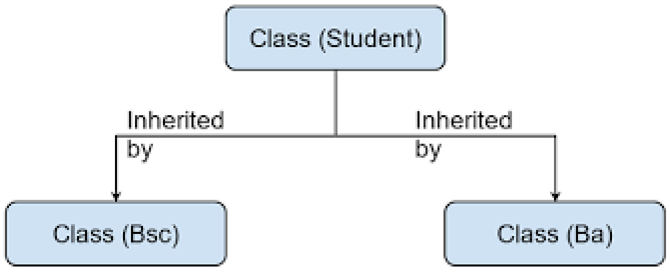
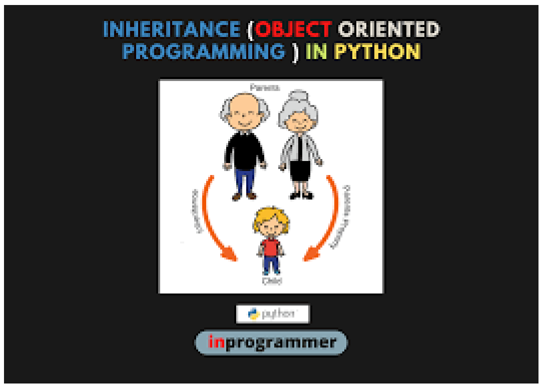
Inheritance

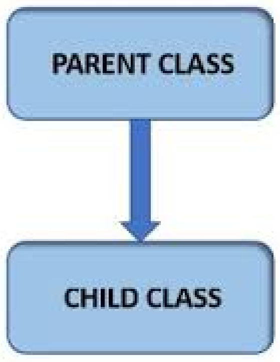
Inheritance is a process of acquiring the properties and behavior of one class inside another class.
Inheritance is a process of grouping all the classes which share common properties and behavior.
Inheritance allows creating new class or data type based on existing class or data type.


$$\begin{tikzcd}
 & \text{Class (Student)} \arrow[dl, "\text{Inherited by}"'] \arrow[dr, "\text{Inherited by}"] & \\
\text{Class (Bsc)} & & \text{Class (Ba)}
\end{tikzcd}$$

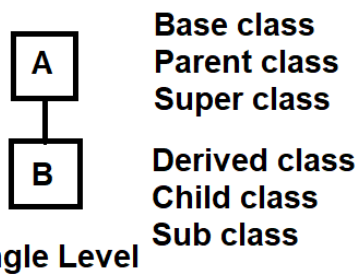


Advantage of inheritance,

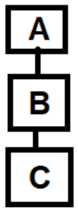
1. Reusability: The attributes and methods of one class can be used inside another class.
2. Easy to understand
3. Extensibility



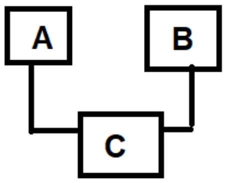
Based on the reusability of classes
1. Single level inheritance
2. Multilevel Inheritance
3. Multiple Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance



