In [5]:
```python
# PYTHON NOTES 12-12-24

#Class and Object in python

class Myclass:
    x=0 # By default it is public
    _y=0 # Protected variable in python
    __z=0 # Private variable
    def __init__(self, a, b, c): # Parameterized Constructor
        self.x=a
        self.y=b
        self.z=c
    def display(self): #First parameter should be self. Acts likes "this" in Java.
        print(f"Value of x: {self.x}, Value of y: {self.y}, Value of z: {self.z}")


if __name__=="__main__":
    obj=Myclass(10, 20, 30) # Calls the default constructor of that class.
    obj.display()

# Public access specifier - Makes field visible anywhere in the program
# Private cannot be accessed from outside the class.
# Does not partitcipate in inheritance.
# Protected acts like a dat amember of its own class. Participates in inheritance.
```

```
Value of x: 10, Value of y: 20, Value of z: 30
```

In [9]:
```python
# Inheritance in Python

class Base:
    x=0
    _y=0
    def __init__(self, a, b):
        self.x=a
        self.y=b
    def display(self):
        print(f"This is inside the Base class.")
        print(self.x, self.y)

class Child(Base):
    z=0
    def __init__(self, a, b, c):
        Base.__init__(self, a, b) #super().__init__(a,b)
        self.z=c
    def display(self):
        print(f"This is inside the Child class.")
        super().display()
# This is how we can call the base class display method from the child class.
        print(self.z)

if __name__=="__main__":
    obj1=Base(10, 20)
    obj2=Child(15, 30, 45)
    obj1.display()
    obj2.display()
# If your Base class consists of a constructor then
```

```
# it is mandatory to define a constructor in the subsequent child classes.
# The Super keyword helps us to call the base class constructor.
```

```
This is inside the Base class.
10 20
This is inside the Child class.
This is inside the Base class.
15 30
45
```

In [8]:

```
# Multiple Inheritance in Python

class Base1:
    def __init__(self, a,):
        self.name=a
    def display1(self):
        print(self.name)

class Base2:
    def __init__(self, b):
        self.roll=b
    def display2(self):
        print(self.roll)

class Child(Base1, Base2):
    def __init__(self, a, b, c):
        Base1.__init__(self, a)
        Base2.__init__(self, b)
        self.marks=c
    def display(self):
        print(self.name, self.roll, self.marks)

if __name__=="__main__":
    ob1=Base1("Sayan")
    ob2=Base2(10)
    ob3=Child("Sourav", 20, 80)
    ob1.display1()
    ob2.display2()
    ob3.display()
```

```
Sayan
10
Sourav 20 80
```