



STRINGS

Autoria de [Carolina Soares](#)

STRINGS

Uma string é um tipo de dado que armazena uma sequência de caracteres.

```
texto = 'Olá mundo'
```

```
print(texto)
```

```
print(type(texto))
```

Saída do código:

Olá mundo

<class 'str'>

STRINGS DE DOCUMENTAÇÃO

```
texto = """
```

```
Está é uma String em Python com multiplas  
linhas. O Python me permite guardar uma  
String com várias linhas de caracteres. As  
Strings são textos que podem conter letras e  
números. Além disso, as Strings podem está  
dentro de aspas simples ou duplas."""
```

ÍNDICE DA STRING

nome = "Maria" Saída do código:

print(nome[0]) M

print(nome[1]) a

print(nome[2]) r

print(nome[3]) i

print(nome[4]) a

ÍNDICE DA STRING

| | |
|------------------------------|------------------|
| <code>nome = "Maria"</code> | Saída do código: |
| <code>print(nome[-1])</code> | a |
| <code>print(nome[-2])</code> | i |
| <code>print(nome[-3])</code> | r |
| <code>print(nome[-4])</code> | a |
| <code>print(nome[-5])</code> | M |

INTERVALO DA STRING

| | | | | | | | | | | |
|--------|---|---|---|---|---|---|---|---|---|---|
| índice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| nome | J | o | ã | o | | S | i | l | v | a |

nome[início : fim : pulo]

INTERVALO DA STRING

| | | | | | | | | | | |
|--------|-----|----|----|----|----|----|----|----|----|----|
| índice | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |
| nome | J | o | ã | o | | S | i | l | v | a |

`nome[início : fim : pulo]`

INTERVALO DA STRING

| | |
|----------------------------------|------------------|
| <code>nome = "João Silva"</code> | Saída do código: |
| <code>print(nome[5:])</code> | Silva |
| <code>print(nome[-3:])</code> | lva |
| <code>print(nome[3:8])</code> | o Sil |
| <code>print(nome[-10:4])</code> | João |
| <code>print(nome[:7:2])</code> | Jã i |

CONCATENANDO STRINGS

```
texto1 = 'Aprendendo'  
texto2 = 'Python'  
print(texto1 + ' ' + texto2)
```

Saída do código:

Aprendendo Python

MULTIPLICANDO A STRING

```
linha = '-'
```

```
print(linha * 20)
```

```
print('Bloco de código')
```

```
print(linha * 20)
```

Saída do código:

Bloco de código

OBTER O TAMANHO DA STRING

```
texto = "abc"
```

Saída do código:

```
print(len(texto))
```

3

STRINGS EM ESTRUTURAS DE REPETIÇÃO

```
txt = 'Python'
```

```
for item in txt:
```

```
    print(item)
```

```
txt = 'Python'
```

```
indice = 0
```

```
while indice < len(txt):
```

```
    print(txt[indice])
```

```
    indice += 1
```

Saída do código:

P

y

t

h

o

n

VERIFICAR SE STRING EXISTE EM OUTRA

```
texto = "Tenha um ótimo dia"
if 'dia' in texto:
    print("A sequência de caractere existe na string!")
```

Saída: A sequência de caractere existe na string!

VERIFICAR SE STRING EXISTE EM OUTRA

```
texto = "Tenha um ótimo dia"
if 'ótimo' not in texto:
    print("A palavra 'ótimo' não existe no texto")
else:
    print("Sim, 'ótimo' existe no texto!")
```

Saída: Sim, 'ótimo' existe no texto!

FUNÇÕES DA STRING

```
palavra = "Céu Limpo"
```

```
palavra.upper() # CÉU LIMPO
```

```
palavra.lower() # céu limpo
```

```
palavra.replace("Limpo", "Nublado") # Céu Nublado
```

```
palavra.split(" ") # ['Céu', 'Limpo']
```

```
palavra.startswith("Céu") # True
```

```
palavra.endswith(".") # False
```

CARACTERES DE ESCAPE NA STRING

```
print("Programação em \"Python\"")
```

Saída: Programação em "Python"

```
print("Programação \n em Python")
```

Saída: Programação

em Python

```
print("Programação \t em Python")
```

Saída: Programação em Python

```
print("Programação e\\b\\b Python")
```

Saída: Programação Python

CARACTERES DE ESCAPE

| Sequência de escape | Significado |
|-------------------------------|---|
| <code>\<newline></code> | A barra invertida e a nova linha foram ignoradas |
| <code>\\</code> | Contrabarra (<code>\</code>) |
| <code>\'</code> | Aspas simples (<code>'</code>) |
| <code>\"</code> | Aspas duplas (<code>"</code>) |
| <code>\a</code> | ASCII Bell (BEL) - um sinal audível é emitido |
| <code>\b</code> | ASCII Backspace (BS) - apaga caractere à esquerda |
| <code>\f</code> | ASCII Formfeed (FF) - quebra de página |
| <code>\n</code> | ASCII Linefeed (LF) - quebra de linha |
| <code>\r</code> | ASCII Carriage Return (CR) - retorno de carro |
| <code>\t</code> | ASCII Horizontal Tab (TAB) - tabulação horizontal |
| <code>\v</code> | ASCII Vertical Tab (VT) - tabulação vertical |
| <code>\ooo</code> | Caractere com valor octal <i>ooo</i> |
| <code>\xhh</code> | Caractere com valor hexadecimal <i>hh</i> |

https://docs.python.org/pt-br/3/reference/lexical_analysis.html#strings

EXERCÍCIOS

1. Utilizando estrutura de repetição imprima cada letra da palavra Python de trás para frente.
2. Um palíndromo é uma palavra ou frase que quando lida de trás para frente permanece igual. Com as palavras a seguir adicione em uma lista e percorra cada elemento, mostrando se a palavra é um palíndromo ou não: "ana", "pelo", "ovo", "reviver", "a grama é amarga", "a mala nada na lama", "telhado", "abacate", "radar", "osso", "viver para viver", "reler", "ame o poema", "sol" , "rever", "sala". Também exiba a quantidade total de palavras que são palíndromos.
3. Considere a seguinte string com nomes de frutas: "banana uva maçã melão abacaxi". Coloque todas as palavras da string em maiúsculo, depois separe as frutas da string adicionando em uma lista utilizando o split. Verifique se uva está na lista e troque pela fruta morango. Por último imprima a lista.
4. Receba uma entrada que pede o nome completo do usuário, verifique se o nome completo do usuário possui "de" se sim utilize o replace para remover a palavra da string, conte o tamanho total do nome completo e por último pegue o primeiro e último nome do usuário para concatenar em outra string. Exiba o nome completo, tamanho e a string com o primeiro e último nome do usuário.
5. Receba uma data do usuário no formato dd/mm/aaaa e imprima a data com o mês escrito por extenso.