'''

Experiment 4.1 : Inheritance with constructor

Name: Khan Arshad Abdulla
Roll No: 20CO24
2021-2022

Diamond Problem (Ambiguity Problem)

THEORY:

Inheritance enables us to define a class that takes all the functionality from a parent class and allows us to add more. In this tutorial, you will learn to use inheritance in Python.

Inheritance is a powerful feature in object oriented programming.

It refers to defining a new class with little or no modification to an existing class. The new class is called derived (or child) class and the one from which it inherits is called the base (or parent) class.

In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class.

In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name

Python provides us the flexibility to inherit multiple base classes in the child class.

'''

```
class Person:
    name=None
    addr=None
    age=None
    contact=None
    def __init__(self,n=None,a=None,age=None,c=None):
        self.name=n
        self.addr=a
        self.age=age
        self.contact=c
        print('I m in Person Class.',Person.__base__)
    def setprop(self,n,a,age,c):
        self.name=n
        self.addr=a
        self.age=age
        self.contact=c
    def getprop(self):
        return self.name,self.addr,self.age,self.contact
```

```
class Employee(Person):
    empno=None
    dept=None
    loc=None
    def __init__(self,e,n,a,age,c,d,l):
        self.empno=e
        self.dept=d
        self.loc=l
        super().__init__(n,a,age,c)

    def setemp(self,e,n,a,age,c,d,l):
        self.empno=e
        self.dept=d
        self.loc=l
        self.setprop(n,a,age,c)
    def getemp(self):
        #return self.empno,self.name,self.age,self.dept,self.loc
        return self.empno,self.getprop(),self.dept

class Student(Person):
    rollno=None
    classroom=None      #FECO or SECO or TECO or BECO
    ptr=None
    def __init__(self,r,n,a,age,c,cr,p):
        self.rollno=r
        self.classroom=cr
        self.ptr=p
        Person.__init__(self,n,a,age,c)     #self should be passed while
calling the base class constructor using base class name
    def setstud(self,r,n,a,age,c,cr,p):
        self.rollno=r
        self.classroom=cr
        self.ptr=p
        self.setprop(n,a,age,c)
    def getstud(self):
        return self.rollno,self.getprop(),self.classroom,self.ptr



#driver code
def main():
    e=Employee(1,"Ahmed Khan","Mumbai",28,9898998989,"Cyber
Security","Pune")
    #e.setemp(1,"Ahmed Khan","Mumbai",28,9898998989,"Cyber
Security","Pune")


    print(e.getemp())
    s=Student("20CO01","Ahmed Shaikh","Navi
Mumbai",18,9899998999,"SECO",8.9)
    #s.setstud("20CO01","Ahmed Shaikh","Navi
Mumbai",18,9899998999,"SECO",8.9)
    print(s.getstud())

if __name__=="__main__":
    main()

'''
```

OUTPUT:

```
I m in Person Class. <class 'object'>
(1, ('Ahmed Khan', 'Mumbai', 28, 9898998989), 'Cyber Security')
I m in Person Class. <class 'object'>
('20CO01', ('Ahmed Shaikh', 'Navi Mumbai', 18, 9899998999), 'SECO', 8.9)
```

CONCLUSION:

In this particular experiment we have successfully implemented inheritance. I understood that by using Inheritance we can inherite functions of parent class into child class, which provides reusability of code. Also, Multiple inheritance is possible in python which was not supported in java.

'''