that derive from SuperClass
class DerivedClass1(SuperClass):
    def derived1_method(self):
        print("Derived class 1 method called")

# define class that derive from DerivedClass1
class DerivedClass2(DerivedClass1):

    def derived2_method(self):
        print("Derived class 2 method called")

# create an object of DerivedClass2
d2 = DerivedClass2()

d2.super_method() # Output: "Super Class method called"

d2.derived1_method() # Output: "Derived class 1 method called"

d2.derived2_method() # Output: "Derived class 2 method called"
```

Output

```
Super Class method called
Derived class 1 method called
Derived class 2 method called
```

In the above example, *DerivedClass2* is derived from *DerivedClass1*, which is derived from *SuperClass*.

It means that *DerivedClass2* inherits all the attributes and methods of both *DerivedClass1* and *SuperClass*.

Hence, we are using *d2* (object of *DerivedClass2*) to call methods from *SuperClass*, *DerivedClass1*, and *DerivedClass2*.

Method Resolution Order (MRO) in Python

If two superclasses have the same method ([function](#)) name and the derived class calls that method, Python uses the MRO to search for the right method to call. For example,

```
class SuperClass1:
    def info(self):
        print("Super Class 1 method called")

class SuperClass2:
    def info(self):
```

```
print("Super Class 2 method called")

class Derived(SuperClass1, SuperClass2):
    pass

d1 = Derived()
d1.info()

# Output: "Super Class 1 method called"
```

Here, *SuperClass1* and *SuperClass2* both of these classes define a method `info()`.

So when `info()` is called using the *d1* object of the *Derived* class, Python uses the **MRO** to determine which method to call.

In this case, the **MRO** specifies that methods should be inherited from the leftmost superclass first, so `info()` of *SuperClass1* is called rather than that of *SuperClass2*.

Also Read:

- [Python Object Oriented Programming](#)
- [Polymorphism in Python](#)
- [self in Python, Demystified](#)

Table of Contents

- [Introduction](#)
- [Python Multiple Inheritance Syntax](#)
- [Example: Python Multiple Inheritance](#)
- [Python Multilevel Inheritance](#)
- [Example: Python Multilevel Inheritance](#)
- [Method Resolution Order \(MRO\) in Python](#)