Python - Sobrecarga de Método

A sobrecarga de métodos é um recurso importante da programação orientada a objetos. As linguagens Java, C++, C# suportam sobrecarga de métodos, mas em Python não é possível realizar sobrecarga de métodos.

Quando você tem uma classe com método de um nome definido mais de um, mas com diferentes tipos de argumento e/ou tipo de retorno, é um caso de sobrecarga de método. Python não oferece suporte a este mecanismo, como mostra o código a seguir -

Exemplo

```
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))
```

Saída

A primeira chamada ao método add() com três argumentos foi bem-sucedida. No entanto, chamar o método add() com dois argumentos conforme definido na classe falha.

```
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'
```

A saída informa que Python considera apenas a definição mais recente do método add(), descartando as definições anteriores.

Para simular a sobrecarga do método, podemos usar uma solução alternativa definindo o valor padrão para os argumentos do método como None, para que possa ser usado com um,

dois ou três argumentos.

Exemplo

```
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))
```

Ele produzirá a seguinte **saída** -

```
60
30
```

Com esta solução alternativa, podemos incorporar a sobrecarga de métodos na classe Python.

A biblioteca padrão do Python não possui nenhuma outra provisão para implementar a sobrecarga de métodos. No entanto, podemos usar a função de despacho de um módulo de terceiros chamado MultipleDispatch para essa finalidade.

Primeiro, você precisa instalar o módulo Multipledispatch.

```
pip install multipledispatch
```

Este módulo possui um decorador @dispatch. Leva o número de argumentos a serem passados para o método a ser sobrecarregado. Defina várias cópias do método add() com o decorador @dispatch conforme abaixo -

Exemplo

```
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))
```

Saída

```
60
30
```