Python - Substituição de método

Você sempre pode substituir os métodos da classe pai. Uma razão para substituir os métodos pai é que você pode querer funcionalidades especiais ou diferentes em sua subclasse.

Exemplo

```
class Parent: # define parent class
   def myMethod(self):
      print ('Calling parent method')

class Child(Parent): # define child class
   def myMethod(self):
      print ('Calling child method')

c = Child() # instance of child
c.myMethod() # child calls overridden method
```

Quando o código acima é executado, ele produz a seguinte **saída** -

```
Calling child method
```

Para entender a herança em Python, tomemos outro exemplo. Usamos a seguinte classe Employee como classe pai -

```
class Employee:
   def __init__(self,nm, sal):
      self.name=nm
      self.salary=sal
   def getName(self):
      return self.name
   def getSalary(self):
      return self.salary
```

A seguir, definimos uma classe SalesOfficer que usa Employee como classe pai. Ele herda as variáveis de instância nome e salário do pai. Além disso, a classe filha possui mais um incentivo variável de instância.

Usaremos a função integrada super() que retorna a referência da classe pai e chama o construtor pai dentro do método __init__() do construtor filho.

```
class SalesOfficer(Employee):
   def __init__(self,nm, sal, inc):
      super().__init__(nm,sal)
      self.incnt=inc
   def getSalary(self):
      return self.salary+self.incnt
```

O método getSalary() é substituído para adicionar o incentivo ao salário.

Exemplo

Declare o objeto das classes pai e filho e veja o efeito da substituição. O código completo está abaixo -

```
class Employee:
   def __init__(self,nm, sal):
      self.name=nm
      self.salary=sal
   def getName(self):
      return self.name
   def getSalary(self):
      return self.salary

class SalesOfficer(Employee):
   def __init__(self,nm, sal, inc):
      super().__init__(nm,sal)
      self.incnt=inc
   def getSalary(self):
      return self.salary+self.incnt

e1=Employee("Rajesh", 9000)
print ("Total salary for {} is Rs {}".format(e1.getName(),e1.getSalary()))
s1=SalesOfficer('Kiran', 10000, 1000)
print ("Total salary for {} is Rs {}".format(s1.getName(),s1.getSalary()))
```

Quando você executa este código, ele produzirá a seguinte **saída** -

```
Total salary for Rajesh is Rs 9000
Total salary for Kiran is Rs 11000
```

Métodos básicos substituíveis

A tabela a seguir lista algumas funcionalidades genéricas da classe de objeto, que é a classe pai de todas as classes Python. Você pode substituir esses métodos em sua própria classe -

Sr. Não	Método, descrição e amostra de chamada
1	**__init__ (self [,args...])** Construtor (com quaisquer argumentos opcionais) Exemplo de chamada: obj = className(args)
2	**__del__(próprio)** Destruidor, exclui um objeto Exemplo de chamada: del obj
3	**__repr__(próprio)** Representação de string avaliável Exemplo de chamada: repr(obj)
4	**__str__(próprio)** Representação de string imprimível Exemplo de chamada: str(obj)