Static method in both parent class and immediate parent.

```
class A:
    a=10
    def __init__(self, x):
        self.x=x
        print('init in parent class')
    @staticmethod
    def disp(x):
        print('Static method disp in parent class')

 class B(A):
     c=30
     def __init__(self, x):
         print('init in immediate parent')

     @staticmethod
     def disp(~~cls~~, x):
         print('Static method disp in immediat class')

 class C(B):
     e=50
     def __init__(self, x):
         print('init in child class')

  ob=B(3)
  oc=C(4)
  A.disp(3)
  ob.disp(2)
  oc.disp(3)
```

o/p:
```
init in immediate parent
init in child class
Static method disp in parent class
Static method disp in immediat class
Static method disp in immediat class
Static method disp in immediate class
```

Static method in all the class [Parent, immediate parent, mi

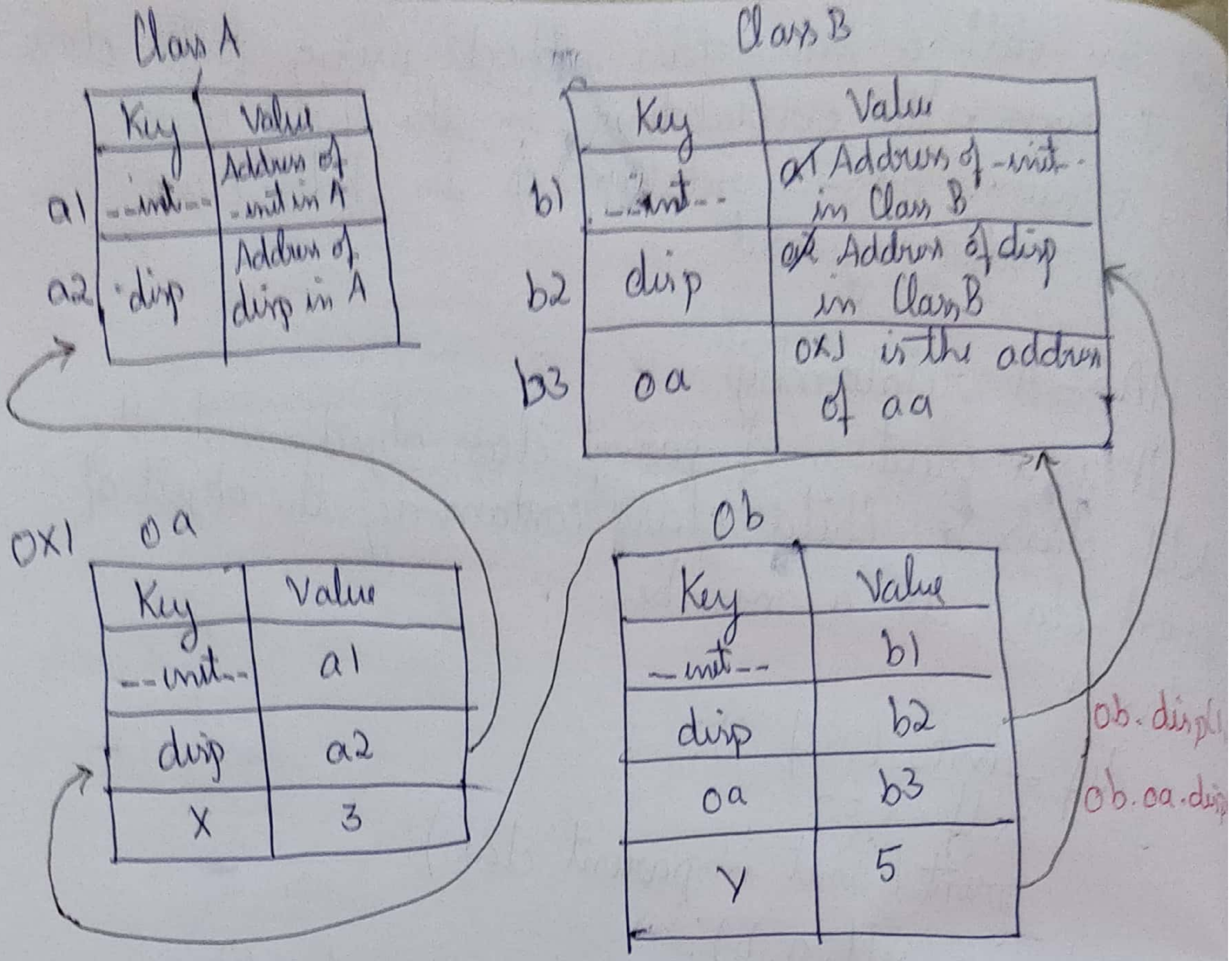
```
class A:
    a=10
    def __init__(self,x):
        self.x=x
        print('init in parent class')
    @staticmethod
    def disp(x):
         print('~~init~~ Static class in ~~immediat~~ class parent')

class B(A):
    c=30
    def __init__(self,x):
        print('init in immediate parent')
    @staticmethod
    def disp(x):
        print('Static method disp in immediate class')

class C(B):
    c=50
    def disp(x):
        print('Static method class in child method')
~~#~~ ob = B(3)
    oc = C(4)
    A.disp(3)
    ob.disp(2)
    oc.disp(3)
```

O/P
```
init in immediate parent
init in immediate parent
Static method disp in parent class
Static method disp in immediate class
Static method disp in child class
```

We can call a base class method using child class object even after overridding in the child class.
① By creating 'has-a' relationship in child class
② Using super keyword

