# Constructor

A constructor is a special type of method (function) which is used to initialize the instance members of the class. A constructor is a class function that begins with double underscore (_). The name of the constructor is always the same __init__().

While creating an object, a constructor can accept arguments if necessary. When we create a class without a constructor, Python automatically creates a default constructor that doesn't do anything. Every class must have a constructor, even if it simply relies on the default constructor.

Two types of constructors in Python

1. **default constructor**
   this is the one, which we have seen in the above example. This constructor doesn’t accept any arguments.

2. **parameterized constructor**
   constructor with parameters is known as parameterized constructor.

# Default constructor

| Example | Output |
|---|---|
| class abc:<br>  def __init__(self):<br>  print "Default Constructor" | Default Constructor<br>Value of A is 10 and B is 20 |

| | |
|---|---|
| def fun(self,a,b):<br>print "Value of A is", a,"and B is", b<br>obj=abc()<br>obj.fun(10,20) | |

# Parameterized constructor

When we declare a constructor in such a way that it accepts the arguments during object creation then such type of constructors are known as Parameterized constructors. As you can see that with such type of constructors we can pass the values (data) during object creation, which is used by the constructor to initialize the instance members of that object.

| Example | Output |
|---|---|
| class abc:<br>def __init__(self,a,b):<br>print "Parameterized Constructor"<br>self.a=a<br>self.b=b<br><br>def fun(self):<br>print "Value of A is", self.a,"and B is", self.b<br>obj=abc(10,20)<br>obj.fun() | Parameterized Constructor<br>Value of A is 10 and B is 20 |

# Destructor

Destructors are called when an object gets destroyed. It’s the polar opposite of the constructor, which gets called on creation.

Destructor can be very useful for releasing resources before coming out of the program like closing files, releasing memories etc. Destructor is defined using __del__() keyword.

| Example | Output |
|---|---|
| class A:<br>  def __init__(self):<br>  print('Hello Python')<br><br>  def __del__(self):<br>  print('Destructor called')<br><br>obj = A()<br>print('Hi Python') | Hello Python<br>Hi Python<br>Destructor called |