Python - Herança

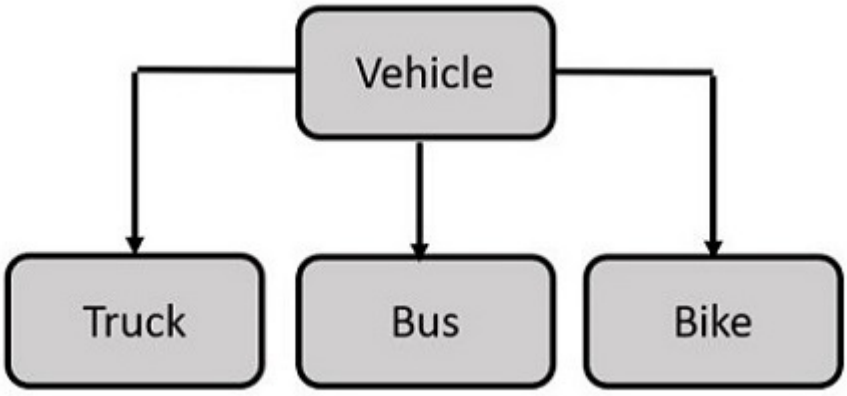
A herança é uma das características mais importantes da metodologia de programação orientada a objetos. É mais frequentemente usado no processo de desenvolvimento de software usando muitas linguagens como Java, PHP, Python, etc.

Em vez de começar do zero, você pode criar uma classe derivando-a de uma classe pré-existente, listando a classe pai entre parênteses após o nome da nova classe.

Em vez de começar do zero, você pode criar uma classe derivando-a de uma classe pré-existente, listando a classe pai entre parênteses após o nome da nova classe.

Se você tiver que projetar uma nova classe cuja maioria dos atributos já esteja bem definida em uma classe existente, então por que redefini-los? A herança permite que os recursos da classe existente sejam reutilizados e, se necessário, estendidos para projetar uma nova classe.

A herança entra em cena quando uma nova classe possui um relacionamento 'IS A' com uma classe existente. Carro É um veículo. Ônibus É um veículo; A bicicleta também é um veículo. Veículo aqui é a classe pai, enquanto carro, ônibus e bicicleta são as classes filhas.



Sintaxe

As classes derivadas são declaradas de forma muito semelhante à classe pai; no entanto, uma lista de classes base das quais herdar é fornecida após o nome da classe -

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
   'Optional class documentation string'
   class_suite
```

Exemplo

```
class Parent: # define parent class
   def __init__(self):
      self.attr = 100
      print ("Calling parent constructor")
   def parentMethod(self):
      print ('Calling parent method')
   def set_attr(self, attr):
      self.attr = attr

   def get_attr(self):
      print ("Parent attribute :", self.attr)

class Child(Parent): # define child class
   def __init__(self):
      print ("Calling child constructor")

   def childMethod(self):
      print ('Calling child method')

c = Child()      # instance of child
c.childMethod()  # child calls its method
c.parentMethod() # calls parent's method
c.set_attr(200)  # again call parent's method
c.get_attr()     # again call parent's method
```

Saída

```
Calling child constructor
Calling child method
Calling parent method
Parent attribute : 200
```

Python - Herança Múltipla

A herança múltipla em Python permite construir uma classe baseada em mais de uma classe pai. A classe Child herda, portanto, os atributos e métodos de todos os pais. O filho pode substituir métodos herdados de qualquer pai.

Sintaxe

```
class parent1:
   #statements
```

```
class parent2:
   #statements

class child(parent1, parent2):
   #statements
```

A biblioteca padrão do Python possui uma função divmod() integrada que retorna uma tupla de dois itens. O primeiro número é a divisão de dois argumentos, o segundo é o valor mod dos dois operandos.

Exemplo

Este exemplo tenta emular a função divmod(). Definimos divisão e módulo de duas classes e, em seguida, temos uma classe div_mod que os herda.

```
class division:
   def __init__(self, a,b):
      self.n=a
      self.d=b
   def divide(self):
      return self.n/self.d
class modulus:
   def __init__(self, a,b):
      self.n=a
      self.d=b
   def mod_divide(self):
      return self.n%self.d

class div_mod(division,modulus):
   def __init__(self, a,b):
      self.n=a
      self.d=b
   def div_and_mod(self):
      divval=division.divide(self)
      modval=modulus.mod_divide(self)
      return (divval, modval)
```

A classe filha tem um novo método div_and_mod() que chama internamente os métodos divide() e mod_divide() de suas classes herdadas para retornar os valores de divisão e mod.

```
x=div_mod(10,3)
print ("division:",x.divide())
```

```
print ("mod_division:",x.mod_divide())
print ("divmod:",x.div_and_mod())
```

Saída

```
division: 3.3333333333333335
mod_division: 1
divmod: (3.3333333333333335, 1)
```

Ordem de Resolução de Método (MRO)

O termo "ordem de resolução de métodos" está relacionado à herança múltipla em Python. Em Python, a herança pode ser distribuída em mais de um nível. Digamos que A seja o pai de B e B o pai de C. A classe C pode substituir o método herdado ou seu objeto pode invocá-lo conforme definido em seu pai. Então, como o Python encontra o método apropriado para chamar.

Cada Python possui um método mro() que retorna a ordem hierárquica que Python usa para resolver o método a ser chamado. A ordem de resolução é de baixo para cima na ordem de herança.

Em nosso exemplo anterior, a classe div_mod herda as classes de divisão e módulo. Portanto, o método mro retorna a ordem da seguinte forma -

```
[<class '__main__.div_mod'>, <class '__main__.division'>, <class '__main__.modulus'>,
```