Python - Method Overloading

Method overloading is a feature of object-oriented programming where a class can have multiple methods with the same name but different parameters. To overload method, we must change the number of parameters or the type of parameters, or both.

Method Overloading in Python

Unlike other programming languages like Java, C++, and C#, Python does not support the feature of method overloading by default. However, there are alternative ways to achieve it.

Example

If you define a method multiple times as shown in the below code, the last definition will override the previous ones. Therefore, this way of achieving method overloading in Python generates error.

Open Compiler

```
class example:
   def add(self, a, b):
      x = a+b
      return x
   def add(self, a, b, c):
      x = a+b+c
      return x

obj = example()

print (obj.add(10,20,30))
print (obj.add(10,20))
```

The first call to add() method with three arguments is successful. However, calling add() method with two arguments as defined in the class fails.

```
60
Traceback (most recent call last):
 File "C:\Users\user\example.py", line 12, in <module>
  print (obj.add(10,20))
```

```
        ^^^^^^^^^^^^^^^
TypeError: example.add() missing 1 required positional argument: 'c'
```

The output tells you that Python considers only the latest definition of add() method, discarding the earlier definitions.

To simulate method overloading, we can use a workaround by defining default value to method arguments as None, so that it can be used with one, two or three arguments.

Example

The below example shows how to achieve method overloading in Python −

```
class example:
   def add(self, a = None, b = None, c = None):
      x=0
      if a !=None and b != None and c != None:
         x = a+b+c
      elif a !=None and b != None and c == None:
         x = a+b
      return x

obj = example()

print (obj.add(10,20,30))
print (obj.add(10,20))
```

It will produce the following **output** −

```
60
30
```

With this workaround, we are able to incorporate method overloading in Python class.

Implement Method Overloading Using MultipleDispatch

Python's standard library doesn't have any other provision for implementing method overloading. However, we can use a dispatch function from a third-party module named **MultipleDispatch** for this purpose.

First, you need to install the **Multipledispatch** module using the following command −

```
pip install multipledispatch
```

This module has a **@dispatch** decorator. It takes the number of arguments to be passed to the method to be overloaded. Define multiple copies of add() method with @dispatch decorator as below −

Example

In this example, we are using multipledispatch to overload a method in Python.

```
from multipledispatch import dispatch
class example:
   @dispatch(int, int)
   def add(self, a, b):
      x = a+b
      return x
   @dispatch(int, int, int)
   def add(self, a, b, c):
      x = a+b+c
      return x

obj = example()

print (obj.add(10,20,30))
print (obj.add(10,20))
```

Output

```
60
30
```