

Listas em Python

Prof. Fábio Procópio

Prof. João Nascimento



2

Relembrando...

- Na [aula passada](#), iniciamos nossos estudos sobre uma estrutura de controle chamada **Estrutura de Repetição (ou Iteração)**. Os assuntos abordados foram:
 - Estrutura do **while**
 - Estrutura do **for**
 - Função **range()**





Introdução – 1 de 2

- **Lista** é uma estrutura de dados composta por elementos organizados de forma linear
 - Cada elemento pode ser acessado a partir de um índice que representa a sua posição na coleção

Índice	0	1	2	3	4	5
Elemento	102030	Fábio	Procópio	True	1.82	[4, 2]

- Representada como uma sequência de elementos separados por vírgula e dentro de colchetes []
- Os índices iniciam em 0, isto é, o primeiro elemento tem índice 0, o segundo índice 1 e, assim, por diante
- Os elementos de uma lista podem ser de vários tipos como os primitivos (int, float, string, booleano) ou os mais complexos como listas, dicionários e objetos
- Durante a definição de uma lista, não é necessário definir o tamanho máximo da lista porque ela cresce dinamicamente



Introdução – 2 de 2

➤ Exemplo da implementação de uma lista:

```
# Criando uma lista vazia com notas de alunos  
notas = []
```

```
# Atribuindo notas dos alunos  
notas = [8.3, 7.1, 2.25, 4]
```

```
# Imprimindo a nota do 2º bimestre  
print("Bim II: {:.2f}".format(notas[1]))
```

```
# Alterando a nota do 3º bimestre  
notas[2] = 6.4
```

```
# Imprimindo todas as notas  
for n in range(4):  
    print(notas[n])
```

É possível adicionar elementos a uma lista usando o método **append()**, o qual será apresentado em breve.



5

Exercício de Fixação 1

- 1) Crie uma lista chamada `regiao_sudeste`
 - a) Armazene os nomes dos estados da região sudeste dentro dela
 - b) Imprima o nome do estado que está armazenado no primeiro índice da lista
 - c) Altere o nome dos estados para o nome das suas respectivas capitais
 - d) Use uma estrutura de repetição para imprimir os elementos da lista



Método `append()`

- Usado para **adicionar** um novo elemento ao fim da lista;
- Sintaxe básica:

```
nome_lista.append(elemento)
```

- Exemplo:

```
alunos = ["Wilma", "Ana", "Patrícia"]  
alunos.append("Beatriz")  
print(alunos)
```



```
['Wilma', 'Ana', 'Patrícia', 'Beatriz']
```



7

Método insert()

- Usado para **inserir** um novo elemento em um **índice específico** da lista;
- Sintaxe básica:

```
nome_lista.insert(indice, elemento)
```

- Exemplo:

```
alunos = ["Wilma", "Ana", "Patrícia", "Beatriz"]  
alunos.insert(2, "Camila")  
print(alunos)
```



2

```
['Wilma', 'Ana', 'Camila', 'Patrícia', 'Beatriz']
```



Método sort()

- Usado para **ordenar** uma lista. Por padrão, a ordenação é ascendente. Para ordenação decrescente, passa-se False como parâmetro;
- Sintaxe básica:

```
nome_lista.sort(reverse=<True | False>)
```

- Exemplos:

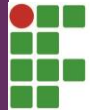
Mesmo que alunos.sort()

```
alunos = ["Wilma", "Ana", "Camila", "Patrícia", "Beatriz"]  
alunos.sort(reverse = False)  
print(alunos)
```

```
alunos = ["Wilma", "Ana", "Camila", "Patrícia", "Beatriz"]  
alunos.sort(reverse=True)  
print(alunos)
```

['Ana', 'Beatriz', 'Camila', 'Patrícia', 'Wilma']

['Wilma', 'Patrícia', 'Camila', 'Beatriz', 'Ana']



Método shuffle()

- Usado para **embaralhar** os elementos de uma lista
 - Para isso, é necessário importar a biblioteca **random**
- Sintaxe básica:

```
random.shuffle(nome_lista)
```

- Exemplo:

```
import random  
  
alunos = ["Wilma", "Ana", "Patrícia", "Beatriz"]  
random.shuffle(alunos)  
print(alunos)
```



Exercício de Fixação 2

Crie um programa que possibilite o usuário informar os nomes de alunos de uma determinada turma. Os nomes devem ser armazenados em uma lista.

