Python Constructor Concept:

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

A constructor is a special type of method (function) which is used to initialize the instance members of the class.
In C++ or Java, the constructor has the same name as its class, but it treats constructor differently in Python. It is used to create an object.

Creating the constructor in python:

In Python, the method the __init__() represents the constructor of the class.

This method is called when the class is instantiated (object created).

It accepts the self-parameter as a first argument which allows accessing the attributes or method of the class.

We can pass any number of arguments at the time of creating the class object, depending upon the __init__() definition.

It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.

Example:

```
class Employee:
    def __init__(self, name, id):
        self.id = id
        self.name = name
    def display(self):
        # print("ID: %d \nName: %s" % (self.id, self.name))
        # print("ID: {1} \nName: {0}".format( self.name,self.id))
        print(f"ID: {self.id} \nName: {self.name}")
emp1 = Employee("John", 101)
emp2 = Employee("David", 102)
# accessing display() method to print employee 1 information
emp1.display()
# accessing display() method to print employee 2 information
emp2.display()
```

Output:

```
ID: 101
Name: John
ID: 102
Name: David
```

Counting the number of objects of a class

--->> The constructor is called automatically when we create the object of the class.

Consider the following example.

Example:

```
class Student:
    count = 0
    def __init__(self):
        Student.count = Student.count + 1
s1=Student()
s2=Student()
s3=Student()
print("The number of students:",Student.count)
```

Output:

```
The number of students: 3
```

Constructors can be of two types.

1. Parameterized Constructor
2. Non-parameterized Constructor

Constructor definition is executed when we create the object of this class. Constructors also verify that there are enough resources for the object to perform any start-up task.

Python Non-Parameterized Constructor

--->> The non-parameterized constructor uses when we do not want to manipulate the value or the constructor that has only self as an argument.

Example:

```
class Student:
    # Constructor - non parameterized
    def __init__(self):
        print("This is non parametrized constructor")
    def show(self,name):
        print("Hello",name)
student = Student()
student.show("John")
```

Python Parameterized Constructor:

---->> The parameterized constructor has multiple parameters along with the self.

Consider the following example.

```
class Student:
    # Constructor - parameterized
    def __init__(self, name):
        print("This is parametrized constructor")
        self.name = name
    def show(self):
        print("Hello",self.name)
student = Student("John")
student.show()
```

Output:

```
This is parametrized constructor
Hello John
```

Python Default Constructor

--->> When we do not include the constructor in the class or forget to declare it, then that becomes the default constructor.

--->> It does not perform any task but initializes the objects.

Example:

```
class Student:
    roll_num = 101
    name = "Joseph"
    def display(self):
        print(self.roll_num,self.name)
st = Student()
```

```
st.display()
```

Output:

```
101   Joseph
```

More than One Constructor in Single class

--->> Let's have a look at another scenario, what happen if we declare the two same constructors in the class.

Example:

```
class Student:
    def __init__(self):
        print("The First Constructor")
    def __init__(self):
        print("The second contructor")
st = Student()
```

Output:

```
The Second Constructor.
```

--->> In the above code, the object st called the second constructor whereas both have the same configuration.

--->> The first method is not accessible by the st object.

--->> Internally, the object of the class will always call the last constructor if the class has multiple constructors.