```
'''
           Experiment 3.2 : Implementing Constructor in Python

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


THEORY:
        Constructors are generally used for instantiating an object. The
task of constructors is to initialize(assign values) to the data members
of the class
when an object of the class is created. In Python the __init__() method
is called the constructor and is always called when an object is created.

                ***Creating the constructor in python***
    In Python, the method the __init__() simulates the constructor of the
class. This method is called
when the class is instantiated. It accepts the self-keyword as a first
argument which allows
accessing the attributes or method of the class.

We can pass any number of arguments at the time of creating the class
object, depending upon
the __init__() definition. It is mostly used to initialize the class
attributes. Every class must
have a constructor, even if it simply relies on the default constructor.

    doctring => documentation string



'''
class Employee:
    '''
    An Employee class having employee attributes like empno, ename, and
sal.
    It also has methods like setprop, display, etc.
    '''
    #class attributes
    empno=None
    ename=None
    sal=None
    dept=None
    loc=None
    def setprop(self,num,name,sal,dept,loc):
        self.empno=num
        self.ename=name
        self.sal=sal
        self.dept=dept
        self.loc=loc
    def getprop(self):
        return self.empno,self.ename,self.sal,self.dept,self.loc

    def
__init__(self,num=None,name=None,sal=None,dept=None,loc=None,obj=None):

        if obj is None:
            self.empno=num
            self.ename=name
            self.sal=sal
            self.dept=dept
            self.loc=loc
        else:
            #print(obj.empno)
            self.empno=obj.empno
            self.ename=obj.ename
            self.sal=obj.sal
            self.dept=obj.dept
            self.loc=obj.loc
            #self=obj
        print("Constructor Executed.")


    # in Python, static methods are simply used as utility functions
    @staticmethod           # decorator
    def retire(age):
        if age>=60:
            print("Employee Retires.")
        else:
            print("Employee can still work.")



#driver code
def main():
    '''
    Our main function of Exp301 having a driver code.
    '''
    e=Employee()
    e.setprop(1,"Sachin",60000,"Computer","Mumbai")
    en,name,sal,dept,l=e.getprop()
    #print(en,name,sal,dept,l)
    e1=Employee()
    e1.setprop(2,"Shamim",54000,"Computer","Navi Mumbai")
    e2=Employee(10,sal=50000,dept="Computer Testing",loc="Pune")
    #e2.setprop(3,"Khatib",50000,"Computer Testing","Navi Mumbai")
    e3=Employee(obj=e)
    print(e3.getprop())
    el=[e,e1,e2]
    '''
    sal=0
    emp_hs=Employee()
    for i in el:
        if i.sal>sal:
            sal=i.sal
            emp_hs=i

    print("Employee with highest salary is",emp_hs.ename)
    Employee.retire(42)
    #print(el[0].ename)

    for i in el:
        print("Employee No.:",i.empno)
        print("Employee Name:",i.ename)
        #print(i.getprop(),type(i))
    '''


if __name__=="__main__":
    print(main.__doc__)
    main()




'''
OUTPUT:

 Our main function of Exp301 having a driver code.

Constructor Executed.
Constructor Executed.
Constructor Executed.
Constructor Executed.
(1, 'Sachin', 60000, 'Computer', 'Mumbai')




CONCLUSION:
        In this particular experiment we have successfully implemented
Constructor of the Employee Class. I understood that constructor of the
class is called
everytime we are creating instance/object of the class.
'''
```