O usuário não sabe a quantidade de alunos da turma e, portanto, você deve criar algum critério que permita a parada do cadastro. Ao fim do cadastro, o programa deve imprimir os nomes dos alunos da turma em ordem alfabética.



len

- Usado para determinar a **quantidade de elementos** existentes em uma lista;
- Sintaxe básica:

```
len(nome_lista)
```

- Exemplo:

```
times = ["ABC", "Mais Querido", "Natal/RN", "Globo", "ABC"]  
tam = len(times)  
print(tam)
```



5



Método count()

- Usado para **contar** a quantidade de ocorrências de um elemento em uma lista;
- Sintaxe básica:

```
nome_lista.count(elemento)
```

- Exemplo:

```
alunos = ["Wilma", "Ana", "Camila", "Patrícia", "Wilma"]  
x = alunos.count("Wilma")  
print(x)
```



2



min(), max() e sum()

- São funções de agregação usadas para realizar algumas operações sobre uma lista
 - **min**: retorna o menor valor de uma lista
 - **max**: retorna o maior valor de uma lista
 - **sum**: retorna a soma dos elementos de uma lista

- Sintaxe básica:

```
min(nome_lista)
max(nome_lista)
sum(nome_lista)
```

- Exemplos:

```
numeros = [1, 2, 3]
print(min(numeros))
print(max(numeros))
print(sum(numeros))
print(sum(numeros)/len(numeros))
```

Calcula a média aritmética dos elementos da lista. Para isso, somaram-se os elementos **sum(numeros)** e dividiu-se o resultado pela quantidade de elementos **len(numeros)**.



Exercício de Fixação 3

Crie um programa no qual o usuário informe N números. Em uma lista, armazene todos os números positivos informados. Ao fim, calcule e imprima a média geométrica entre o menor e o maior elementos da lista.



Método remove()

- Usado para **remover** a primeira ocorrência de um **elemento especificado** na lista;
- Sintaxe básica:

```
nome_lista.remove(elemento)
```

- Exemplo:

```
alunos = ["Wilma", "Ana", "Camila", "Patrícia", "Beatriz"]  
alunos.remove("Ana")  
print(alunos)
```



```
['Wilma', 'Camila', 'Patrícia ', 'Beatriz']
```



Método pop()

- Usado para **apagar** um **índice específico** da lista. Caso não seja informado, o último elemento será apagado;
- Sintaxe básica:

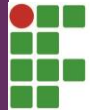
```
nome_lista.pop([índice])
```

- Exemplo:

```
alunos = ["Wilma", "Ana", "Camila", "Patrícia", "Beatriz"]  
alunos.pop(0) #apaga Wilma  
alunos.pop()  #apaga Beatriz  
print(alunos)
```



```
['Ana', 'Camila', 'Patrícia ']
```

Método clear()

➤ Usado para **esvaziar** uma lista;

➤ Sintaxe básica:

```
nome_lista.clear()
```

➤ Exemplo:

```
alunos = ["Wilma", "Ana", "Camila", "Patrícia", "Beatriz"]  
alunos.clear()  
print(alunos)
```





Método copy()

➤ Usado para **fazer uma cópia** de uma lista;

➤ Sintaxe básica:

```
nome_lista.copy()
```

➤ Exemplo:

```
alunos = ["Wilma", "Camila", "Patrícia", "Beatriz"]  
backup = alunos.copy()  
print(backup)
```



```
['Wilma', 'Camila', 'Patrícia ', 'Beatriz']
```



Exercício de Fixação 4

Crie um programa no qual o usuário informe N números. Em uma lista, armazene todos os números positivos informados.

Em seguida, pergunte ao usuário o que ele deseja remover da lista: números pares ou números ímpares. De acordo com a resposta, percorra a lista e remova os elementos que atendem a solicitação do usuário.



Método index()

- Usado para **identificar** a posição da **primeira ocorrência** de um determinado valor especificado;

- Sintaxe básica:

```
nome_lista.index(valor)
```

- Exemplo:

```
times = ["ABC", "Mais Querido", "Natal/RN", "Globo", "ABC"]  
x = times.index("ABC")  
print(x)
```

Embora **ABC** apareça no índice 0 e no índice 4, o método **index()** retorna apenas o índice que apresenta a primeira ocorrência de um valor especificado. No caso de **ABC**, é o índice 0.

