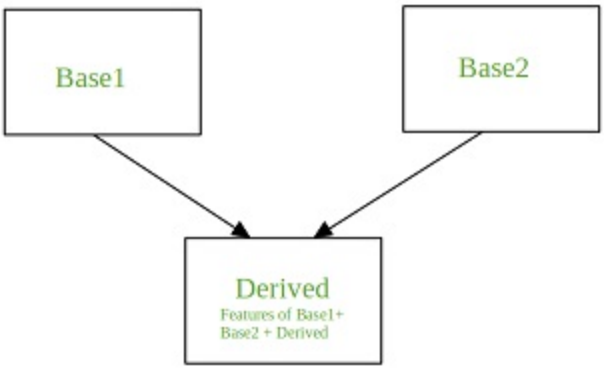
OOPs - Multiple vs Multilevel Inheritence In Python

*By Sudhanshi

Multiple Inheritence

- When you inherit a child class from more than one base classes that situation is known as Multiple Inheritance.
- The child class claims the properties and methods of all the parent classes.
- Multiple Inheritence has two class levels base class and derived class.
- It is not commonly used because it makes our system more complex



In [1]:
```
class A:
    def func1(self):
        return "I am Sudhanshi"
class B:
    def func2(self):
        return "I Like coding"
class C(A, B):
    pass
s1 = C()
print(s1.func1(), "and", s1.func2())
```

```
I am Sudhanshi and I Like coding
```

In [31]:
```
class Father:
    def F_likes(self):
        print('Father enjoys gardening and programming')
class Mother:
    def M_likes(self):
        print('Mother Loves cooking and travelling')
class child(Father,Mother):
    def likes(self):
        Father.F_likes(self)
        Mother.M_likes(self)
        print('I like programming and travelling')
```

In [3]:
```
cl= child()
cl.likes()
```

```
Father enjoys gardening and programming
Mother Loves cooking and travelling
I like programming and travelling
```

In [4]:
```
cl.M_likes()
```

```
Mother Loves cooking and travelling
```

In [5]:
```
class Addition:
    num1= 200
    num2= 300
    def add(self):
        sum=self.num1+self.num2
        print("Addition of num1 and num2 is:",sum)

class Subtraction:
    num3 = 50
    def sub(self):
        sub = self.num2 - self.num3
        print("Subtraction of num2 and num3 is :",sub)

class Output(Addition,Subtraction):
    num4 = 5
    def multiply(self):
        multiply = self.num1 + self.num2 - self.num3 * self.num4
        print("The final output is:",multiply)
```

In [6]:
```
operation = Output()
operation.add()
```

```
Addition of num1 and num2 is: 500
```

In [7]:
```
operation.sub()
```

```
Subtraction of num2 and num3 is : 250
```

In [8]:
```
operation.multiply()         #200 + 300 - 50 * 5
                                #200 +300 - 250
                                 #500-250
                                   #250
```

```
The final output is: 250
```

In [9]:
```
class Person:
    #defining constructor
    def __init__(self, name, age,id):
        self.name = name
        self.age = age
        self.id = id

    #defining class methods
    def showName(self):
        print(self.name)

    def showAge(self):
        print(self.age)

    def showId(self):
        print(self.id)

    #end of class definition
```

```
# defining another class
class Employee: # Person is the
    def __init__(self, Empskills):
        self.Empskills =  Empskills

    def getskills(self):
        return self.Empskills


class Details(Person, Employee): # extends both Person and Employee class
    def __init__(self, name, age, id,Empskills,salary):
        self.salary = salary
        Person.__init__(self, name, age, id)
        Employee.__init__(self, Empskills)

    def getsalary(self):
        return self.salary
```

In [10]:
```
# Create an obj of the subclass

Empdetails = Details('Arundhati',32,1011237,'Python',37000)
```

In [11]:
```
Empdetails.getskills()
```

Out[11]: 'Python'

