Constructor Concept:

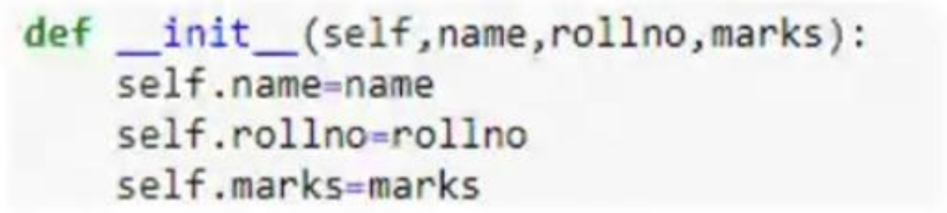
- Constructor is a special method in python.
- The name of the constructor should be __init__(self)
- Constructor will be executed automatically at the time of object creation.
- The main purpose of constructor is to declare and initialize instance variables.
- Per object, constructor will be executed only once.
- Constructor can take at least one argument(at least self)
- Constructor is optional and if we are not providing any constructor

  then python will provide default constructor.

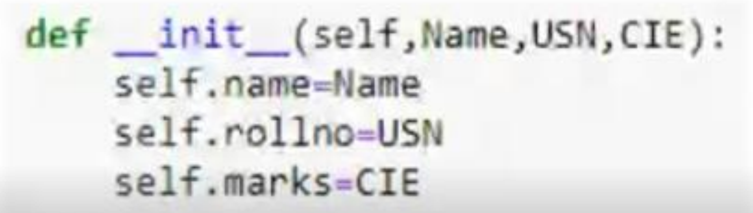
DEEPA N P

Example:

```
def __init__(self,name,rollno,marks):
        self.name=name
        self.rollno=rollno
        self.marks=marks
```



```
def __init__(self,name,rollno,marks):
    self.name=name
    self.rollno=rollno
    self.marks=marks
```



```
def __init__(self,Name,USN,CIE):
    self.name=Name
    self.rollno=USN
    self.marks=CIE
```

Method vs Constructor

Method	Constructor
Name of method can be any name	Constructor name should be always __init__
Method will be executed if we call that method	Constructor will be executed automatically at the time of object creation.
Per object, method can be called any number of times.	Per object, Constructor will be executed only once
Inside method we can write logic	Inside Constructor we have to declare and initialize instance variables

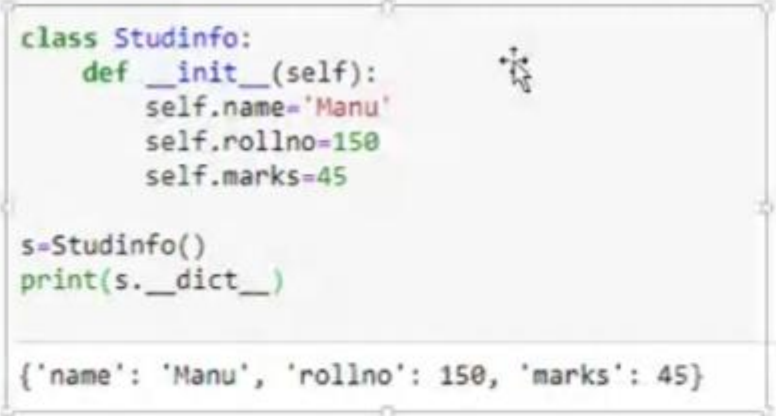
Types of Variables:

Inside Python class 3 types of variables are allowed.

1. Instance Variables (Object Level Variables)
2. Static Variables (Class Level Variables)
3. Local variables (Method Level Variables)

Inside Constructor by using self variable:

- We can declare instance variables inside a constructor by **using self** keyword. Once we creates object, automatically these variables will be added to the object.



```
class Studinfo:
    def __init__(self):
        self.name='Manu'
        self.rollno=150
        self.marks=45

s=Studinfo()
print(s.__dict__)
```

```
{'name': 'Manu', 'rollno': 150, 'marks': 45}
```

Inside Instance Method by using self variable:

