```
class Human:
    def birth(self):
        self.gender=input("Enter gender")
    def naming(self):
        self.name=input("Enter name:")
    def intro(self):
        print("Hi i am a",self.gender,"called",self.name)
```

```
h=Human()
h.birth()
h.gender
```

```
Enter gendermale
'male'
```

```
h.naming()
```

```
Enter name:amar
```

```
h.name
```

```
'amar'
```

```
h.intro()
```

```
Hi i am a male called amar
```

```
class Human:
    def birth(self):
        print("Reference:",id(self))
        self.gender=input("Enter gender")
    def naming(self):
        self.name=input("Enter name:")
    def intro(self):
        print("Hi i am a",self.gender,"called",self.name)
```

```
h1=Human()
h2=Human()
```

```
h1.birth()
h2.birth()
```

```
Reference: 140354234205904
Enter gendermale
Reference: 140354234205088
Enter genderfemale
```

```
h2.naming()
```

```
Enter name:wonder woman
```

```
h1.naming()
```

```
Enter name:superman
```

```
id(h1)
```

```
140354234205904
```

```
id(h2)
```

```
140354234205088
```

```
h1.intro()
```

```
Hi i am a male called superman
```

```
h2.intro()
```

```
Hi i am a female called wonder woman
```

```
class Human:
    def birth(self):
        print(self)
        self.gender=input("Enter gender")
        self.naming()
    def naming(self):
        self.name=input("Enter name:")
    def intro(self):
        print("Hi i am a",self.gender,"called",self.name)
```

```
h=Human()

```

```
h.birth()
```

```
<__main__.Human object at 0x7fa6c445c0d0>
```

```
h.intro()
```

```
Hi i am a femal called wonder woman
```

```
class Human:
    def __init__(self):#constructor
        print("Obejct created:",id(self))
        self.gender=input("Enter gender")
        self.name=input("Enter name:")
    def __str__(self):
        return "Hi i am a "+self.gender+" called "+self.name
```

```
h=Human()
```

```
Obejct created: 140354234129760
Enter genderfemale
Enter name:cat woman
```

```
print(h)
```

```
<__main__.Human object at 0x7fa6c4449070>
```

```
print(h)
```

```
Hi i am a female called cat woman
```

```
class Person:
    def __init__(self):
        self.name=input("Enter name:")
    def __del__(self):
        print("R.I.P",self.name)
```

```
p=Person()
```

```
Enter name:heena
R.I.P zeena
```

```
#code style 1 :Basic user
class Human:
    def __init__(self):#constructor
        print("Obejct created:",id(self))
        self.gender=input("Enter gender")
        self.name=input("Enter name:")
    def __str__(self):
        return "Hi i am a "+self.gender+" called "+self.name
```

```
#code style-2 industry manner
class Human:
    def __init__(self,name,gender):#constructor
        print("Obejct created:",id(self))
```

```
        self.gender=gender
        self.name=name
    def __str__(self):
        return "Hi i am a "+self.gender+" called "+self.name
```

```
n=input("Name:")
g=input("Gender:")
h=Human(n,g)
```

```
Name:shaktiman
Gender:male
Obejct created: 139730521559920
```

```
print(h)
```

```
Hi i am a male called shaktiman
```

```
class Person:
    def __init__(self,gender,name,number):
        self.gender=gender
        self._name=name
        self.__number=number
    def __vibhishan(self):
        print(self.__number)
```

```
p=Person("male","amar",9821601163)
```

```
p.__vibhishan()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-28-b202dd54045f> in <cell line: 1>()
----> 1 p.__vibhishan()

AttributeError: 'Person' object has no attribute '__vibhishan'
```

SEARCH STACK OVERFLOW

```
class Human:
    count=0
    def __init__(self,name,gender):#constructor
        print("Obejct created:",id(self))
        self.gender=gender
        self.name=name
        Human.count+=1

    def __str__(self):
        return "Hi i am a "+self.gender+" called "+self.name

    def populaion(self):
        print("Total humans:",Human.count)
```

```
h=Human("amar","male")
```

```
Obejct created: 139730264722256
```

```
h2=Human("amrita","female")
```

```
Obejct created: 139730521677296
```

```
h3=Human("samrita","female")
```

```
Obejct created: 139730521676432
```

```
h.populaion()
```

```
Total humans: 3
```

```
Human.count
```

```
5
```

```
class A:
    def mA1(self):
        print("mA1 called")
    def _mA2(self):
        print("mA2 called")
    def __mA3(self):
        print("mA3 called")
```

```
class B(A):
    def mB1(self):
        print("mB1 called")
    def _mB2(self):
        print("mB2 called")
    def __mB3(self):
        print("mB3 called")
```

```
obj=B()
```

```
obj.__mA3()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-52-9c3b559fb3f9> in <cell line: 1>()
----> 1 obj.__mA3()

AttributeError: 'B' object has no attribute '__mA3'
```

SEARCH STACK OVERFLOW

```
class A:
    def my(self):
```

```
        print("A's my")
    def myA(self):
        print("A's")
class B:
    def my(self):
        print("B's my")
    def myB(self):
        print("B's")
class C:
    def my(self):
        print("C's my")
    def myC(self):
        print("C's")

```

```
class X(B,C,A):
    def myself(self):
        print("hi X")
```

```
obj=X()
```

```
#override
class Parent:
    def welcome(self):
        print("hello how are you how do you do bla bla")
    def other(self):
        print("other things")
class Child(Parent):
    def welcome(self):
        print("hi")
    def chill(self):
        print("just chilling")

