100 XP

Métodos de cadeia de caracteres no Python

3 minutos

Cadeias de caracteres são um dos tipos de método mais comuns no Python. Muitas vezes, você precisará manipulá-las para extrair informações ou ajustar a um determinado formato. O Python inclui vários métodos de cadeia de caracteres projetados para fazer as transformações mais comuns e úteis.

Os métodos de cadeia de caracteres fazem parte do tipo `str`. Isso significa que os métodos se aplicam a variáveis de cadeia de caracteres e também diretamente a partes da cadeia de caracteres. Por exemplo, o método `.title()` retorna a cadeia de caracteres com iniciais em maiúsculas e pode ser usado diretamente com uma cadeia de caracteres:

Python

```
print("temperatures and facts about the moon".title())
```

Saída: `Temperatures And Facts About The Moon`

Os mesmos comportamento e uso ocorrem em uma variável:

Python

```
heading = "temperatures and facts about the moon"
heading_upper = heading.title()
print(heading_upper)
```

Dividir uma cadeia de caracteres

Um método comum de cadeia de caracteres é `.split()`. Sem argumentos, o método separa a cadeia de caracteres em cada espaço. Isso cria uma lista composta por cada palavra ou número existente separado(a) por espaço:

Python

```
temperatures = "Daylight: 260 F Nighttime: -280 F"
temperatures_list = temperatures.split()
print(temperatures_list)
```

Saída: `['Daylight:', '260', 'F', 'Nighttime:', '-280', 'F']`

Neste exemplo, você está lidando com várias linhas, portanto, o caractere de nova linha (implícito) pode ser usado para dividir a cadeia de caracteres ao final de cada linha, criando linhas individuais:

Python

```
temperatures = "Daylight: 260 F\n Nighttime: -280 F"
temperatures_list = temperatures.split('\n')
print(temperatures_list)
```

Saída: `['Daylight: 260 F', 'Nighttime: -280 F']`

Esse tipo de divisão torna-se útil quando você precisa de um loop para processar ou extrair informações ou quando você está carregando dados de um arquivo de texto.

Pesquisar uma cadeia de caracteres

Alguns métodos de cadeia de caracteres podem procurar conteúdos antes do processamento, sem usar um loop. Vamos supor que você tem duas frases que discutem as temperaturas em diversos planetas e luas. No entanto, você está interessado apenas em temperaturas

relacionadas à nossa lua. Ou seja, se as frases não estão falando sobre a Lua da Terra, elas não devem ser processadas para extrair informações.

A maneira mais simples de descobrir se uma determinada palavra, caractere ou grupo de caracteres existe em uma cadeia de caracteres é usar o operador `in`:

Python

```
print("Moon" in "This text will describe facts and challenges with space travel")
```

Saída: `False`

Python

```
print("Moon" in "This text will describe facts about the Moon")
```

Saída: `True`

Uma abordagem para localizar a posição de uma palavra específica em uma cadeia de caracteres é usar o método `.find()`:

Python

```
temperatures = """Saturn has a daytime temperature of -170 degrees Celsius, while Mars has -28 Celsius."""
print(temperatures.find("Moon"))
```

Saída: `-1`

O método `.find()` retorna um `-1` quando a palavra não é encontrada ou retorna o índice (o número que representa o local na cadeia de caracteres). É assim que ele se comportaria se você estivesse pesquisando a palavra *Marte*:

Python

```
temperatures = """Saturn has a daytime temperature of -170 degrees Celsius, while Mars has -28 Celsius."""
print(temperatures.find("Mars"))
```

Saída: `64`

`64` é a posição em que `"Mars"` aparece na cadeia de caracteres.

Outra maneira de pesquisar conteúdo é usar o método `.count()`, que retorna o número total de ocorrências de uma determinada palavra em uma cadeia de caracteres:

Python

```
temperatures = """Saturn has a daytime temperature of -170 degrees Celsius, while Mars has -28 Celsius."""
print(temperatures.count("Mars"))
print(temperatures.count("Moon"))
```

Output

```
1
0
```

As cadeias de caracteres em Python diferenciam letras minúsculas de minúsculas, o que significa que *Lua* e *lua* são consideradas palavras distintas. Para fazer uma comparação sem diferenciar maiúsculas de minúsculas, você pode converter uma das cadeias de caracteres em letras minúsculas usando o método `.lower()`:

Python

```
print("The Moon And The Earth".lower())
```

Saída: `the moon and the earth`

Semelhante ao método `.lower()`, as cadeias de caracteres têm um método `.upper()` que faz o oposto, convertendo cada caractere em letras maiúsculas:

Python

```
print("The Moon And The Earth".upper())
```

Saída: `THE MOON AND THE EARTH`

Dica

Quando você está pesquisando e verificando conteúdos, a abordagem mais adequada é converter a cadeia de caracteres para minúsculas a fim de evitar que isso atrapalhe nas correspondências encontradas. Por exemplo, se você estiver contando o número de vezes que a palavra *o* aparece, o método não contará as ocorrências de *O*, embora ambos correspondam à mesma palavra. Você pode usar o método `.lower()` para alterar todos os caracteres para minúsculos.

