Python

Python Constructors

Constructors in Python

Prerequisites: Object-Oriented Programming in Python, Object-Oriented Programming in Python | Set 2

Constructors are generally used for instantiating an object. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. In Python the __init__() method is called the constructor and is always called when an object is created.

Syntax of constructor declaration :

```
def __init__(self):

    # body of the constructor
```

Types of constructors :

- **default constructor:** The default constructor is a simple constructor which doesn't accept any arguments. Its definition has only one argument which is a reference to the instance being constructed.
- **parameterized constructor:** constructor with parameters is known as parameterized constructor. The parameterized constructor takes its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

Example of default constructor :

Python3

```python
class GeekforGeeks:

    # default constructor
    def __init__(self):
        self.geek = "GeekforGeeks"

    # a method for printing data members
    def print_Geek(self):
        print(self.geek)

# creating object of the class
obj = GeekforGeeks()

# calling the instance method using the object obj
obj.print_Geek()
```

Output

```
GeekforGeeks
```

Example of the parameterized constructor :

Python3

```python
class Addition:
    first = 0
    second = 0
    answer = 0

    # parameterized constructor
    def __init__(self, f, s):
        self.first = f
        self.second = s

    def display(self):
        print("First number = " + str(self.first))
        print("Second number = " + str(self.second))
        print("Addition of two numbers = " + str(self.answer))

    def calculate(self):
        self.answer = self.first + self.second

# creating object of the class
# this will invoke parameterized constructor
obj1 = Addition(1000, 2000)

# creating second object of same class
obj2 = Addition(10, 20)

# perform Addition on obj1
obj1.calculate()

# perform Addition on obj2
obj2.calculate()

# display result of obj1
obj1.display()

# display result of obj2
obj2.display()
```

Output

```
First number = 1000

Second number = 2000

Addition of two numbers = 3000

First number = 10

Second number = 20

Addition of two numbers = 30
```

Constructor in Python

A constructor in Python is a special type of method which is used to initialize the instance members of the class.

```
class class_name:
    def __init__(self):
        #default Constructor

    def __init__(self,a,b,c):
        #parameterized Constructor
```

Example:

Python

```python
class MyClass:
    def __init__(self, name=None):
        if name is None:
            print('Default constructor called')
        else:
            self.name = name
            print('Parameterized constructor called with name', self.name)

    def method(self):
        if hasattr(self, 'name'):
            print('Method called with name', self.name)
        else:
            print('Method called without a name')
# Create an object of the class using the default constructor
obj1 = MyClass()
# Call a method of the class
obj1.method()
# Create an object of the class using the parameterized constructor
obj2 = MyClass("John")
# Call a method of the class
obj2.method()
```

Output

```
Default constructor called

Method called without a name

('Parameterized constructor called with name', 'John')

('Method called with name', 'John')
```

Explanation:

In this example, we define a class MyClass with both a default constructor and a parameterized constructor. The default constructor checks whether a parameter has been passed in or not, and prints a message to the console accordingly. The parameterized constructor takes in a single parameter name and sets the name attribute of the object to the value of that parameter.

We also define a method method() that checks whether the object has a name attribute or not, and prints a message to the console accordingly.

We create two objects of the class MyClass using both types of constructors. First, we create an object using the default constructor, which prints the message "Default constructor called" to the console. We then call the method() method on this object, which prints the message "Method called without a name" to the console.

Next, we create an object using the parameterized constructor, passing in the name "John". The constructor is called automatically, and the message "Parameterized constructor called with name John" is printed to the console. We then call the method() method on this object, which prints the message "Method called with name John" to the console.

Overall, this example shows how both types of constructors can be implemented in a single class in Python.

Advantages of using constructors in Python:

- **Initialization of objects**: Constructors are used to initialize the objects of a class. They allow you to set default values for attributes or properties, and also allow you to initialize the object with custom data.
- **Easy to implement:** Constructors are easy to implement in Python, and can be defined using the __init__() method.
- Better readability: Constructors improve the readability of the code by making it clear what values are being initialized and how they are being initialized.
- **Encapsulation**: Constructors can be used to enforce encapsulation, by ensuring that the object's attributes are initialized correctly and in a controlled manner.

Disadvantages of using constructors in Python:

- **Overloading not supported:** Unlike other object-oriented languages, Python does not support method overloading. This means that you cannot have multiple constructors with different parameters in a single class.
- **Limited functionality:** Constructors in Python are limited in their functionality compared to constructors in other programming languages. For example, Python does not have constructors with access modifiers like public, private or protected.
- **Constructors may be unnecessary:** In some cases, constructors may not be necessary, as the default values of attributes may be sufficient. In these cases, using a constructor may add unnecessary complexity to the code.

Overall, constructors in Python can be useful for initializing objects and enforcing encapsulation. However, they may not always be necessary and are limited in their functionality compared to constructors in other programming languages.