

Departamento de Engenharia de Sistemas e Computação

Strings

Manipulação de texto e formatação

João Araujo

Algoritmos Computacionais I

Aula 22

Roteiro

- Strings: literais, aspas e caracteres especiais (Cap. 6.1)
- Strings são sequências imutáveis
- Fatiando strings (6.2)
- Operações e métodos sobre strings (6.3)
- Formatação: `format` e f-strings
- Aplicações: palíndromos e formatação de nomes

Strings: aspas e caracteres especiais

```
ordem1 = '"Romanes\neunt\ndomus"?\n'
ordem2 = "\'Romani ite domum\'"
print(ordem1)
print(ordem2)
```

```
"Romanes
eunt
domus"?

'Romani ite domum'
```

\n nova linha; \t tabulação; \\ barra; \' e \" aspas. Aspas simples ou duplas são equivalentes: use uma para poder escrever a outra dentro.

Fonte: code/Cap6/prog.6.01_romani.py

Aspas triplas: textos longos

```
ordem = '''Romanes eunt domus?
"Romani ite domum"'''
print(ordem)
```

- Triplas aspas (''' ou """) aceitam várias linhas e aspas dentro
- São também usadas nas *docstrings* das funções (Aula 10)
- Para "quebrar" uma string longa no código, a barra invertida \ no fim da linha continua o texto

Fonte: code/Cap6/prog.6.02_celacanto.py, 6.03_aspastriplas.py

Strings são sequências

```
frase = 'Celacanto'
print(len(frase))           # 9
print(frase[0], frase[-1]) # C o
for letra in frase:
    print(letra)
for i in range(len(frase)):
    print(i, frase[i])
print('a' in frase)         # True
print('can' in frase)       # True (subcadeia)
```

- A *leitura* funciona como nas listas: índices, `len`, `for`, `in` (que procura *subcadeias*). Comandos que *modificam* (`append`, `sort`, ...) não existem em strings: Exercício 6.2

Fonte: `code/Cap6/prog.6.04_percorrer_string.py, 6.05_percorrer_string_com_range.py`

Strings são imutáveis

```
frase = 'Romani'
frase[1] = 'o'          # TypeError: 'str' object does not support
                        # item assignment
```

Solução: construir uma *nova* string

```
frase = 'Romani'
frase = frase[:-2] + 'o'
print(frase)            # Romao
frase = frase.replace('o', '0')   # também devolve nova string
```

Lembre da Aula 18

Strings (e números, tuplas) são imutáveis: passá-las a uma função nunca as altera. Veja `code/Cap6/prog.6.08_passagem_listas_strings.py`.

Fonte: `code/Cap6/prog.6.06_string_imutavel.py`, `6.07_string_imutavel1.py`

Fatiando strings

```
frase = 'Romani ite domum'
print(frase[7:10])      # ite
print(frase[:6])        # Romani
print(frase[7:])        # ite domum
print(frase[7:-1])      # ite domu
print(frase[:])         # Romani ite domum
print(frase[::-1])      # mumod eti inamoR
```

- Mesma regra das listas: `s[ini:fim:passo]`, com `fim` exclusivo
- `s[::-1]` inverte a string (Exercício 6.4)

Fonte: `code/Cap6/prog.6.09_fatia_string.py`

Métodos de string

```
s = '  Python é Divertido!  '
print(s.upper())             # '  PYTHON É DIVERTIDO!  '
print(s.lower())             # minúsculas
print(s.strip())             # remove espaços das pontas
print(s.strip().title())     # 'Python É Divertido!'
print(s.replace('Python', 'Java'))
print(s.count('i'))          # 2
print(s.find('é'))           # posição (ou -1 se não achar)
print(s.strip().startswith('Py'), s.strip().endswith('!'))
print('42'.isdigit(), 'abc'.isalpha())
```

Métodos de string sempre devolvem uma nova string (a original não muda).