Verificar conteúdo

Há ocasiões em que você terá que processar textos para extrair informações apresentadas de modo irregular. Por exemplo, a seguinte cadeia de caracteres é mais simples de processar do que um parágrafo não estruturado:

Python

```
temperatures = "Mars Average Temperature: -60 C"
```

Para extrair a temperatura média em Marte, você pode fazer bem com os seguintes métodos:

Python

```
temperatures = "Mars Average Temperature: -60 C"
parts = temperatures.split(':')
print(parts)
print(parts[-1])
```

Output

```
['Mars average temperature', ' -60 C']
' -60 C'
```

O código anterior confia que tudo o que vem após os dois-pontos (`:`) corresponde a dados de temperatura. A cadeia de caracteres é dividida em `:`, o que produz uma lista de dois itens. Usar `[-1]` na lista retorna o último item, que é a temperatura neste exemplo.

Se o texto for irregular, você não poderá usar os mesmos métodos de divisão para obter o valor. Você deve fazer um loop sobre os itens e verificar se os valores são de um determinado tipo. O Python tem métodos que ajudam a verificar o tipo de cadeia de caracteres:

Python

```
mars_temperature = "The highest temperature on Mars is about 30 C"
for item in mars_temperature.split():
    if item.isnumeric():
        print(item)
```

Saída: `30`

Como o método `.isnumeric()`, `.isdecimal()` pode verificar se há cadeias de caracteres que se parecem com decimais.

Importante

Pode ser surpreendente descobrir que `"-70".isnumeric()` retornaria `False`. Isso ocorre porque todos os caracteres da cadeia de caracteres precisariam ser numéricos e o traço (`-`) não é. Se você precisar verificar números negativos em uma cadeia de caracteres,

o método `.isnumeric()` não funcionará.

Há validações adicionais que você pode aplicar nas cadeias de caracteres para checar valores. Para números negativos, o traço aparece como um prefixo do número, podendo ser detectado com o método `.startswith()`:

```python
print("-60".startswith('-'))
```

Saída: `True`

Da mesma forma, o método `.endswith()` ajuda a verificar o último caractere de uma cadeia de caracteres:

```python
if "30 C".endswith("C"):
print("This temperature is in Celsius")
```

Saída: `This temperature is in Celsius`

Texto de transformação

Há outros métodos que ajudam em situações em que o texto precisa ser transformado em outra coisa. Até agora, você viu cadeias de caracteres que podem usar *C* para *Celsius* e *F* para *Fahrenheit*. Você pode usar o método `.replace()` para encontrar e substituir ocorrências de um caractere ou grupo de caracteres:

```python
print("Saturn has a daytime temperature of -170 degrees Celsius, while Mars has -28 Celsius.".replace("Celsius", "C"))
```

Saída: `Saturn has a daytime temperature of -170 degrees C, while Mars has -28 C.`

Conforme mencionado anteriormente, `.lower()` é uma excelente maneira de normalizar o texto para fazer uma pesquisa que não faça distinção entre maiúsculas e minúsculas. Vamos verificar rapidamente se algum texto discute temperaturas:

```python
text = "Temperatures on the Moon can vary wildly."
print("temperatures" in text)
```

Saída: `False`

```python
text = "Temperatures on the Moon can vary wildly."
print("temperatures" in text.lower())
```

Saída: `True`

Talvez você não precise fazer a verificação que não faz distinção entre maiúsculas e minúsculas o tempo todo, mas converter todas as letras para minúsculas é uma boa estratégia quando o texto mistura letras maiúsculas e minúsculas.

Depois de dividir o texto e realizar as transformações, talvez seja necessário reunir todas as partes novamente. Assim como o método `.split()` pode separar caracteres, o método `.join()` pode uni-los novamente.

O método `.join()` requer um objeto iterável (como uma lista do Python) como um argumento, portanto, seu uso é diferente de outros métodos de cadeia de caracteres:

```python
moon_facts = ["The Moon is drifting away from the Earth.", "On average, the Moon is moving about 4cm every year."]
```

```
print(' '.join(moon_facts))
```

Saída: `The Moon is drifting away from the Earth. On average, the Moon is moving about 4cm every year.`

Neste exemplo, `' '` é usado para colocar todos os itens na lista.

Unidade seguinte: Exercício – Transformar texto usando métodos de cadeia de caracteres

Continuar >