100 XP

Usar argumentos de função no Python

3 minutos

Agora que você sabe como criar uma função sem entradas, a próxima etapa é criar funções que exigem argumento(s). O uso de argumentos torna as funções mais flexíveis, pois elas podem fazer mais e condicionar as funções que elas executam.

Como exigir argumentos

Se você estiver realizando um piloto de um foguete, uma função sem entradas obrigatórias seria como um computador com um botão para informar o tempo. Se você pressionar o botão, uma voz computadorizada informará o tempo. Mas uma entrada obrigatória pode ser o destino, por exemplo, para calcular a distância do percurso. As entradas obrigatórias são chamadas de *argumentos* da função.

Para exigir um argumento, coloque-o entre parênteses:

Python

```
def distance_from_earth(destination):
    if destination == "Moon":
        return "238,855"
    else:
        return "Unable to compute to that destination"
```

Tente chamar a função `distance_from_earth()` sem argumentos:

Python

```
distance_from_earth()
```

Output

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: distance_from_earth() missing 1 required positional argument:
'destination'
```

O Python gera `TypeError` com uma mensagem de erro que diz que a função requer um argumento chamado `destination`. Se o computador do foguete precisar computar a distância do percurso até um destino, ele deverá solicitar o destino como obrigatório. O código de exemplo tem dois caminhos como resposta: um é a Lua e o outro é qualquer outro lugar. Use a Lua como entrada para obter uma resposta:

Python

```
distance_from_earth("Moon")
```

Output

```
238,855
```

Como há uma condição *abrangente*, tente usar qualquer outra cadeia de caracteres como destino para verificar o comportamento:

Python

```
distance_from_earth("Saturn")
```

Output

```
Unable to compute to that destination
```

Vários argumentos obrigatórios

Para usar vários argumentos, você deve separá-los por vírgula. Você criará uma função capaz de calcular quantos dias levará para alcançar um destino com base na distância e na velocidade constante empregada:

Python

```
def days_to_complete(distance, speed):
    hours = distance/speed
    return hours/24
```

Agora, use a distância da Terra até a Lua para calcular quantos dias levaria para chegar à Lua no limite de velocidade típico de uma rodovia comum de 75 milhas por hora:

Python

```
days_to_complete(238855, 75)
```

Output

```
132.69722222222222
```

Funções como argumentos

Você pode usar o valor da função `days_to_complete()` e atribuí-lo a uma variável, depois passá-lo para `round()` (uma função interna que arredonda para o valor para o número inteiro mais próximo) a fim de obter um número inteiro:

Python

```
total_days = days_to_complete(238855, 75)
round(total_days)
```

Output

```
133
```

No entanto, um padrão útil é passar diretamente funções para outras funções em vez de atribuir os valores retornados:

Python

```
round(days_to_complete(238855, 75))
```

Output

```
133
```

Dica

Embora a passagem de funções diretamente para outras funções como entrada seja útil, há possibilidade dessa prática reduzir a legibilidade do código. Esse padrão é especialmente problemático quando as funções exigem muitos argumentos.

Unidade seguinte: Exercício – Usar funções no Python

Continuar >