split e join

```
nome = 'João Araujo Ribeiro'
lista = nome.split(' ')
print(lista)                          # ['João', 'Araujo', 'Ribeiro']
nome_formatado = lista[-1] + ', ' + lista[1][0] + '. ' + lista[0]
print(nome_formatado)                 # Ribeiro, A. João

dados = '21.3;22.1;23.4'
valores = [float(x) for x in dados.split(';')]
print(valores)                        # [21.3, 22.1, 23.4]
print(' - '.join(['a', 'b', 'c'])) # a - b - c
```

- `split(sep)`: string → lista; `sep.join(lista)`: lista → string
- Exercícios 6.8 e 6.9: nomes com mais ou menos de três partes

Fonte: `code/Cap6/prog.6.10_formata_nome.py, 6.11_formata_nomealt.py`

Formatação de strings

```
nome, nota = 'Ana', 8.456
print('Aluno {} tirou {:.1f}'.format(nome, nota))
print(f'Aluno {nome} tirou {nota:.1f}')          # f-string
print(f'{nome:>10}|{nota:<8.2f}|{42:05d}')       # alinhamento e zeros
print(f'{0.256:.1%}  {1234567:,}')                # 25.6%  1,234,567
print(f'{"Graus":>9}{"Seno":>9}')
```

```
Aluno Ana tirou 8.5
Aluno Ana tirou 8.5
       Ana|8.46    |00042
25.6%  1,234,567
    Graus     Seno
```

> direita, < esquerda, ^ centro; `.2f` duas casas; `d` inteiro.

Fonte: `code/Cap4/prog.4.a_exemplos_format.py`

Aplicação: palíndromos

Uma palavra é palíndromo se é igual lida de trás para a frente (*ovo*, *arara*, *radar*).

```
def palindromo(palavra):
    return palavra == palavra[::-1]

print(palindromo('ovo'))        # True
print(palindromo('Ovo'))        # False  <- diferencia maiúsculas!
```

Exercícios 6.5 a 6.7: outras formas de testar; ignorar maiúsculas com `lower()`; frases inteiras ("A sogra má te dá amor, Mateus, e ama o tema do Mateus"). Dicas para frases: remover espaços e pontuação, eliminar acentos. Fonte: `code/Cap6/prog.6.a_palindromo.py`

Palíndromo em frases

```python
def palindromo_frase(frase):
    """Diz se a frase é palíndromo, ignorando espaços e maiúsculas."""
    s = frase.lower().replace(' ', '')
    return s == s[::-1]

print(palindromo_frase('Socorram me subi no onibus em Marrocos'))
```

- Passo a passo: `lower()` → `replace(' ', '')` → comparar com o inverso
- Acentos e pontuação exigem tratamento extra: um bom *caso-limite* para testar

Contando espaços e palavras (Exercício 6.3)

```
texto = input('Digite uma frase: ')
espacos = 0
for c in texto:
    if c == ' ':
        espacos += 1
print('Espaços:', espacos)
print('Também:', texto.count(' '))
print('Palavras:', len(texto.split()))
```

- Compare a solução manual (`for`) com o método pronto (`count`)
- `split()` sem argumento separa por *qualquer* quantidade de espaços

Para praticar: exercícios do livro

Site do livro – Capítulo 6

- Strings: **6.1** (Romani ite domum), **6.2** (o que das listas funciona em strings), **6.3** (contando espaços)
- Fatiamento: **6.4** (string invertida), **6.5, 6.6, 6.7** (palíndromos)
- Operações: **6.8, 6.9** (nome composto)

Enunciados e soluções comentadas em
`https://www.araujo.eng.uerj.br/livro/exercicios/cap6/`

Exercícios adicionais (sem solução): `https://www.araujo.eng.uerj.br/livro/extras/`

Resumo da aula

- String é uma sequência *imutável* de caracteres: índices, fatias, `len`, `in`
- Métodos: `upper`, `lower`, `strip`, `replace`, `find`, `split`, `join`...
- Formatação com f-strings: `f'{x:.2f}'`
- Próxima aula: arquivos