```

```
objc=Child()
objp=Parent()
```

```
objp.welcome()
```

```
hello how are you how do you do bla bla
```

```
obj.welcome()
```

```
hi
```

```
class Circle:
    def readR(self,r):
        self.r=r
    def area(self):
        print("Area of Circle:",3.14*self.r**2)
```

```
c=Circle()
r=float(input("Read r:"))
c.readR(r)
c.area()
```

```
Read r:5.6
Area of Circle: 98.4704
```

```
class Emp:
    def setdetails(self,id,name,salary):
        self.__id=id
        self.__name=name
        self.__salary=salary
    def printdetails(self):
        print(self.__id,self.__name,self.__salary)
```

```
class Emp:
    def __init__(self,id,name,salary):
        self.__id=id
        self.__name=name
        self.__salary=salary
    def __str__(self):
        return str(self.__id)+" "+self.__name+" "+str(self.__salary)
```

```
class Emp:
    count=202300
    def __init__(self,name,salary):
        Emp.count+=1
        self.__id=Emp.count
        self.__name=name
        self.__salary=salary
        print("Record added:ID-->",self.__id)
    def __str__(self):
        return str(self.__id)+" "+self.__name+" "+str(self.__salary)
    def getempid(self):
        return self.__id
```

```
emplist=[]
while True:
    print("1.Create\n2.Search\n3.Delete\n0.Exit")
    ch=int(input(":"))
    if ch==1:
        n=input("Enter name:")
        s=float(input("salary:"))
        e=Emp(n,s)
        emplist.append(e)
    elif ch==2:
        tempid=int(input("Enter id:"))
        flag=False
        for i in emplist:
            if i.getempid()==tempid:
                print("Found:")
                flag=True
                print(i)
```

```
                break
        if flag==False:
            print("Not Found")
    elif ch==3:
        tempid=int(input("Enter id:"))
        flag=False
        for i in range(len(emplist)):
            if emplist[i].getempid()==tempid:
                print("Found:")
                flag=True
                print("Deleted:",emplist.pop(i))
                break
        if flag==False:
            print("Not Found")

    elif ch==0:
        print("bye")
        break
    else:
        print("Wrong choice given")
```

```
1.Create
2.Search
3.Delete
0.Exit
:1
Enter name:aaaaa
salary:9000
Record added:ID--> 202304
1.Create
2.Search
3.Delete
0.Exit
:1
Enter name:bbbb
salary:4567
Record added:ID--> 202305
1.Create
2.Search
3.Delete
0.Exit
:3
Enter id:202304
Found:
Deleted: 202304 aaaaa 9000.0
Not Found
1.Create
2.Search
3.Delete
0.Exit
:9
Wrong choice given
1.Create
2.Search
3.Delete
```

```
0.Exit
:0
bye
```

```
emplist=[]
for i in range(3):
    n=input("Enter name:")
    s=float(input("salary:"))
    e=Emp(n,s)
    emplist.append(e)
```

```
Enter name:aaaa
salary:1111
Record added:ID--> 202301
Enter name:bbbb
salary:2222
Record added:ID--> 202302
Enter name:cccc
salary:3333
Record added:ID--> 202303
```

```
print(emplist)
for i in emplist:
    print(i)
```

```
[<__main__.Emp object at 0x7f159401e3d0>, <__main__.Emp object at 0x7f158c1a14f0>, <__main__.Emp object at 0x7f157cac8f40>]
202301 aaaa 1111.0
202302 bbbb 2222.0
202303 cccc 3333.0
```

```

n=input("Enter name:")
s=float(input("salary:"))
e1=Emp(n,s)
print(e)#e.printdetails()
```

```
Enter name:amar
salary:900
Record added:ID--> 202304
202303 zeena 8000
```

```
e2=Emp("Heman",5000)
```

```
Record added:ID--> 202307
```

```
e3=Emp("zeena",8000)
```

```
Record added:ID--> 202308
```

```
print(e)
```

```
202303 zeena 8000
```

```
class Insan:
    pass
```

```
i=Insan()
```

```
setattr(i,"name","xmax")
```

```
setattr(i,"__contact",9821601163)
```

```
getattr(i,"age","not given")
```

```
'not given'
```

```
#super()---->allow access of super class's methods
class A:
    def __init__(self,d1):
        print("A",d1)
class B(A):
    def __init__(self,d1,d2):
        super().__init__(d1)
        print("B",d2)
class X(B):
    def __init__(self,d1,d2,d3):
        super().__init__(d1,d2)
        print("X",d3)
obj=X(11,22,33)
```

```
A 11
B 22
X 33
```

```
class Person:
    def __init__(self):
       print("the one")
    def __init__(self):
       print("the two")
    def __init__(self):
       print("the last")
    def __del__(self):
        print("R.I.P",self.name)
```

```
p=Person()
```

```
the last
```

```
class triangle:
    def setLH(self,l,h):
        self.l=l
        self.h=h
    def area(self):
        print("Area is:",(0.5*self.l*self.h))
```

```
t=triangle()
l=float(input("L:"))
h=float(input("H:"))
t.setLH(l,h)
t.area()
```

```
L:56
H:12
Area is: 336.0
```

```
#create a bank class
# has amount,accountno,name
# createaccount(constructor)-user only gives name and amount
# account number auto generated
#withdraw(amount):should not be -ve amount and min
# balance is 2000 else reject transection
#deposite(amount):amount can not be -ve
#checkbalance():shows balance and account holder name

'''
menu driven code to
1 create account
2 withdraw
3 deposite
4 check balance

1---->create by takingvalues and auto generate account number
2/3/4--->ask account number ,search account and then operate
```