MANIPULAÇÃO DE STRINGS

IFBA – Campus Ilhéus

Curso: Back End com Python

Objetivo da Aula

- Trabalhar com textos de maneira eficiente.

Operações básicas com strings

Operações básicas com strings

1. Concatenar Strings
2. Repetir Strings
3. Acessar Caracteres
4. Fatiar Strings

Operações básicas com strings

- Concatenar Strings
  - A concatenação é a operação de juntar duas ou mais strings.
  - Exemplo:

```
nome = "Maria"
sobrenome = "Silva"
nome_completo = nome + " " + sobrenome
print(nome_completo)  # Saída: Maria Silva
```

Operações básicas com strings

- Repetir Strings
  - É possível repetir uma string usando o operador *.
  - Exemplo:

```
frase = "Python "  *  3
print(frase)  # Saída: Python Python Python
```

Operações básicas com strings

- Acessar caracteres
  - Os caracteres individuais em uma string são acessados usando índices (índices começam em 0).
  - Exemplo:

```
texto = "Python"
print(texto[0])  # Saída: P (primeiro caractere)
print(texto[-1]) # Saída: n (último caractere)
```

Operações básicas com strings

- Fatiar Strings
  - É possível extrair uma parte da string usando fatiamento (slicing).
  - Exemplo:

```
texto = "Python"
print(texto[0:3])  # Saída: Pyt (do índice 0 até 2)
print(texto[2:])   # Saída: thon (do índice 2 até o final)
print(texto[:4])   # Saída: Pyth (do início até o índice 3)
```

Métodos Úteis para Manipulação de Strings

lower(), upper(), strip(), split(), join()

Função len()

- A função **len()** retorna o tamanho (ou comprimento) de um objeto, ou seja, quantos itens ele possui.
- Em uma **string**, ela retorna quantos caracteres existem, incluindo espaços e símbolos.
- Sintaxe:
  - len(objeto)

Função len()

- Exemplos:

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

mensagem = "Olá, mundo!"
print(len(mensagem))
# Saída: 11
```

Métodos Úteis para Manipulação de Strings

- lower()
  - Transforma todos os caracteres de uma string em minúsculas.
  - Exemplo:

```
texto = "Python é incrível!"
print(texto.lower())  # Saída: python é incrível!
```

Métodos Úteis para Manipulação de Strings

- upper()
  - Transforma todos os caracteres de uma string em maiúsculas.
  - Exemplo:

```
texto = "Python é incrível!"
print(texto.upper())    # Saída: PYTHON É INCRÍVEL!
```

Métodos Úteis para Manipulação de Strings

- strip()
  - Remove espaços em branco no início e no final da string. Também pode remover caracteres específicos, se fornecido.
  - Exemplo:

```
texto = "   Olá, Mundo!   "
print(texto.strip())  # Saída: Olá, Mundo!

texto_com_caracteres = "***Python***"
print(texto_com_caracteres.strip("*"))     # Saída: Python
```

Métodos Úteis para Manipulação de Strings

- split()
  - Divide uma string em uma lista de substrings com base em um delimitador (por padrão, o espaço).
  - Exemplo:

```
texto = "Python é incrível!"
palavras = texto.split()  # Divide por espaços
print(palavras)  # Saída: ['Python', 'é', 'incrível!']
```

Métodos Úteis para Manipulação de Strings

- split() - continuação
  - Também é possível especificar outro delimitador:
  - Exemplo:

```
data = "2025-08-07"
partes = data.split("-")  # Divide por '-'
print(partes)  # Saída: ['2025', '08', '07']
```

Métodos Úteis para Manipulação de Strings

- join()
  - Junta uma lista de strings em uma única string, usando um separador.
  - Exemplo:

```
palavras = ['Python', 'é', 'incrível!']
resultado = "  ".join(palavras)
print(resultado)  # Saída: Python é incrível!
```

Métodos Úteis para Manipulação de Strings

- join() - continuação
  - Também é possível escolher outro delimitador:
  - Exemplo:

```
resultado_com_virgula = ",".join(palavras)
print(resultado_com_virgula)  # Saída: Python,é,incrível!
```

Formatação de strings

Formatação de strings

- A formatação de strings permite que você insira variáveis em uma string de maneira legível e controlada.

Formatação de strings

- Formatando com o método format()
  - Permite inserir variáveis dentro de uma string com {} como marcadores de posição.
  - Exemplo:

```
nome = "Maria"
idade = 30
mensagem = "Olá, {}! Você tem {} anos.".format(nome, idade)

print(mensagem)  # Saída: Olá, Maria! Você tem 30 anos.
```

Formatação de strings

- método format() - continuação
  - É possível também referenciar diretamente os parâmetros:

```
mensagem = "Olá, {0}! Você tem {1} anos.".format(nome, idade)

print(mensagem)  # Saída: Olá, Maria! Você tem 30 anos.
```

Formatação de strings

- Formatação com F-Strings (Python 3.6+)
  - As f-strings permitem interpolação de variáveis diretamente na string, usando {}. Isso é mais eficiente e legível.
  - Exemplo:

```
nome = "Maria"
idade = 30
mensagem = f"Olá, {nome}! Você tem {idade} anos."
print(mensagem)  # Saída: Olá, Maria! Você tem 30 anos.
```

Formatação de strings

- Formatação Numérica com F-Strings
  - É possível também formatar números diretamente nas f-strings, por exemplo, para controlar o número de casas decimais.
  - Exemplo:

```
valor = 123.4567

print(f"Valor formatado: {valor:.2f}")
# Saída: Valor formatado: 123.46
```

Exercícios

1. Crie um programa que pergunte o nome e o sobrenome de uma pessoa, depois imprima o nome completo.
2. Peça para o usuário digitar uma palavra e imprima essa palavra 5 vezes, separada por espaços.
3. Peça ao usuário para digitar uma frase com espaços extras no início e no final. Use o método strip() para remover esses espaços e mostre a frase corrigida.

Exercícios

4. Crie um programa que receba uma lista de palavras separadas por vírgula (como uma string) e as converta em uma lista. Depois, junte novamente as palavras usando o símbolo “&” como separador (Use split() e join()).
5. Crie um programa que peça o nome, idade e cidade de uma pessoa e mostre uma frase formatada usando f-strings, como: "Meu nome é [nome], tenho [idade] anos e moro em [cidade]."