100 XP

Usar um número variável de argumentos no Python

5 minutos

No Python, você pode usar qualquer número de argumentos e argumentos de palavra-chave sem declarar nenhum deles. Essa capacidade é útil quando uma função pode receber um número desconhecido de entradas.

Número variável de argumentos

Os argumentos das funções são necessários. Mas quando você estiver usando um número variável de argumentos, a função permitirá que qualquer número de entradas (incluindo `0`) seja passado. A sintaxe para usar um número variável de argumentos é inserir como prefixo um asterisco (`*`) antes do nome do argumento.

A seguinte função imprime os argumentos recebidos:

Python

```
def variable_length(*args):
    print(args)
```

> **Observação**
>
> Não é necessário chamar o número variável de argumentos `args`. Você pode usar qualquer nome de variável válido. Embora seja comum ver `*args` ou `*a`, você deve tentar usar a mesma convenção em todo o projeto.

Nesse caso, `*args` está instruindo a função para aceitar qualquer número de argumentos (inclusive `0`). Dentro da função, agora `args` está disponível como a variável que contém todos os argumentos como uma tupla. Experimente a função passando qualquer número ou tipo de argumentos:

Python

```
variable_length()
()
```

```
variable_length("one", "two")
('one', 'two')
variable_length(None)
(None,)
```

Como você pode ver, não há restrição quanto ao número ou aos tipos de argumentos passados.

Um foguete passa por várias etapas antes do lançamento. Dependendo das tarefas ou de eventuais atrasos, essas etapas podem levar mais tempo do que o planejado. Vamos criar uma função de comprimento variável que pode calcular quantos minutos até o lançamento, considerando quanto tempo cada etapa vai levar:

Python

```python
def sequence_time(*args):
    total_minutes = sum(args)
    if total_minutes < 60:
        return f"Total time to launch is {total_minutes} minutes"
    else:
        return f"Total time to launch is {total_minutes/60} hours"
```

Experimente a função passando qualquer número de minutos:

Python

```python
sequence_time(4, 14, 18)
```

Output

```
Total time to launch is 36 minutes.
```

Python

```python
sequence_time(4, 14, 48)
```

Output

```
Total time to launch is 1.1 hours.
```

Observação

Quando você usa argumentos variáveis, os valores não são mais atribuídos a nomes de variáveis. Todos os valores fazem parte do nome de variável *abrangente* que usa o asterisco (`args` nestes exemplos).

Número variável de argumentos de palavra-chave

Para que uma função aceite qualquer número de argumentos de palavra-chave, use uma sintaxe semelhante. Nesse caso, são necessários dois asteriscos:

Python

```
def variable_length(**kwargs):
    print(kwargs)
```

Experimente a função de exemplo, que imprime os nomes e valores passados como `kwargs`:

Python

```
variable_length(tanks=1, day="Wednesday", pilots=3)
{'tanks': 1, 'day': 'Wednesday', 'pilots': 3}
```

Se você já estiver familiarizado com os dicionários do Python, notará que os argumentos de palavra-chave de comprimento variável são atribuídos como um dicionário. Para interagir com as variáveis e os valores, use as mesmas operações que um dicionário.

Observação

Assim como acontece com os argumentos comuns de número variável, não é necessário usar `kwargs` quando você está usando um número variável de argumentos de palavra-chave. Você pode usar qualquer nome de variável válido. Embora seja comum ver `**kwargs` ou `**kw`, você deve tentar usar a mesma convenção em todo o projeto.

Nessa função, vamos usar um número variável de argumentos de palavra-chave para relatar os astronautas atribuídos à missão. Como essa função permite qualquer número de argumentos de palavra-chave, ela pode ser reutilizada, independentemente do número de astronautas atribuídos:

Python

```
def crew_members(**kwargs):
    print(f"{len(kwargs)} astronauts assigned for this mission:")
```

```
    for title, name in kwargs.items():
        print(f"{title}: {name}")
```

Experimente com a equipe da Apollo 11:

Python

```
crew_members(captain="Neil Armstrong", pilot="Buzz Aldrin", command_pilot="Michael Collins")
3 astronauts assigned for this mission:
captain: Neil Armstrong
pilot: Buzz Aldrin
command_pilot: Michael Collins
```

Como você pode passar qualquer combinação de argumentos de palavra-chave, certifique-se de evitar palavras-chaves repetidas. Palavras-chave repetidas resultarão em um erro:

Python

```
crew_members(captain="Neil Armstrong", pilot="Buzz Aldrin", pilot="Michael Collins")
  File "<stdin>", line 1
SyntaxError: keyword argument repeated: pilot
```

Unidade seguinte: Exercício – Trabalhar com argumentos de palavra-chave

Continuar >