**Duck Typing Python**

=====================

In python we are not going to declare the object type explicitly bcz it is Dynamically typed programming language....

Then how can you decide this object is what type. if it is walks like duck and talks like duck .....

It is called duck type

Ex

class Duck():

def talk(self):

print("Quack Quack....")

class Dog():

def bark(self):

print("Bou Bou....")

class Goat():

def talk(self):

print("Myaa Myaa....")

duck=Duck()

dog=Dog()

goat=Goat()

l=[dog,duck,goat]

for obj in l:

if hasattr(obj,'talk'):

obj.talk()

elif hasattr(obj,'bark'):

obj.bark()

**OverLoading**

================

a.Operator overloading

b.Method Overloading

c.Constructor Overloadin

***\* operator overloading \****

===========================

+operaor

10+20 =30

'durga'+'soft'=durgasoft

\* operator

10\*20=200

'durga'\*3=durgadurgadurga

*#example 1***class** Books():  
 **def** \_\_init\_\_(self,pages):  
 self.pages=pages  
  
 **def** \_\_add\_\_(self,other):  
 **return** (self.pages+other.pages)  
  
  
b1=Books(100,)  
b2=Books(200,)  
  
print(**"Total no of pages"**,b1+b2)  
  
*#example 2***class** Student():  
 **def** \_\_init\_\_(self,marks):  
 self.marks=marks  
 **def** \_\_lt\_\_(self, other):  
 **return** self.marks<other.marks  
  
  
s1=Student(100)  
s2=Student(200)  
print(**"s1<s2"**,s1<s2)  
  
  
*#example 3***class** Employee():  
 **def** \_\_init\_\_(self,name,salary):  
 self.name=name  
 self.salary=salary  
  
 **def** \_\_mul\_\_(self, other):  
 **return** self.salary\*other.days  
  
  
**class** TimeSheet():  
 **def** \_\_init\_\_(self,name,days):  
 self.name=name  
 self.days=days  
  
  
e=Employee(**'Nitish'**,500)  
t=TimeSheet(**'Nitish'**,25)  
print(e.name,**'\'s'**,**"Total Salary is "**,e\*t)

**Method Overloading**

*#exmp 1***class** Test():  
 **def** sum(self,a=**None**,b=**None**,c=**None**):  
 **if** a!=**None and** b!=**None and** c!=**None**:  
 res=a+b+c  
 **elif** a!=**None and** b!=**None**:  
 res=a+b  
 **else**:  
 res=(**'Not valid Input'**)  
 print(res)  
  
  
t=Test()  
t.sum(12,165,656)  
  
  
*#exmp 2***class** Student():  
 **def** sum(self,\*a):  
 res=0  
 **for** i **in** a:  
 res=res+i  
 print(res)  
  
s=Student()  
s.sum(12,10)

**Python doesn’t support method Overloading and constructor over loading**