Aggregation
Aggregation is a special type of composition.
Where contained object is inject to container object using constructor or method.
In aggregation contained object exist independent of container object. If container object is destroy, still contained object is exists.
In composition if contained object is destroyed, it also destroy contained object.

Example:
```
class A:
    def __init__(self,x):
        self.__x=x
    def get_x(self):
        return self.__x

class B:
    def __init__(self,y,a):
        self.__y=y
        self.__a=a
    def add(self):
        return self.__y+self.__a.get_x()

def main():
    obja=A(200)
    objb=B(100,obja)
    result=objb.add()
    print(result)
main()
```

Output:
```
300
>>>
```

Example:
```
class Employee:
    def __init__(self,empno,ename,salary):
        self.__empno=empno
        self.__ename=ename
        self.__salary=salary
    def salary_compare(self,e):
        if self.__salary==e.__salary:
            return True
        else:
            return False
def main():
    emp1=Employee(101,"naresh",5000)
    emp2=Employee(102,"suresh",7000)
    res=emp1.salary_compare(emp2)
    print(res)
    emp3=Employee(103,"kishore",7000)
    res=emp3.salary_compare(emp2)
    print(res)
main()
```

Output:

```
False
True
>>>
```

Example:

```
class Student:
    def __init__(self,r,n):
        self.__rno=r
        self.__name=n
    def get_rno(self):
        return self.__rno
    def get_name(self):
        return self.__name
class Marks:
    def __init__(self,s1,s2):
        self.__sub1=s1
        self.__sub2=s2
    def find_result(self,s):
        print(f'Rollno {s.get_rno()}')
        print(f'Name {s.get_name()}')
        if self.__sub1<40 or self.__sub2<40:
            print('Result Fail')
        else:
            print('Result Pass')
def main():
    stud1=Student(1,"suresh")
    marks_stud1=Marks(30,70)
    marks_stud1.find_result(stud1)

main()
```

Output:

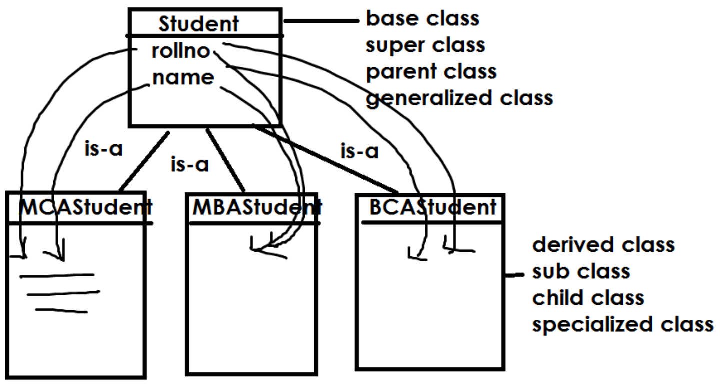
```
Rollno 1
Name suresh
Result Fail
>>>
```

Inheritance (IS-A)

Inheritance is process of acquiring the properties and behavior of one class inside another class.

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

One class derived from another class



Inheritance allows use the content of one class inside another class without creating object.

It provides reusability of variables and methods.

Type of inheritances

1. Single level inheritance
2. Multi level inheritance
3. Multiple inheritance

4. Hierarchical inheritance
5. Hybrid inheritance