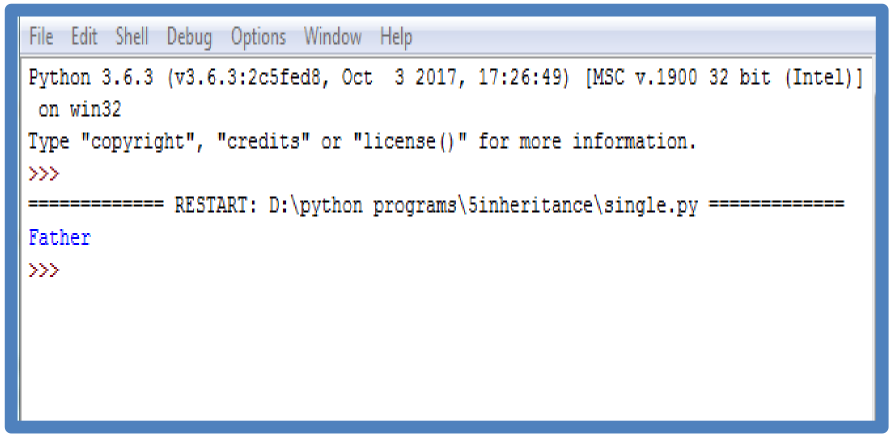
# Single Inheritance

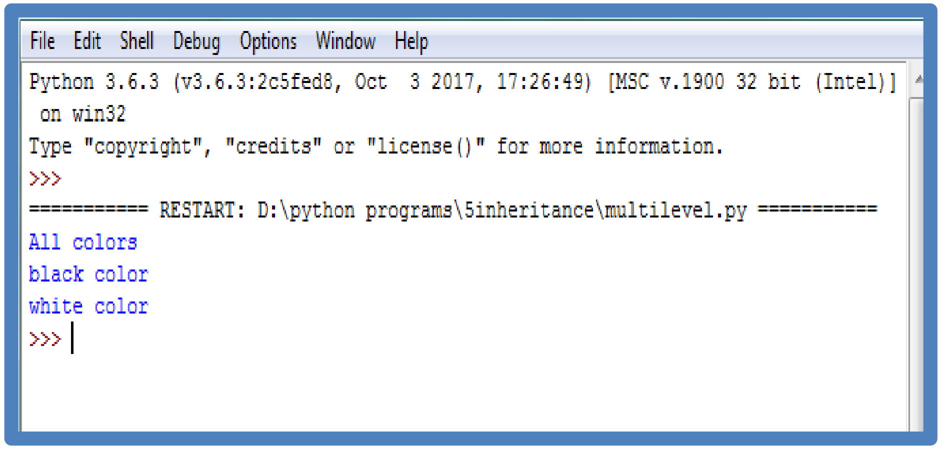
```
class father:
    def f2(self):
        print("Father")

class child(father):
    def f3(self):
        print("child")
        pass

obj=child()
obj.f2()
```



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: D:\python programs\5inheritance\single.py =============
Father
>>>
```

# Multilevel Inheritance

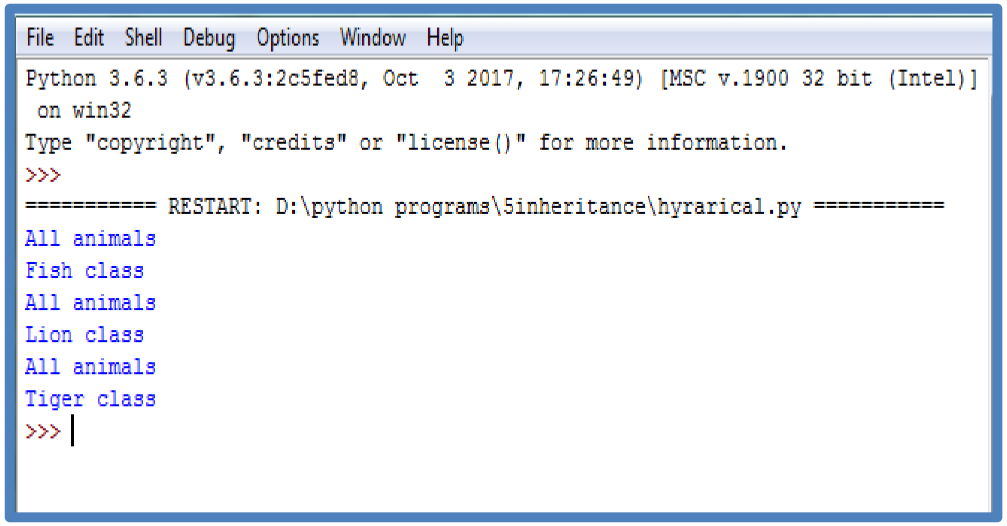
```
class color:
    def f1(self):
        print("All colors")
class black(color):
    def f2(self):
        print("black color")
class white(black):
    def f3(self):
        print("white color")
obj=white()
obj.f1()
obj.f2()
obj.f3()
```



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: D:\python programs\5inheritance\multilevel.py ===========
All colors
black color
white color
>>>
```

