


Departamento de Engenharia de Sistemas e Computação

# Estruturas de Dados I

## Sequências e listas

---

João Araujo

Algoritmos Computacionais I

Aula 16

# Roteiro

- O problema: guardar muitos valores (Cap. 5.1)
- Listas em Python: criação, índices, `len` (5.2)
- Percorrendo uma lista: `for`, `range(len())`, `enumerate`
- Listas são mutáveis
- Lendo dados e construindo listas com `append`

# O problema: muitas variáveis

```
nota1 = 7; nota2 = 8; nota3 = 10; nota4 = 9
media = (nota1 + nota2 + nota3 + nota4) / 4
```

- E se a turma tiver 40 alunos? E 4 000 medições de um sensor?
- Precisamos de uma **sequência**: uma coleção *ordenada* de valores acessíveis por posição
- Em Python: `list`, `tuple`, `str` (texto é uma sequência de caracteres), `range`

**Hoje**

A **lista**: a estrutura de dados mais usada em Python.

# Criando e acessando listas

```
notas = [7, 8, 10, 9]
print(notas[0])         # 7  (o primeiro índice é 0)
print(notas[3])         # 9
print(notas[-1])        # 9  (índice negativo: a partir do fim)
print(len(notas))       # 4
soma = notas[0] + notas[1] + notas[2] + notas[3]
print("A média é igual a ", soma / 4)
```

| índice | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| negativo | -4 | -3 | -2 | -1 |
| notas | 7 | 8 | 10 | 9 |

Fonte: `code/Cap5/prog.5.1_notas.py`

# Percorrendo uma lista

**Direto pelos elementos** (mais simples):

```
notas = [7, 8, 10, 9]
soma = 0
for nota in notas:
    soma += nota
print("A média é igual a ", soma / len(notas))
```

**Pelos índices** (quando precisa da posição):

```
soma = 0
for i in range(len(notas)):
    soma = soma + notas[i]
```

Fonte: `code/Cap5/prog.5.5_notas_com_lista.py, 5.2_notas_com_range.py`

# enumerate: índice e valor juntos

```
dias_da_semana = ['Domingo', 'Segunda-feira', 'Terça-feira',
    'Quarta-feira', 'Quinta-feira', 'Sexta-feira', 'Sábado']
for i, dia in enumerate(dias_da_semana, 1):
    print('O dia', i, 'é', dia)
```

```
O dia 1 é Domingo
O dia 2 é Segunda-feira
...
```

O segundo argumento de `enumerate` é o valor inicial da contagem.

Fonte: `code/Cap5/prog.5.a_dias_enumerate.py`

# Listas são mutáveis

```
notas = [7, 8.1, 10, 9]
print(notas)            # [7, 8.1, 10, 9]
notas[3] = 6
print(notas)            # [7, 8.1, 10, 6]
```

- Pode-se alterar um elemento no lugar; o tamanho continua igual
- Diferente de strings e tuplas (**imutáveis**), que veremos adiante
- Listas aceitam tipos *misturados*: `['Ana', 21, 8.5, True]`
- Índice fora do intervalo ⇒ `IndexError`

Fonte: `code/Cap5/prog.5.6_lista_mutavel.py`

# Dobrando todos os valores

```
lista = [1, 2, 3, 5, 7]
for i in range(len(lista)):
    lista[i] *= 2
print(lista)            # [2, 4, 6, 10, 14]
```

**Pegadinha**

`for i in lista: i *= 2` não altera a lista! A variável `i` é só uma cópia do valor. Para modificar os elementos é preciso usar o *índice*.

- Alternativa elegante: `for i, item in enumerate(lista): lista[i] = item * 2`

Fonte: `code/Cap5/prog.5.9_dobra_lista2.py, 5.10_duplica_enumerate.py`

# Construindo listas com append

```
numeros = []                    # lista vazia
for i in range(10):
    n = int(input('Digite um número:'))
    numeros.append(n)           # acrescenta ao final
print('A lista criada é:', numeros)
```

- Padrão mais comum: começar com [] e acrescentar valores
- Útil quando não se sabe o tamanho de antemão (leitura de dados, resultados de cálculos)

Fonte: code/Cap5/prog.5.26_listas_append.py

# Exemplo: temperaturas em Celsius (Exercício 5.1)

```
celsius = [22.5, 25.0, 19.8, 30.1, 27.4]
fahrenheit = []
for c in celsius:
    fahrenheit.append(9 / 5 * c + 32)

for c, f in zip(celsius, fahrenheit):
    print(f"{c:6.1f} C = {f:6.1f} F")
```

- `zip` percorre duas listas em paralelo
- Todos os dados na mesma estrutura; uma só função poderia processá-los

# Funções úteis para listas de números

```
notas = [7, 8, 10, 9, 5.5]
print(len(notas))                  # 5
print(sum(notas))                  # 39.5
print(min(notas), max(notas))      # 5.5 10
print(sum(notas) / len(notas))     # 7.9
print(sorted(notas))               # [5.5, 7, 8, 9, 10]
print(7 in notas, 3 in notas)      # True False
```

- Já resolvemos o "Problema 4" da Aula 9 (dias acima da média): agora é só percorrer a lista com um contador

# Atividades

1. Leia 5 temperaturas e mostre a média, a maior, a menor e quantas ficaram acima da média.
2. Dada a lista de notas de uma turma, calcule a nota média e liste os índices dos alunos abaixo de 5.
3. Escreva uma função `media(lista)` que devolve a média (e trate a lista vazia).
4. Reescreva o programa de Fahrenheit (Exercício 5.2) imprimindo a tabela do **último** para o primeiro valor.

# Para praticar: exercícios do livro

## Site do livro – Capítulo 5

- Processando uma lista: **5.1** (temperaturas), **5.2** (Fahrenheit invertido), **5.3** (com `try/except`)
- **5.4** (`while`), **5.5** (ímpares para pares), **5.6** (tabela de $\pi$)

Enunciados e soluções comentadas em
`https://www.araujo.eng.uerj.br/livro/exercicios/cap5/`

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

# Resumo da aula

- Lista = sequência ordenada e mutável; índices a partir de 0 (e negativos a partir do fim)
- `len`, `sum`, `min`, `max`, `in`
- Percorra por elementos, por índices ou com `enumerate`
- Para alterar elementos, use o índice
- Próxima aula: fatiamento, compreensão de listas e métodos