Python Constructors and Destructors

Anoop S Babu

Faculty Associate

Dept. of Computer Science & Engineering

bsanoop@am.amrita.edu

Constructors in Python

- The __init__() method inside a class serves as constructor.
- Its purpose is to initialize the object.
- This method runs as soon as the object is instantiated.
- Can have single or any number of parameters

Constructors in Python - Example

```python
class Student:
    """Base class to all Students"""
    def __init__(self,regno,name,program):
        self.regno = regno
        self.name = name
        self.program = program

S1 = Student(101,"Ananya","MCA")
```

Object instantiation statement invokes the constructor

Self Parameter

- Self is a pointer pointing to the invoking object.
- Like ‘this’ in C++
- Self works as a parameter of function
- But it is not used while calling

```
class Student:
    """Base class to all Students"""
    def __init__(self,regno,name):
        self.regno = regno
        self.name = name

    def dispStudentInfo(self):
        print("Register Number: ",self.regno)
        print("Name: ",self.name)

S1 = Student(101,"Ananya")
S1.dispStudentInfo()
```

Methods

- Methods – functions defined inside the body of the class
- Used to define the behaviors of the object
- Creation similar to a normal function.
- One mandate parameter – self
- To call the method use *objName.methodName()*

Methods - Example

```
class Student:
    """Base class to all Students"""
    def __init__(self,regno,name):
        self.regno = regno
        self.name = name

    def dispStudentInfo(self):
        print("Register Number: ",self.regno)
        print("Name: ",self.name)

S1 = Student(101,"Ananya")
S1.dispStudentInfo()
```

Method definition

Calling a method

Passing object as method parameter

Program

```
class Point:
    def __init__(self,x,y):
        self.x = x
        self.y = y
    def equals(self,obj):
        if (self.x == obj.x and self.y == obj.y):
            return True
        else:
            return False

P1 = Point(2,5)
P2 = Point(2,5)
P3 = Point(3,5)
print("P1 == P2: ",P1.equals(P2))
print("P1 == P3: ",P1.equals(P3))
```

Output

```
P1 == P2:  True
P1 == P3:  False
```

Self parameter with method

- Self is used to call a method from another one

```
class A:
    def m1(self):
        print("In m1---->Called from Method 2")
    def m2(self):
        print("Method 2")
        self.m1()
a = A()
a.m2()
```

$\leftarrow$ throws an error if called without self

Output

```
Method 2
In m1---->Called from Method 2
```

Destructors in Python

- Python automatically deletes an object that is no longer in use
- It’s called as garbage collection
- Python periodically performs garbage collection
- Explicitly do this using a destructor
- A special method __del__()
- Explicitly invoked when an object is about to be destroyed

Destructors in Python – Example

Program

```python
class Student:
    """Base class to all Students"""
    def __init__(self,regno,name):
        self.regno = regno
        self.name = name

    def dispStudentInfo(self):
        print("Register Number: ",self.regno)
        print("Name: ",self.name)
    def __del__(self):
        className = self.__class__.__name__
        print(className," destroyed")

S1 = Student(101,"Ananya")
S1.dispStudentInfo()
del S1
S1.dispStudentInfo()
```

Output

```
Register Number:  101
Name:  Ananya
Student  destroyed
...
NameError: name 'S1' is not defined
```