# Hyrarical Inheritance

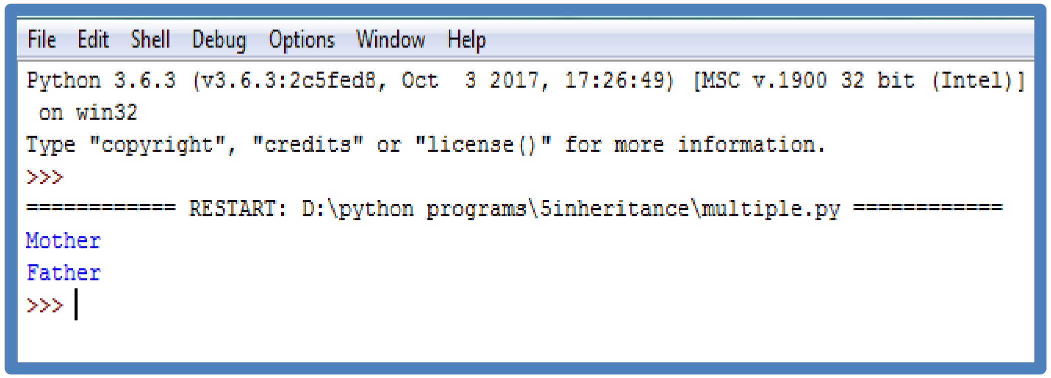
```
class animal:
    def f1(self):
        print("All animals")
class fish(animal):
    def f2(self):
        print("Fish class")
class lion(animal):
    def f3(self):
        print("Lion class")
class tiger(animal):
    def f4(self):
        print("Tiger class")
obj1=fish()
obj2=lion()
obj3=tiger()
obj1.f1()
obj1.f2()
obj2.f1()
obj2.f3()
obj3.f1()
obj3.f4()
```



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: D:\python programs\5inheritance\hyrarical.py ===========
All animals
Fish class
All animals
Lion class
All animals
Tiger class
>>>
```

# Multiple Inheritance

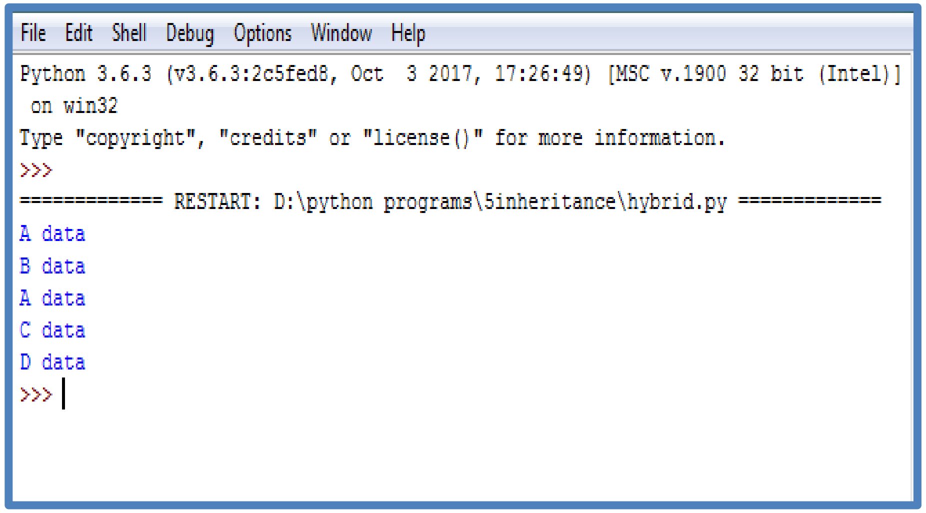
```
class mother:
    def f1(self):
        print("Mother")

class father:
    def f2(self):
        print("Father")

class child(mother,father):
    def f3(self):
        pass

obj=child()
obj.f1()
obj.f2()
obj.f3()
```



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: D:\python programs\5inheritance\multiple.py ============
Mother
Father
>>> |
```