We can also declare instance variables inside instance method by using self variable. If any instance variable declared inside instance method, that instance variable will be added once we call that method.

```python
class Test:
    def __init__(self):
        self.a=10
        self.b=20

    def m1(self):
        self.c=30


t=Test()
t.m1()
print(t.__dict__)
```

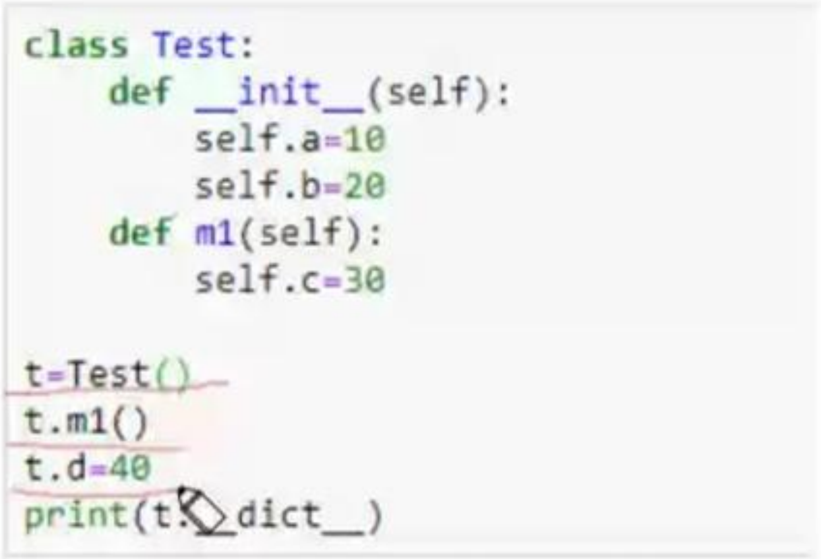
```
{'a': 10, 'b': 20, 'c': 30}
```

```python
class Test:
    def __init__(self):
        self.a=10
        self.b=20

    def m1(self):
        self.c=30


t=Test()
#t.m1()
print(t.__dict__)
```

```
{'a': 10, 'b': 20}
```

Outside of the class by using object reference variable:

- We can also add instance variables outside of a class to a particular object.

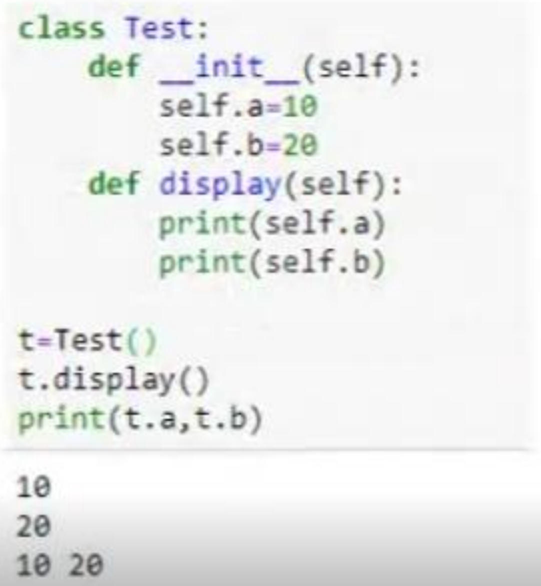


```
class Test:
    def __init__(self):
        self.a=10
        self.b=20
    def m1(self):
        self.c=30

t=Test()
t.m1()
t.d=40
print(t.__dict__)
```

```
{'a': 10, 'b': 20, 'c': 30, 'd': 40}
```

How to access Instance variables:

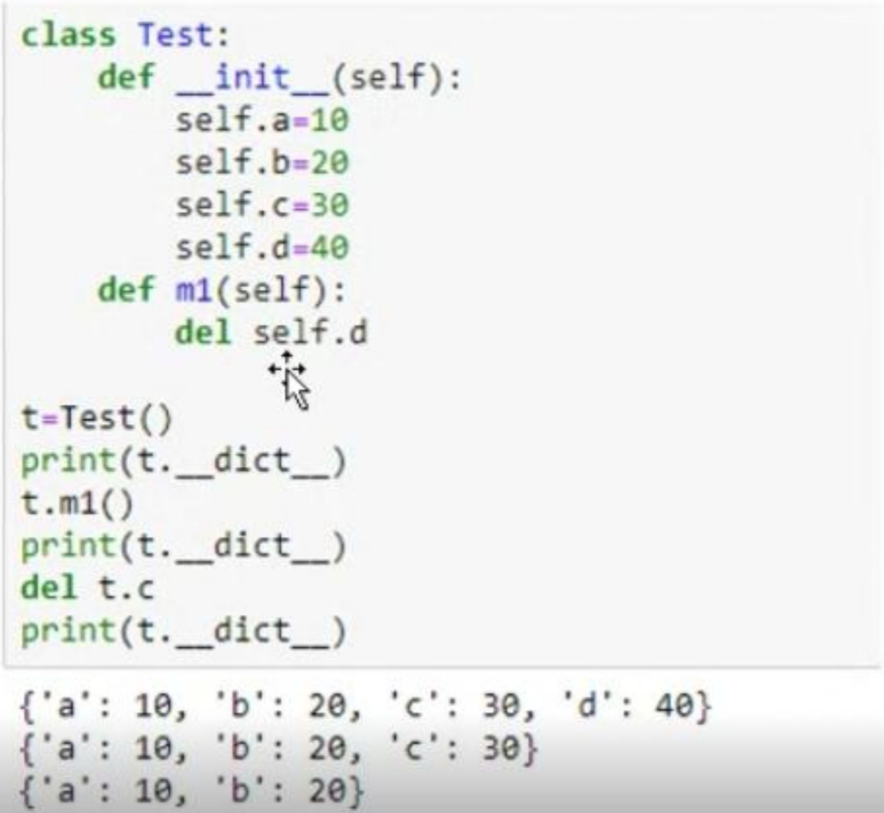
We can access instance variables with in the class by using **self variable** and outside of the class by using object reference.



```python
class Test:
    def __init__(self):
        self.a=10
        self.b=20
    def display(self):
        print(self.a)
        print(self.b)

t=Test()
t.display()
print(t.a,t.b)
```

```
10
20
10 20
```

Example:

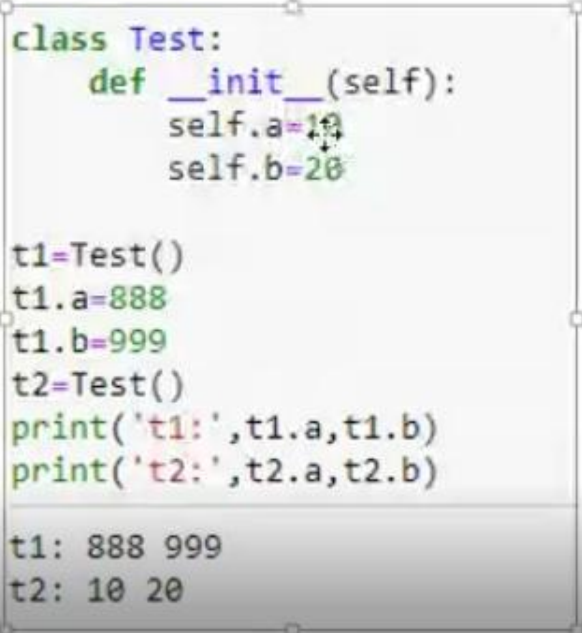


```
class Test:
    def __init__(self):
        self.a=10
        self.b=20
        self.c=30
        self.d=40
    def m1(self):
        del self.d


t=Test()
print(t.__dict__)
t.m1()
print(t.__dict__)
del t.c
print(t.__dict__)
```

```
{'a': 10, 'b': 20, 'c': 30, 'd': 40}
{'a': 10, 'b': 20, 'c': 30}
{'a': 10, 'b': 20}
```

Example

If we change the values of instance variables of one object then those changes won't be reflected to the remaining objects, because for every object we are separate copy of instance variables are available.



```python
class Test:
    def __init__(self):
        self.a=10
        self.b=20

t1=Test()
t1.a=888
t1.b=999
t2=Test()
print('t1:',t1.a,t1.b)
print('t2:',t2.a,t2.b)
```

```
t1: 888 999
t2: 10 20
```