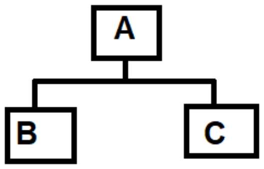
Single Level Inheritance



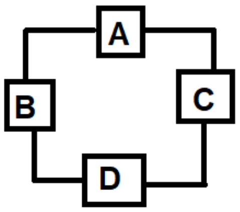
Multi Level Inheritance



Multiple Inheritance



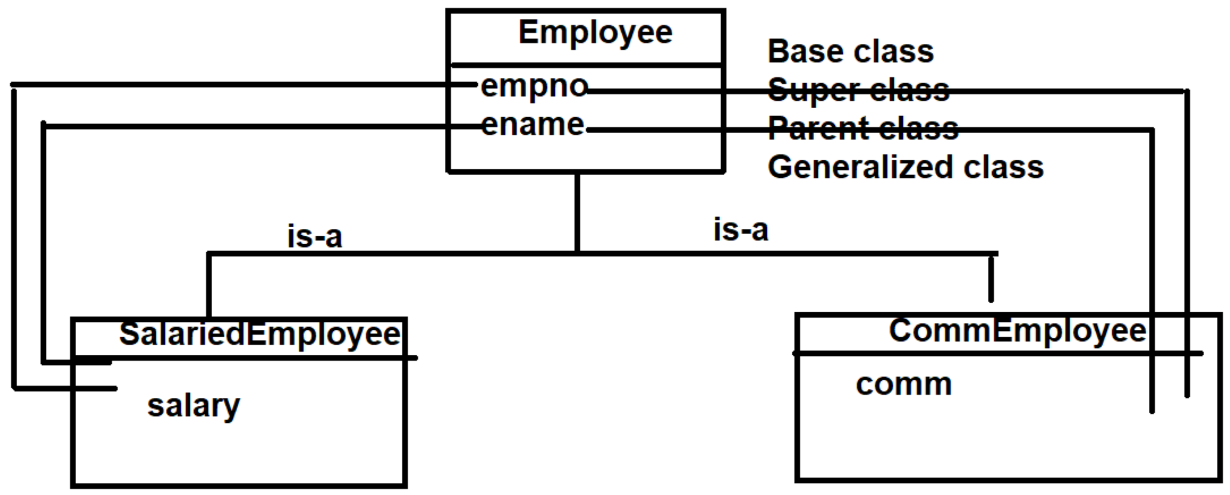
Hierarchical Inheritance



Hybrid Inheritance

Syntax:

```
class <derived-class-name>(base-class-name,base-class,base-class):
                variables
                methods
```



Methods of base class are automatically inherited within derived class.

Example:

```
class A:
    def m1(self):
        print("m1 of A class")
    def m2(self):
        print("m2 of A class")

class B(A):
    def m3(self):
        print("m3 of B class")


objb=B()
objb.m1()
objb.m2()
objb.m3()
```

Output:

```
m1 of A class
m2 of A class
m3 of B class
```

Variables/Properties of base class are not inherited automatically within derived class. In order to inherit properties of base class within derived class, constructor of derived class must call the constructor of base class.

super() function
This function returns reference of super class or parent class. Using super(), subclass/child class can refer to the members of parent class.

super().<method-name>
super().variable-name

Example:

```
class A:
    def __init__(self):
        self.x=100
        self.y=200

class B(A):
    def __init__(self):
        super().__init__()
        self.p=300
        self.q=400


objb=B()
print(objb.p,objb.q)
print(objb.x,objb.y)
```

Output:

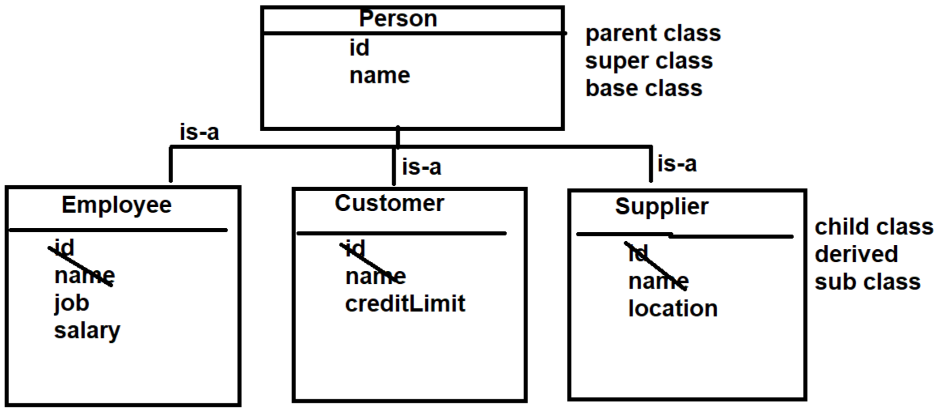
```
300 400
100 200
```

Example:

```
class A:
    def __m1(self):
        print("private method m1 of A")
    def _m2(self):
        print("protected method m2 of A")
    def m3(self):
        print("public method m3 of A")

class B(A):
    def m4(self):
        print("public method m4 of B")



objb=B()
objb.m3()
objb.m4()
```

Output:
```
public method m3 of A
public method m4 of B
```

Inheritance is process of grouping all the classes which share common properties and behavior.



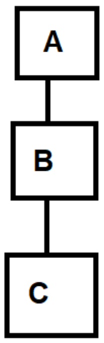
Example:
```
class Person:
    def __init__(self):
        self.__name=None
    def setName(self,n):
        self.__name=n
    def getName(self):
        return self.__name

class Student(Person):
    def __init__(self):
        super().__init__()
        self.__course=None
    def setCourse(self,c):
        self.__course=c
    def getCourse(self):
        return self.__course

stud1=Student()
stud1.setName("naresh")
stud1.setCourse("python")
print(f'Student Name {stud1.getName()}')
print(f'Student Course {stud1.getCourse()}')
```

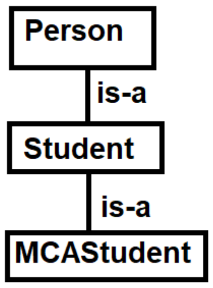
Output:

```
Student Name naresh
Student Course python
```

Multilevel Inheritance

If a class is derived from another derived class, it is called multilevel inheritance.





Example:

```
class A:
    def __init__(self):
        print("constructor of A")

class B(A):
    def __init__(self):
        super().__init__()
        print("constructor of B")

class C(B):
    def  __init__(self):
        super().__init__()
        print("constructor of C")

objc=C()
```

Output:

```
constructor of A
constructor of B
constructor of C
```