In [12]:
```
Empdetails.getsalary()
```

Out[12]: 37000

In [13]:
```
Empdetails.showName()
```

Arundhati

In [14]:
```
Empdetails.showAge()
```

32

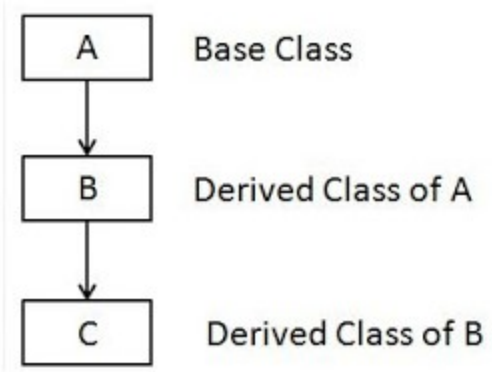
In [15]:
```
Empdetails.showId()
```

1011237

Multilevel Inheritence Class

- When you inherit a class from a derived class, then it’s called multilevel inheritance.
- This class has three levels base class,intermediate class and derived class.
- Multilevel class is widely used because in comparison of multiple class this is less complex

Multilevel inheritance in Python



In [16]:
```
class Father:
    def showF(self):
        print('Father Class Method')

class Son(Father):
    def showS(self):
        print('Son Class Method')

class Grandson(Son):
    def showGs(self):
        print('Grandson Class Method')
```

In [17]:
```
obj = Grandson()
```

In [18]:
```
obj.showF()
obj.showGs()
obj.showS()
```

```
Father Class Method
Grandson Class Method
Son Class Method
```

The Super keyword returns temporary object of the superclass,that allow us to access the methods of the superclass.

In [19]:
```
class Father:                          #Superclass1
    def __init__(self):
        super().__init__()         #Calling parent class constructor
        print('Father class constructor')
    def showF(self):
        print('Father class Method')

class Mother:                        #Superclass2
    def __init__(self):
        super().__init__()
        print('Mother class constructor')
    def showF(self):
        print('Mother class Method')

class Son(Father,Mother):    #Subclass
```

```
    def __init__(self):
        super().__init__()
        print('Son class constructor')
    def showF(self):
        print('Son class Method')
```

In [20]:

```
s=Son()
```

```
Mother class constructor
Father class constructor
Son class constructor
```

In [21]:

```
# Base class
class Grandfather:

    def __init__(self, grandfathername):
        self.grandfathername = grandfathername

# Intermediate class
class Father(Grandfather):
    def __init__(self, fathername, grandfathername):
        self.fathername = fathername

# Derived class
class Son(Father):
    def __init__(self,sonname, fathername, grandfathername):
        self.sonname = sonname
        Father.__init__(self, fathername, grandfathername)
        Grandfather.__init__(self, grandfathername)
    def print_name(self):
        print('Grandfather name :', self.grandfathername)
        print("Father name :", self.fathername)
        print("Son name :", self.sonname)

#  Driver code
s1 = Son('Mahesh', 'Suresh', 'Ramesh')
print(s1.grandfathername)
s1.print_name()
```

```
Ramesh
Grandfather name : Ramesh
Father name : Suresh
Son name : Mahesh
```

In [22]:

```
print(s1.fathername)
```

```
Suresh
```

In [23]:

```
print(s1.grandfathername)
```

```
Ramesh
```

In [30]:

```
class Person:

    def __init__(self, name,age):
        self.name = name
        self.age=age

# Intermediate class
class Employee(Person):
    def __init__(self, Id, salary):
        = Id
```

```
        self.salary=salary

 #Derived class
class details(Employee):
    def __init__(self,name,age,Id,salary,experience,dependents):
        self.experience=experience
        self.dependents=dependents

        # invoking constructor of Father class
        Employee.__init__(self,Id,salary)
        Person.__init__(self, name,age)

    def print_details(self):
        return(' Name :', self.name)
        return("Age :", self.age)
        return("Emp_ID :", self.id)
        return('Salary:',self.salary)
        return('Experience:',self.experience)
        return('Dependents:',self.dependents)
```

In [25]:
```
Emp = details('Prince', 42, 1100023,620000,15,'No')
```

In [26]:
```
Emp.dependents
```

Out[26]:
```
'No'
```

In [27]:
```
Emp.name
```

Out[27]:
```
'Prince'
```

In [ ]:
```

```