## Hybrid Inheritance

```
class a:
    def f1(self):
        print("A data")
class b(a):
    def f2(self):
         a.f1(self)
         print("B data")
class c(a):
    def f3(self):
         a.f1(self)
         print("C data")
 class d(b,c):
    def f4(self):
        b.f2(self)
        c.f3(self)
        print("D data")
obj=d()
obj.f4()
```



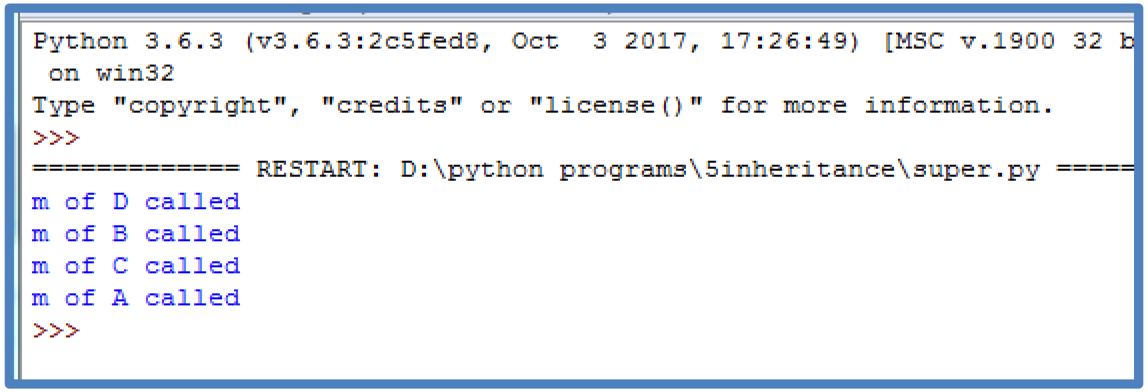
```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: D:\python programs\5inheritance\hybrid.py =============
A data
B data
A data
C data
D data
>>> |
```

## Use of super()

**Super.** With the super() built-in, we can get the parent of a class.In hybrid inheritance, super class calls multiple time,to remove this, we can use super function.

# Use of super()

```
class A:
    def m(self):
        print("m of A called")
class B(A):
    def m(self):
        print("m of B called")
        super().m()
 class C(A):
    def m(self):
        print("m of C called")
        super().m()
class D(B,C):
    def m(self):
        print("m of D called")
       super().m()
x=D()
x.m()
```



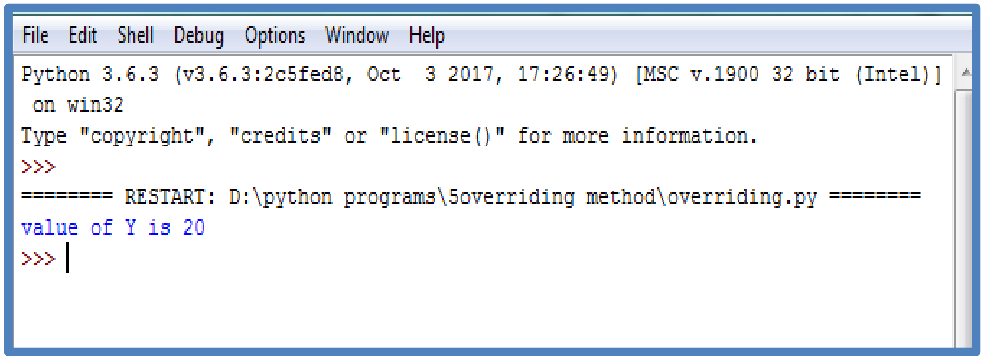
```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 b
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: D:\python programs\5inheritance\super.py =====
m of D called
m of B called
m of C called
m of A called
>>>
```

## Method overriding

Override means having two methods with the same name but doing different tasks. It means that one of the methods overrides the other.

If there is any method in the superclass and a method with the same name in a subclass, then by executing the method, the method of the corresponding class will be executed.

```
class a:
    def f1(self):
        self.x=10
        print("value of X is",self.x)
class b(a):
    def f1(self):
        self.y=20
        print("value of Y is",self.y)
obj=b()
obj.f1()
```



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: D:\python programs\5overriding method\overriding.py ========
value of Y is 20
>>> |
```