polymorphism s-

He use for method Overboading & Overstoding

Method Overlanding :

It consiste of same name but different parameter

Ens class Demos

def mi(self):

print (" @ parameters")

def mi(selfia):

print (" / parameter")

def mi(self,a,b):

print (" a parrouneters")

0 = Demo()

0.m1()

Olps Boror it missing 2 parameters?

If we remove all parameters definiself, a) as definiself): & def miself, a, b): as def miself): then the output will be

Olpt par 2 pagrameters

In method overdooding, it try to Enecutes only last method. The can conclude that method overloading is not going to Support by python.

As there in java of in other languages we use data types, but in python we no need.

ENG Class Demos

def mi (self, a):

print (" o parameters", a)

0 = Demo()

0.ml ("dhaoima")

0.W1(10)

0.m1(8.5)

Op- Operameters dharma

O parameters 10

O parameters 2.5

In python, By default a support is there for Method Overdeading.

No we can conclude in python those is no need the Method Overwloading.

```
they present pattories of bottoff
Class A:
                translate of withhomsty in tod
   def m1(self):
        print ("I'm from mi") when you what al
Class B(A):
    def mi(self);
        print ("I don't need implementation")
 0 2B()
 O.mi()
Olpe-I don't need Emplementation
House Same method name is taken & it takes second method.
If we also want to implement parent method.
         mi(self):
 Class A:
          print ("I'm from mi")
  class 8:
      def mi(self)?
           print ("I don't need implementation")
           Super()·MI()
   0=8()
  0-m1()
 Olp=9 don't need implementation
     gim from m1.
```

```
Hethod Overstoling means with same method name
 but implementation is different.
het take one more Example.
 Class Parent:
      def car (self):
          print (Benx Carn)
  class Child (Parent):
       def car (self):
           Print ("BMW")
  oschilder was it was some some
  O-Car()
  olps BMW
  Superil is a keyword that helps to print from parent
  Class Parent:
       def con(self)?
          Print (Benz Cart)
   colors Child (Parent);
       det car(self):
          print (BMW) Statement for the Bay (
          Super (1. Car()
  O=childi)
  0. (arl)
O[p & BHW
```

Benz