Creating instance variables within class using instance method

Inside class, instance variables are bind with “**self**”.
(OR) any variable inside class access or bind with “**self**” is called instance variable.

Example:

```
class Employee:
    def create_properties(self):
        self.empno=1
        self.ename="naresh"
        self.salary=5000
    def print_employee(self):
        print(f'EmployeeNo {self.empno}')
        print(f'EmployeeName {self.ename}')
        print(f'Salary {self.salary}')



emp1=Employee()
emp1.create_properties()
emp1.print_employee()
emp2=Employee()
emp2.create_properties()
emp2.print_employee()
```

Output

```
EmployeeNo 1
EmployeeName naresh
Salary 5000
EmployeeNo 1
EmployeeName naresh
Salary 5000
```

Example:

```
class Employee:
    def create_properties(self,e,en,s):
```

```
        self.empno=e
        self.ename=en
        self.salary=s
    def print_employee(self):
        print(f'EmployeeNo {self.empno}')
        print(f'EmployeeName {self.ename}')
        print(f'Salary {self.salary}')



emp1=Employee()
emp1.create_properties(1,"naresh",6000)
emp1.print_employee()
emp2=Employee()
emp2.create_properties(2,"suresh",8000)
emp2.print_employee()
```

Output
EmployeeNo 1
EmployeeName naresh
Salary 6000
EmployeeNo 2
EmployeeName suresh
Salary 8000

Constructor

Constructor is instance method.
Constructor is a magic method and executed automatically whenever object of a class is created.
Constructor is used for initialization of object (OR) creating instance variable.
It is used to define properties of the object.
The name of the constructor is __init__(self).

Constructor can be defined,
1. Without parameters
2. With parameters

Note: block of code which has to execute on creation of object is written inside constructor.

Constructor without parameters does not receive values.
Constructor with parameters receives values.

Example:

```
class Product:
    def __init__(self):
        self.prodname=None # I.V
        self.price=None # I.V



p1=Product()
print(p1.prodname,p1.price)
p2=Product()
print(p2.prodname,p2.price)
```

Output

None None
None None

Example:

```
class Product:
    def __init__(self,pn=None,p=None):
        self.prodname=pn
        self.price=p


p1=Product("mouse",200)
p2=Product("keyboard",1500)
print(p1.prodname,p1.price)
print(p2.prodname,p2.price)
p3=Product()
print(p3.prodname,p3.price)
```

```
c1=complex()
print(c1.real,c1.imag)
c2=complex(1.2,1.5)
print(c2.real,c2.imag)
```

Output
mouse 200
keyboard 1500
None None
0.0 0.0
1.2 1.5

What is difference between constructor instance method and normal instance method?

Instance method	**Constructor**
This method name can be any name.	Constructor name must be __init__
Within class, we can define any number of instance methods	Within class, we can define only one constructor
Instance methods must be called explicitly (OR) these methods are not called automatically	This method is called automatically on creation of object
Instance method can be invoked any number of times	Constructor is invoked only once.
The job of instance method setter and getter operation.	The job of constructor is initialization.

Example:

```
class Date:
    def __init__(self):
        self.dd=None
        self.mm=None
        self.yy=None
```

```
dob=Date()
dob.dd=10
dob.mm=8
dob.yy=2024

print(dob.dd,dob.mm,dob.yy)

doj=Date()
doj.dd=1
doj.mm=6
doj.yy=2022

print(doj.dd,doj.mm,doj.yy)
```

Output
10 8 2024
1 6 2022

Example:

```
class Date:
    def __init__(self):
        self.dd=None
        self.mm=None
        self.yy=None
    def set_date(self,d,m,y):
        self.dd=d
        self.mm=m
        self.yy=y
    def print_date(self):
        print(f'{self.dd}/{self.mm}/{self.yy}')

dob=Date()
dob.set_date(10,8,2024)
dob.print_date()

doj=Date()
```

```
doj.set_date(1,6,2024)
doj.print_date()
```

Output
10/8/2024
1/6/2024

The advantage of encapsulation is data hiding, preventing data access outside functions or unrelated operations.

Access Modifiers

Access modifiers define the accessibility of the members of the class.
Python provides 3 access modifiers

1. Private
2. Public
3. Protected