**Types of Inheritance in Python and Examples of PythonClass Inheritance**

There are two kinds of inheritance in Python - multiple and multilevel inheritance.

**Multiple Inheritance in Python**

# Python example to show working of multiple inheritance

class grand\_parent:

def fun1(self):

print("Fun1 working..")

class parent:

def fun2(self):

print("fun2 working..")

class child(grand\_parent, parent):

def fun3(self):

print("Child Class Function")

ob = child()

ob.fun1()

ob.fun2()

ob.fun3()

**Output**

Fun1 working..

fun2 working..

Child Class Function

**Multilevel Inheritance in Python**

class **Human**:

def **speak**(self):

print("Human Speaking")

#The child class Parent inherits the base class Human

class Parent(Human):

def eat(self):

print("Eat Food")

#The child class child inherits another child class Parent

class child(Parent):

def drink(self):

print("Drink Milk")

d = child()

d.eat()

d.speak()

d.drink()

**Output:**

Eat Food

Human Speaking

Drink Milk

Multiple Inheritance refers to the mechanism when the properties of multiple classes are inherited by a single child class. Say, there are two classes - A and B - and the programmer desires to create a new class that has the properties of both A and B, then:

The above lines depict how the characteristics of both classes A and B are inherited in the same child class C with the help of the multiple inheritance mechanism. As is visible above, instead of mentioning just one class name within parentheses for defining the child class, two different class names, separated by a comma, have been mentioned to do the needful. As the properties of any given number of classes can be inherited by the child class, the syntax can be written as:

In case of multilevel inheritance, the classes will be inherited at multiple separate levels. Say, there are three classes named A, B and C - A is the super-class, B the **sub(child)** class, and C is referred to as the sub class of B.

***Let’s refer to a simple example to explain the mechanism of multilevel inheritance in Python:***

**Overriding Methods in Python**

**Example Overriding Methods in Python**

class **Rtangle**():

def \_\_init\_\_(self,length,breadth):

self.length = length

self.breadth = breadth

def **getArea**(self):

print self.length\*self.breadth," The area of rectangle"

class **Square**(**Rtangle**):

def \_\_init\_\_(self,side):

self.side = side

Rtangle.\_\_init\_\_(self,side,side)

def **getArea**(self):

print self.side\*self.side," The area of square"

s = Square(**5**)

r = Rtangle(**3**,**5**)

s.getArea()

r.getArea()

Output

**25** The area of square

**15** The area of rectangle

**Conclusion**

Inheritance in python is very useful mechanisms that help programmers attain the best results for their codes. Inheritance allows coders to create any general class and then extend the same to specialized class. Inheritance also sets the base for writing better codes. Inheritance helps child classes access all data fields as well as the methods / functions of the parent class/ classes. In addition, more fields and methods can be added without having to write the already written codes from the scratch -thereby eliminating the concerns of code duplication