



# STRINGS

---

Ruan Pablo Medeiros

# INTRODUÇÃO

---

- O que são Strings?
- Importância das Strings na programação

# CRIANDO STRINGS

---

- Declaração de Strings.
- Utilização de aspas simples ( ' ' ) e duplas ( " " ) para definir uma String.
- Strings Multilinhas.



```
# Exemplos de criação de Strings
```

```
string1 = 'Olá, mundo!'
```

```
string2 = "Python é incrível!"
```

```
string_multilinha = """Esta é uma string  
que abrange múltiplas linhas."""
```

# ACESSANDO CARACTERES EM UMA STRING

---

- Indexação em Python
- Indexação negativa

```
● ● ●  
  
# Exemplos de acesso a caracteres em uma String  
s = "Python"  
print(s[0])    # Saída: 'P'  
print(s[-1])   # Saída: 'n'
```

# OPERAÇÕES BÁSICAS COM STRINGS

---

- Concatenação de strings
- Repetição de Strings
- Comparação de Strings

# OPERAÇÕES BÁSICAS COM STRINGS

---

```
● ● ●  
  
# Exemplos de operações básicas com Strings  
s1 = "Olá"  
s2 = "Mundo"  
concatenacao = s1 + " " + s2    # Concatenação  
repeticao = s1 * 3                # Repetição  
comparacao = (s1 == s2)         # Comparação  
  
print(concatenacao)  # Saída: 'Olá Mundo'  
print(repeticao)      # Saída: 'OláOláOlá'  
print(comparacao)    # Saída: False
```

# MÉTODOS DE MANIPULAÇÃO DE STRINGS

---

- `len()`: Obtendo o comprimento de uma String.
- `upper()`, `lower()`, `capitalize()`: Convertendo entre maiúsculas e minúsculas.
- `strip()`, `lstrip()`, `rstrip()`: Removendo espaços em branco.
- `split()`: Dividindo uma String em substrings.
- `join()`: Juntando substrings em uma única String.
- `replace()`: Substituindo partes de uma String.
- `find()`, `index()`: Encontrando a posição de substrings.

# MÉTODOS DE MANIPULAÇÃO DE STRINGS

```
# Exemplos de métodos de manipulação de Strings
s = "  Olá, mundo!  "
comprimento = len(s)           # Comprimento
maiusculas = s.upper()         # Maiúsculas
minusculas = s.lower()         # Minúsculas
capitalizada = s.strip().capitalize() # Capitalizada e remoção de espaços
em branco
split_str = s.split(",")       # Divisão
joined_str = "-".join(['a', 'b', 'c']) # Junção
substituicao = s.replace("mundo", "Python") # Substituição
posicao = s.find("mundo")       # Encontrar posição

print(comprimento)    # Saída: 15
print(maiusculas)     # Saída: '  OLÁ, MUNDO!  '
print(minusculas)     # Saída: '  olá, mundo!  '
print(capitalizada)   # Saída: 'Olá, mundo!'
print(split_str)      # Saída: ['  Olá', ' mundo!  ']
print(joined_str)     # Saída: 'a-b-c'
print(substituicao)    # Saída: '  Olá, Python!  '
print(posicao)         # Saída: 7
```



# FORMATAÇÃO DE STRINGS

---

- Utilizando f-strings para formatação de Strings.
- Métodos de formatação anteriores (`.format()`).

# FORMATAÇÃO DE STRINGS



```
# Exemplos de formatação de Strings
nome = "Maria"
idade = 30
altura = 1.65

# Usando f-strings
mensagem = f"Olá, meu nome é {nome}, tenho {idade} anos e minha altura é {altura} metros."

# Usando o método .format()
mensagem_format = "Olá, meu nome é {}, tenho {} anos e minha altura é {} metros.".format(nome, idade, altura)

print(mensagem)
print(mensagem_format)
```

# FORMATAÇÃO DE STRINGS

---

- Quando você faz a formatação de uma variável dentro de uma string em Python, especialmente usando formatos literais de string (f-strings) ou o método `.format()`, você pode especificar o tipo de formatação que deseja aplicar à variável.

# FORMATAÇÃO DE STRINGS

---

- `d` ou `i` - Inteiros:
  - `{variavel:d}` ou `{variavel:i}`: Formata a variável como um número inteiro.
- `f` - Ponto flutuante:
  - `{variavel:f}`: Formata a variável como um número de ponto flutuante.
- `e` ou `E` - Notação científica:
  - `{variavel:e}` ou `{variavel:E}`: Formata a variável em notação científica.

# FORMATAÇÃO DE STRINGS

---

- **g** ou **G** - Compactação:
  - **{variavel:g}** ou **{variavel:G}**: Usa a formatação mais compacta, escolhendo entre notação decimal ou notação científica, **dependendo do tamanho da** variável.
- **s** - String:
  - **{variavel:s}**: Formata a variável como uma string.
- **%** - Porcentagem:
  - **{variavel:.2%}**: Formata a variável como um percentual com duas casas decimais.

# CARACTERES DE ESCAPE

---


- Uso de caracteres de escape, como '\n', '\t', '\\', etc.

```
# Exemplos de caracteres de escape
print("Olá\nMundo!")    # Saída: Olá
                        #      Mundo!
print("Python\té\tincrível!") # Saída: Python   é   incrível!
print("Caminho do arquivo: C:\\diretorio\\arquivo.txt") # Saída: Caminho do
arquivo: C:\diretorio\arquivo.txt
```

# CARACTERÍSTICAS IMUTÁVEIS DAS STRINGS

---

- Em Python a estrutura de uma String é parecida como uma lista, que armazena em cada posição um caractere. Porém a String é imutável, ou seja, sem o uso de um método específico não podemos alterar o conteúdo de uma posição.

```
  
# Exemplo de imutabilidade das Strings  
s = "Python"  
# Tentativa de alterar um caractere da String resulta em erro  
s[0] = 'p' # Erro: TypeError: 'str' object does not support item assignment
```

# SLICING EM STRINGS

---

- Na aula anterior aprendemos sobre Slice em listas, o mesmo conceito se aplica para extrair Sub-strings de uma String.

```
• • •  
  
# Exemplo de slicing em Strings  
s = "Python"  
substring = s[2:5]    # Obtendo 'tho'  
print(substring)      # Saída: 'tho'
```



# ITERANDO SOBRE STRINGS

---

- Utilizando loops `for` para iterar sobre os caracteres de uma String.

```
• • •  
  
# Exemplo de iteração sobre Strings  
s = "Python"  
for char in s:  
    print(char)  
  
# Saída:  
# P  
# y  
# t  
# h  
# o  
# n
```

# EXERCÍCIOS

---

1. Uma palavra é palíndromo quando a sua leitura sendo feita da esquerda para direita ou da direita para a esquerda resulta na mesma palavra. Ex: Ana, Renner, Oto, radar... Escreva um programa que peça para o usuário digitar uma frase e diga para o usuário se ela é ou não palíndromo, use como saída as Strings "É palíndromo" e "Não é palíndromo".
2. Escreva um programa onde o usuário digita uma frase e como saída mostre a quantidade de palavras que a frase possui.
3. Escreva um programa que leia separadamente uma frase e dois caracteres, em seguida substitua todas as ocorrências do primeiro caractere lido pelo segundo caractere na frase e mostre na tela a nova frase para o usuário.
4. Escreva um programa que leia uma lista de animais, em seguida mostre essa lista em uma String com os animais separados por vírgula.