Python - Genéricos

Em Python, genéricos é um mecanismo com o qual você define funções, classes ou métodos que podem operar em vários tipos, mantendo a segurança do tipo. Com a implementação do Generics enable é possível escrever código reutilizável que pode ser utilizado com diferentes tipos de dados. Ele garante a promoção da flexibilidade do código e da correção do tipo.

Genéricos em Python são implementados usando dicas de tipo. Este recurso foi introduzido no Python a partir da versão 3.5.

Normalmente, você não precisa declarar um tipo de variável. O tipo é determinado dinamicamente pelo valor atribuído a ele. O interpretador do Python não realiza verificações de tipo e, portanto, pode gerar exceções de tempo de execução.

O novo recurso de dicas de tipo do Python ajuda a informar ao usuário o tipo esperado dos parâmetros a serem passados.

As dicas de tipo permitem especificar os tipos esperados de variáveis, argumentos de função e valores de retorno. Os genéricos ampliam esse recurso introduzindo variáveis de tipo, que representam tipos genéricos que podem ser substituídos por tipos específicos ao usar a função ou classe genérica.

Exemplo 1

Vejamos o exemplo a seguir que define uma função genérica -

```
from typing import List, TypeVar, Generic
T = TypeVar('T')
def reverse(items: List[T]) -> List[T]:
   return items[::-1]
```

Aqui, definimos uma função genérica chamada 'reversa'. A função recebe uma lista ('List[T]') como argumento e retorna uma lista do mesmo tipo. A variável de tipo 'T' representa o tipo genérico, que será substituído por um tipo específico quando a função for usada.

Exemplo 2

A função função reverse() é chamada com diferentes tipos de dados -

```
numbers = [1, 2, 3, 4, 5]
reversed_numbers = reverse(numbers)
print(reversed_numbers)
```

```
fruits = ['apple', 'banana', 'cherry']
reversed_fruits = reverse(fruits)
print(reversed_fruits)
```

Ele produzirá a seguinte **saída** -

```
[5, 4, 3, 2, 1]
['cherry', 'banana', 'apple']
```

Exemplo 3

O exemplo a seguir usa genéricos com uma classe genérica -

```
from typing import List, TypeVar, Generic
T = TypeVar('T')
class Box(Generic[T]):
   def __init__(self, item: T):
      self.item = item
   def get_item(self) -> T:
      return self.item
Let us create objects of the above generic class with int and str type
box1 = Box(42)
print(box1.get_item())

box2 = Box('Hello')
print(box2.get_item())
```

Ele produzirá a seguinte **saída** -

```
42
Hello
```