① HAS-A relationship
It is a creation of parent class object inside the child class ie Child class containing the object of parent class as a member.

```
class A:
    def __init__(self,x):
        self.x=x
        print('init in parent class')
    def disp(self,a,b):
        print('Object method disp in class A')
        return a+b
class B(A):
    oa=A(3)
    def __init__(self,y):
        self.y=y
        print('Init in immediate parent (child) class')
    def disp(self,a,b):
        print('Object method al in class B')
        return a-b
ob=B(5)
print(A.disp(ob,2,3))
print(B.disp(ob,4,5))
print(ob.disp(4,3))
```



SUPER KEYWORD

Syntax:

Super(ChildClassName, ObjectName).MethodName

where
child className ; the classname whose parent class member to be accessed.
Object Name: Object reference of child class.

```
Class A:
  def __init__(self,x):
      self.x=x
      print('init in parent class')
  @ class method
  def disp(cls,a,b):
      print('Class method a1 in class A')
      return a+b
```

```
class B(A):
    oa =A(3)
    def __init__(self, y):
        self.y=y
        print('init in immediate parent (child) class')
    @classmethod
    def disp(cls a, b):
        print('Class method disp in ClassB')

ob = B(5)
print (A.disp (2,3))  # invokes object method in classA
print (B.disp (3,4))                                classB
print (ob. disp (4,3))                              classB
print (ob. aa. disp (5,6))                          classA
print (super (B. ob). disp (3,4))
o/p:
    init in parent class
    init in immediate parent (child) class
    Class method a1 in classA
    5
    Class method a1 in classB
    -1
    Class method a1 in classB
    1
    Class method a1 in classA
    11
    Class method a1 in classA
    7
```

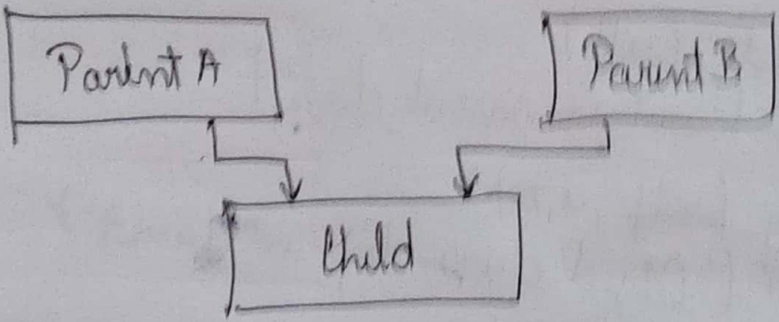
Scanned by CamScanner

Invoking the base class method using child class object even after overriding in multilevel inheritance

```
Class A:
    def _init_(self):
        print('init in parent class')
    def disp(self,a,b):
        print('Object class disp in classA')
        return a+b
Class B(A):
    oa=A(3)
    def --init(self ):
        print('init of immediate class')
    def disp(self,a
        print('Object class disp in class B')
Class C(B):
    ob=B(5)
    def --init(self):
        print('init of child class')
    def disp(self,c
        print('Object class disp in class C')
oc=C()
print(A.disp(2,3))
print(B.disp(5,6))
print(C.disp(10))
print(oc.ob.oa.disp(10,20))
print(oc.ob.disp(10,15))
```

MULTIPLE INHERITANCE

When new child class is created using more tha
one parent class.



SYNTAX

```
Class A:
    ===
Class B:
    ===
Class C(A,B) or Class(B,A)
```

~~When~~ Then order of inheritence is from right to left
ie C(A,B) is defined
then first class B will be inherited followed by
class A members.

Ex
```
class A:
    a=10
Class B:                     OP: 10 20 30
    b=20
class C(A,B)
    c=30
print (C.a, C.b, C.c)
class A:
    a=10
class B:
    a=20                     op: 10 30
Class C
    c=30
print (C.a, C.c)
```

Class A

	Key	Value
a1	a	10

Class B

	Key	Value
b1	b	20

C(A,B)

Key	Value
b	b1
a	a1
c	30

Same Variable

Class A

	Key	Value
a1	a	10

Class B

Key	Value
a	20

Key	Value
a	~~20~~ 10
c	30

```
class A:
    a=10
    def __init__(self):
        print('init in parent A')

class B:
    a=20
    def __init(self):
        print('init in parent B')
class C(B,A):
    c =30

print(C.a, C.c)
oc = C()
o/p: 20 30
    init in parent B
```

When parent constructor is defined in both the parent class, then the child class will contain references of lastly inherited parent class.

A

	Key	Value
a1	a	10
a2	__init__	Address of init in A

B

	Key	Value
b1	a	20
b2	__init ~~in~~ ~~address of B~~	Address of __init in B

Key	Value
a	~~a1~~ b1
_init	~~a2~~ b2
c	30

When the constructor is defined in both the parent class and the child class also has the constructor. Object of child class will have constructor of its parent class.

```
class A:
    a=10
    def __init__(self):
        print('init in Parent A')
class B:
    a=20
    def __init__(self):
        print('init in Parent B')
class C:
    c=30
    def __init__(self):
        print('init in Parent C')
print(C.a, C.c)
oc = C()
```

Class A

	Key	Value
a1	a	10
a2	__init__	init in A

Class B

	Key	Value
b1	a	20
b2	__init__	Address of __init__ in B

Class C

Key	Value
a	a1 b1
__init	Address of __init of C

Object C

Key	Value
a	~~a1~~ ~~b1~~ c1
__init__	~~a2~~ ~~b2~~ Address of __init of C
c	30