Strings

String

▼ Interpolação de string com (**%**)

> ❗ Semelhante a linguagem C

- Sintaxe:
  - `variavel = "Produto: %s , preço %.2f " % (produto, preco)`
- int (**%d**), float (**%f**), str (**%s**), hexadecimal (**%X**)

▼ Formatar string (**f**)

- Variável dentro de string
  - Exemplo: `print ( f "{ nome } tem { altura } de altura" )`
- Casa decimal
  - Exemplo: `f "{ altura :.2f }"`

▼ **format**( )

```
a, b, c = "Pedro", "Silva", 1.1
string = "nome: {}, sobrenome: {}, valor: {}"
print(string.format(a, b, c))
```

Exibe: nome: Pedro, sobrenome: Silva, valor: 1.1

▼ String iterável

- Cada caractere de uma string pode ser relacionado a um índice, semelhante ao funcionamento de um vetor. Exemplo:

  G u s t a v o

  0 1 2 3 4 5 6

  `print (nome[ 2 ])` - Exibe: s

- Também existem índices negativos. Exemplo:

  G u s t a v o

  -7 -6 -5 -4 -3 -2 -1

- Operador **in** retorna um valor bool para verificar se uma sequencia de caracteres está contida em uma string
  - Exemplo: `print ( "avo" in nome)` - Exibe: True

▼ Fatiamento de strings (**[i : f : p]**)

- Pega "fatias" de um vetor, sendo **i** = inicio, **f** = fim(para uma posição antes do fim) e **p** = passos(de um por um, dois por dois...). Exemplo para string = "0123456789":
  - `print (variavel [:6] )` - Exibe: 012345
  - `print (variavel [0::2] )` - Exibe: 02468
  - `print (variavel [::-1] )` - Exibe: 9876543210
  - `print (variavel [1:9:2] )` - Exibe: 135

▼ Calcular tamanho da string (**len( )**)

- É utilizado para contar a quantidade de caracteres de uma string. Também pode ser utilizado para calcular o tamanho de um vetor. Exemplo:

```
nome = "Joao Pedro"
print(len(nome))
# Exibe 10
```

- Sempre retorna um valor int

▼ Substituir caractere (**replace( )** ou **import re, re.sub( )**)

- Função utilizada para substituir ou remover caracteres de uma string. Exemplo:

```
cpf = "123.456.789-00"
cpf_limpo = cpf.replace(".", "").replace("-", "")
print(cpf_limpo)  # Exibe "12345678900"
```

- Uma alternativa para isso poderia ser utilizar o método `sub()` da biblioteca re
  - Sintaxe: re.sub(caracteres que devem ser removidos, substituto, string)
  - Exemplo:

```
import re
cpf = "123.456.789-00"
cpf_limpo = re.sub(r"[.-]", "", cpf)
print(cpf_limpo)  # Exibe "12345678900"
```

▼ Converter para minúsculo ou maiúsculo (**lower( )** e **upper( )**)

- texto.lower( ) - converte uma string para letras minúsculas
- texto.upper( ) - converte uma string para letras maiúsculas

▼ Contar frequência de um caractere em uma string (**count( )**)

- É utilizado como método para contar a quantidade de vezes que um caractere apareceu em uma string, funcionando apenas para o tipo string. Exemplo de uso:

```
frase = "Frase de exemplo"
qtd_apareceu_e = frase.count("e")  # Quantas vezes o "e" apareceu na
print(qtd_apareceu_e)  # Exibe 4
```

▼ Primeira letra (**startswith( )**)

```
sair = input("Sair? [S]im ou [N]ao")
sair = sair.startswith("S")
sair = sair.startswith("N")
```

▼ Dividir uma string (**split( )**)

- Método que é utilizado para dividir uma string em uma lista de substrings com base em um delimitador especificado (por padrão, espaços), que é passado no argumento do método. Exemplo:

```
frase = "Frase de exemplo"
lista_frases = frase.split(" ")  # Escolhe " " como delimitador
print(lista_frases)  # Exibe: ['Frase', 'de', 'exemplo']
```

▼ Eliminar espaço nas extremidades de uma string (**strip( )**)

- O método `strip()` remove os espaços em branco (ou caracteres especificados) do início e do fim de uma string. Exemplo:

```
texto = "  Olá mundo!   "
limpo = texto.strip()
print(limpo)  # Exibe: "Olá mundo!"
```

- Exemplo definindo um caractere específico:

```
texto = "###Python###"
limpo = texto.strip("#")
print(limpo)  # Exibe: "Python"
```

- `rstrip()` e `lstrip()` servem para remover o caractere da direita e da esquerda, respectivamente

▼ Unir lista de string em uma única string (**join( )**)

- O método `join()` une uma lista de strings em uma única string, usando um delimitador especificado entre elas. Exemplo:

```
itens = ["maçã", "banana", "cereja"]
lista = ", ".join(itens)
print(lista)  # Saída: "maçã, banana, cereja"
```

▼ Adicionar caracteres (**<**, **^**, **>**)

- `variavel = "ABC"`

- `print(f"{variavel:>10}.")` - Exibe :____ABC. (Coloca caracteres antes da variavel)
- `print(f"{variavel:<10}.")` - Exibe: ABC____. (Coloca caracteres depois da variavel)
- `print(f"{variavel:^10}.")` - Exibe:__ABC__. (Centraliza a variavel entre caracteres)