Manipulação de Strings

Fundamentos de Programação em Python

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizagem

Ao final desta aula, você será capaz de:

- Compreender os conceitos fundamentais de strings em Python.
- Acessar e manipular substrings utilizando indexação, fatiamento e métodos de formatação.
- Percorrer strings e buscar informações.

Conceitos Fundamentais

Diferente de `int`, `float` e `bool`, strings (`str`) são um **tipo de dado composto**, pois são formadas de pedaços menores: caracteres.

> Uma **string** é uma sequência de caracteres.

Strings podem ser definidas de várias formas:

```
s1 = 'Texto com aspas simples'
s2 = "Texto com aspas duplas"
s3 = '''Texto de múltiplas linhas'''
```

Caracteres de Escape

Caracteres precedidos por `\` que representam um significado especial.

Importância: Essenciais para formatar strings complexas, lidar com dados de arquivos e construir expressões regulares.

Caractere de Escape	Significado	Exemplo	Resultado
`\n`	Nova Linha	`print("Olá\nMundo")`	Olá Mundo
`\t`	Tabulação Horizontal	`print("Nome:\tSobrenome")`	`Nome:  Sobrenome`
`\\`	Barra Invertida Literal	`print("C:\\arquivo.txt")`	C:\arquivo.txt
`\'`	Aspas Simples Literais	`print("what\'s up)"`	what's up

Comprimento de um String

Usamos a função `len()` para aferir o comprimento de uma string (o número de caracteres da string).

> A função `len()` recebe um tipo de dado composto e retorna o seu comprimento.

```
text = "Python"
print(len(text)) # Saída: 6
```

Indexação de Strings em Python

Cada caractere de uma string possui um índice, começando do **0**.

```
            +---+---+---+---+---+---+
   texto <- | P | y | t | h | o | n |
            +---+---+---+---+---+---+
              0   1   2   3   4   5    (Índices Positivos)
            ---------------------------
             -6  -5  -4  -3  -2  -1    (Índices Negativos)
```

```python
texto = "Python"
print(texto[0])   # P
print(texto[3])   # h
print(texto[-1])  # n
```

Slicing em Strings

Fatiamento permite extrair partes de uma string. Sintaxe: `string[start:stop]`

```
texto = "Python"
print(texto[0:4])  # Pyth
print(texto[:3])   # Pyt
print(texto[2:])   # thon
```

Para praticar:

1. Extraia `"yt"` da string `"Python"`.
2. Use slicing para exibir os últimos 3 caracteres de uma string digitada pelo usuário.

Strings são imutáveis

Uma vez criadas, seus valores não podem ser modificados diretamente.

```
frase = "Python"
frase[0] = "J"
print(frase)  # Saída: Jython ???
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[48], line 2
      1 frase = "Python"
----> 2 frase[0] = 'J'
      3 print(frase)

TypeError: 'str' object does not support item assignment
```

Strings são imutáveis (cont.)

Uma vez criadas, seus valores não podem ser modificados diretamente.

```
frase = "Python"
frase[0] = "J"
print(frase)  # Saída: Jython ???
```

Para modificar um caractere da string, você pode, por exemplo, fazer o seguinte:

```
frase = "Python"
nova_frase = "J" + frase[1:]
print(frase)  # Saída: Jython
```

Métodos de Manipulação de Strings

Python fornece diversos métodos úteis para strings:

```
s = "Python é incrível!"
print(s.lower())  # python é incrível!
print(s.upper())  # PYTHON É INCRÍVEL!
print(s.strip())  # Remove espaços extras
```

Operações com Strings

Concatenação (`+`)

```
nome = "João"
sobrenome = "Silva"
nome_completo = nome + " " + sobrenome
print(nome_completo)  # João Silva
```

Repetição (`*`)

```
print("Python! " * 3)  # Python! Python! Python!
```

Métodos úteis

`split()` e `join()`

```
frase = "João,Maria,Ana"
nomes = frase.split(",")  # ['João', 'Maria', 'Ana']
print(" e ".join(nomes))  # João e Maria e Ana
```

`replace()`

```
s = "Eu gosto de chocolate."
print(s.replace("chocolate", "sorvete"))  # Eu gosto de sorvete.
```

f-Strings (Formatação)

```
nome = "Alice"
idade = 25
print(f"Meu nome é {nome} e eu tenho {idade} anos.")
# Saída: Meu nome é Alice e eu tenho 25 anos.
```

Percorrendo Strings com Loops

Podemos iterar sobre uma string com `for`:

```
frase = "Python"
for letra in frase:
    print(letra)
```

Verificando substrings

Usamos o operador `in` para verificar se uma substring (caractere único ou um conjunto de caracteres) está presente em uma string.

```
texto = "Bom dia"
x = "dia" in texto
print(x)
```

Conclusão

Nesta aula, aprendemos:

- Como criar, acessar e modificar strings.
- Métodos úteis para manipulação de strings.
- Operações como concatenação, repetição e formatação.

Dúvidas e Discussão