Aula 7: Strings

Luís Felipe

UFF

22 de Setembro de 2025

Luís Felipe

22/09/25

Strings

- Em Python, strings são listas imutáveis de caracteres.
- Podem ser representadas entre aspas simples (' ') ou aspas duplas (" ").
- Cada caractere ocupa uma posição (índice), mas os elementos não podem ser alterados.

Exemplos:

```
1  texto = "Olá turma"
2  print(texto)          # Olá turma
3
4  texto = 'Olá turma'
5  print(texto)          # Olá turma
6  print(texto[2])       # á
7
8  texto[2] = 'z'
9  Traceback (most recent call last):
10   File "<python-input-155>", line 1, in <module>
11     texto[2] = 'z'
12     ~~~~~^^^
13 TypeError: 'str' object does not support item assignment
```

Luís Felipe

22/09/25

Strings

O caractere '\n' pode fazer parte de uma string e ele só causa a mudança de linha no comando print.

```
>>> texto = 'olha\nvoce'
>>> texto
'olha\nvoce'
>>> print(texto)
olha
voce
```

Luís Felipe

22/09/25

Strings – Concatenação, Repetição e Slicing

- O operador `+` concatena duas strings.
- O operador `*` repete a concatenação.
- O **slicing** retorna a parte da string entre duas posições.
- A string vazia é representada como `''`.

Exemplos:

```
'qwerty' + 'poiuy'
# 'qwertypoiuy'

3 * 'abc'
# 'abcabcabc'

a = 'qwerty'
a[2:4]  # 'er'

vazia = ''
```

Luís Felipe

22/09/25

Strings como Listas e Inversão

- Strings podem ser percorridas com um `for`, como listas.
- Para inverter, por exemplo, adicionamos cada caractere no início de uma nova string.

```
1  st = input("Digite um texto: ")
2  inv = ''
3  for x in st:
4      inv = x + inv
5  print(inv)
```

```
1  st = "olaturma"
2  inv = ''
3  for i in range(len(st)):
4      inv = st[i] + inv
5  print(inv)   # amrutalo
```

```
1  st = input("Digite um texto: ")
2  inv = ''
3  for x in st:
4      inv = x + inv
5      print(inv)
6  print(inv)
7
8  # Entrada: olaturma
9  # Saída: o, lo, alo, talo, utalo, rutalo, mrutalo, amrutalo, amrutalo
```

Luís Felipe

22/09/25

Métodos úteis de Strings

- `strip()`: remove espaços/brancos do início e fim.
- Operador `in`: verifica se uma substring está presente.
- `find()`: retorna o índice da primeira ocorrência (ou -1).

```
1  aa = ' \n abcndef \n'
2
3  print(aa.strip())                    # 'abcndef'
4
5  print('tho' in 'python')             # True
6
7  print('thor' in 'python')            # False
8
9  p = 'python'
10 print(p.find('tho'))                 # 2
11 print(p.find('thor'))                # -1
```

Luís Felipe

22/09/25

Separando e Substituindo Strings

- `split(sep)`: separa string em lista usando um separador.
- `replace(a, b)`: substitui todas as ocorrências de `a` por `b`.

```
1  a = "1; 2; 3"
2  print(a.split('; '))
3  # ['1', '2', '3']
4
5  b = "ouviram do ipiranga margens"
6  print(b.split())
7  # ['ouviram', 'do', 'ipiranga', 'margens']
8
9  a = "abcabcdfgabc abc a b c"
10 print(a.replace("abc",""))
11 # 'dfg a b c'
```

Luís Felipe

22/09/25

Transformando String em Lista e vice-versa

- `list(s)`: converte string em lista de caracteres.
- `"".join(lista)`: junta lista em uma string.

  Obs.: O método join() em Python só funciona se cada elemento da lista for uma string.

```
1  a = "abc\n;abc"
2  x = list(a)
3  print(x)                # ['a', 'b', 'c', '\n', ';', 'a', 'b', 'c']
4
5  print("".join(x))       # 'abc\n;abc'
6
7  x = ['a', 'b', 'c']
8  print("".join(x))       # abc
9  print("-".join(x))      # a-b-c
10
11 x = [1, 2, 3]
12 print("".join(l))
13 # TypeError: sequence item 0: expected str instance, int found
```

Solução: converter cada item para string antes:

```
1  y = "".join(str(i) for i in x)  # '123'
```

Luís Felipe

22/09/25

Contando Palavras em um Texto

Exercício: Contar o número de palavras em um texto.

```
1  st = input("Digite um texto: ")
2  pontuacao = [".", ",", ":", ";", "!", "?"]
3
4  for pont in pontuacao:
5      st = st.replace(pont, " ")
6  numPal = len(st.split())
7
8  print("Num. palavras:", numPal)
```

Luís Felipe

22/09/25

Encontrando todas as ocorrências de uma palavra

Exercício: Fazer a busca de todas as ocorrências de uma palavra em um texto.

```
1  st = input("Entre com um texto: ")
2  subst = input("Entre com uma palavra: ")
3
4  pos = []
5  tamRemovido = 0
6
7  while subst in st:
8      aux = st.find(subst)
9      pos.append(aux + tamRemovido)
10     st = st[aux+1:]
11     tamRemovido += aux + 1
12
13 print(pos)
```

Explique o que o código faz em cada etapa.

Luís Felipe

22/09/25

Outros métodos de Strings – count, upper, lower

- `count(subStringProcurada)`: retorna a quantidade de ocorrências da substring na string.
- `upper()`: retorna uma cópia da string com todos os caracteres alfabéticos convertidos para maiúsculos.
- `lower()`: retorna uma cópia da string com todos os caracteres alfabéticos convertidos para minúsculos.

Exemplos:

```
1  texto = "Python é poderoso. python é versátil."
2  print(texto.count("Python"))  # 1
3  print(texto.count("python"))  # 1
4
5  print(texto.upper())  # "PYTHON É PODEROSO. PYTHON É VERSÁTIL."
6  print(texto.lower())  # "python é poderoso. python é versátil."
```

Luís Felipe

22/09/25

Exemplos com count()

- count() conta ocorrências não sobrepostas.
- Podemos usar parâmetros opcionais para definir onde a busca começa e termina.

Exemplos:

```
1  frase = "banana"
2
3  # Conta 'ana' na string (não sobrepõe)
4  print(frase.count("ana"))      # 1
5
6  # Conta 'a'
7  print(frase.count("a"))        # 3
8
9  # Contando em parte da string
10 print(frase.count("a", 2))     # 2   (começa do índice 2)
11 print(frase.count("a", 2, 4))  # 1   (entre índices 2 e 3)
```

Observação: Para contar sobreposições, seria necessário um algoritmo customizado.