



Creating Classes

Class is like a container which includes all methods and variables.

The name of the class immediately follows the keyword *class* followed by a colon as follows –

class ClassName:





Example of class declaration and object creation

class abc:

a=10

def fun1(self,n1,n2):

self.nn1=n1

self.nn2=n2

self.nn1=self.nn1+self.nn2

print("sum is",self.nn1)

o1 = abc()

print(o1.a)

o1.fun1(10,20)

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\python programs\4object and classes\basic.py =====
10
sum is 30
>>> |
```





Self??

All methods in python have first parameter self .
This parameter refers to the object which
invokes the method. When you call a method
you don't need to pass anything to self
parameter.





Constructor

The first method `__init__()` is a special method, Python calls this method when you create a new instance of this class. This method is called class constructor or initialization method

Python doesn't have explicit constructors like C++ or Java, but the `__init__()` method in Python is something similar, it is strictly speaking not a constructor. It behaves in many ways like a constructor, e.g. it is the first code which is executed, when a new instance of a class is created.



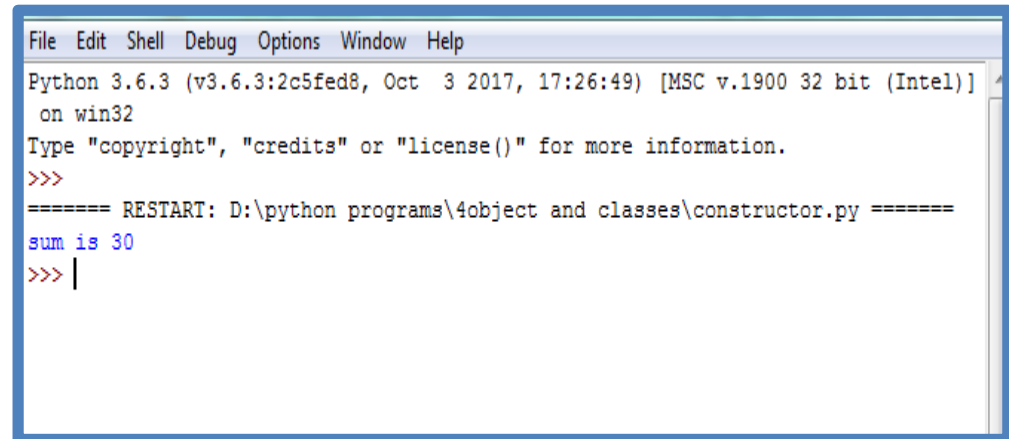


Constructor

```
class Abc:
    def __init__(self, n1,n2):
        self.nn1 = n1
        self.nn2 = n2
    def fun1(self):

        self.nn1=self.nn1+self.nn2
        print("sum is",self.nn1)
```

```
o1 = Abc(10,20)
o1.fun1()
```



```
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\python programs\4object and classes\constructor.py =====
sum is 30
>>> |
```





Destructor:-

There is no "real" destructor, but something similar, i.e. the method `__del__`. It is called when the instance is about to be destroyed.

```
class Abc:
```

```
    def __init__(self, n1,n2):
```

```
        self.nn1 = n1
```

```
        self.nn2 = n2
```

```
    def __del__(self):
```

```
        print ("destructor")
```

```
    def fun1(self):
```

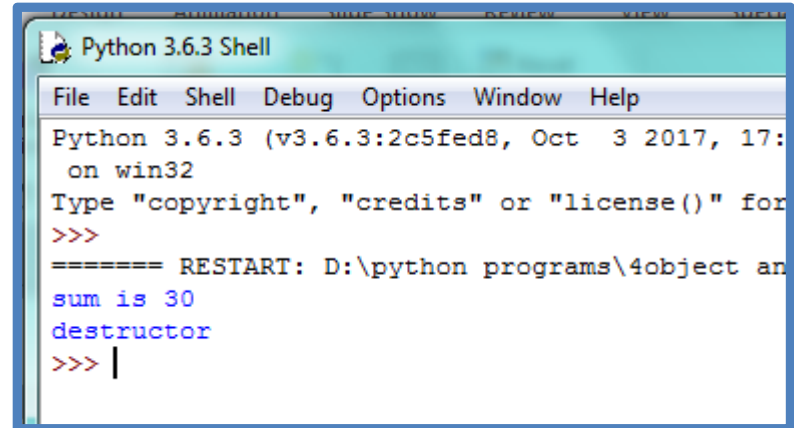
```
        self.nn1=self.nn1+self.nn2
```

```
        print("sum is",self.nn1)
```

```
o1 = Abc(10,20)
```

```
o1.fun1()
```

```
del o1
```



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:
on win32
Type "copyright", "credits" or "license()" for
>>>
===== RESTART: D:\python programs\4object an
sum is 30
destructor
>>> |
```