0



- Agora, veja o que ocorre ao procurar por um **Time da Série D** que não está na lista:

```
times = ["ABC", "Mais Querido", "Natal/RN", "Globo", "ABC"]  
x = times.index("América RN")  
print(x)
```

```
x = times.index("América RN")  
ValueError: 'América RN' is not in list
```





in – 1 de 2

- Operador lógico utilizado para verificar se um determinado elemento é membro de uma lista:
- Exemplo:

```
times = ["ABC", "Mais Querido", "Natal/RN", "Globo", "ABC"]  
v = "ABC"  
if v in times:  
    print("Salve o Mais Querido!")
```



Salve o Mais Querido!

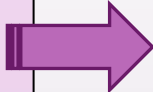


in – 2 de 2

➤ Também pode ser usado para percorrer os elementos de uma lista:

➤ Exemplo:

```
times = ["ABC", "Mais Querido", "Natal/RN", "Globo", "ABC"]  
for v in times:  
    print(v)
```



```
ABC  
Mais Querido  
Natal/RN  
Globo  
ABC
```



Exercício de Fixação 4

Construa um programa no qual o usuário cadastre um número indeterminado de alunos de sua turma.

Ao fim do cadastro, o usuário deve informar qual aluno deseja excluir do cadastro. Caso o aluno informado não tenha sido localizado, o programa deve apresentar uma mensagem informando “Aluno não cadastrado.”. Caso contrário, o programa deve apagar o nome do aluno do cadastro e informar a mensagem “Aluno excluído com sucesso.”. Por fim, o programa deve imprimir o cadastro dos alunos ordenado alfabeticamente.



Comparação de listas

- Duas listas são iguais se, e somente se, têm o mesmo comprimento e todos os elementos das mesmas posições são iguais;
- Sintaxe básica:

```
lista1 == lista2
```

- Exemplo:

```
lista1 = ["ABC", "Mais Querido", "Natal/RN"]  
lista2 = ["Mais Querido", "ABC", "Natal/RN"]
```

```
if lista1 == lista2:  
    print("Listas são iguais.")  
else:  
    print("Listas são diferentes.")
```

Apesar das duas listas armazenarem os mesmos elementos e terem o mesmo comprimento, o elemento **ABC** não está armazenado na mesma posição das duas listas.

Listas são diferentes.



enumerate()

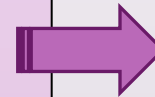
- Pode ser usada para percorrer uma lista e obter, simultaneamente, os elementos e os respectivos índices de uma lista;

- Sintaxe básica:

```
for indice, elemento in enumerate(nome_lista):  
    comandos a serem executados
```

- Exemplo:

```
lista = ["ABC", "Mais Querido", "Natal/RN"]  
  
for ind, time in enumerate(lista):  
    print("{} --> {}".format(ind, time))
```



```
0 --> ABC  
1 --> Mais Querido  
2 --> Natal/RN
```



Material complementar

► Para complementar este material, assista às vídeo aulas que falam sobre

- 1) [Fatiamento de listas](#)
- 2) [Conversão de uma *string* para elementos de uma lista](#)
- 3) [Concatenação de elementos de uma lista](#)



Exercícios de Fixação

- 1) Crie um programa que carregue uma lista para armazenar N números inteiros positivos (considere que o usuário sempre informará valores distintos e inteiros positivos). Ao fim, mostre o menor e o maior número e em qual índice eles se encontram.
- 2) Construa um programa que crie uma lista A para armazenar 5 números informados pelo usuário. O programa também deve preencher uma lista B, contendo 5 números, cujos valores também são informados pelo usuário. Por fim, o programa deve colocar os valores que estão na lista A para a lista B e vice-versa.
- 3) Crie um programa para gerar a série de Fibonacci até o 15º termo e armazene-a dentro de uma lista. A série numérica de Fibonacci é a seguinte: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ...



Referências

- 1) EXCRIPT. **Curso de Python - [Aula 42] – Estrutura de Dados - Listas.** Disponível em <https://www.youtube.com/watch?v=smqY-ljrrUQ>. Acessado em 30 jan. 2019.
- 2) EXCRIPT. **Curso de Python - [Aula 46] – Listas III.** Disponível em <https://www.youtube.com/watch?v=aufzS7rrmnM>. Acessado em 30 jan. 2019.
- 3) EXCRIPT. **Curso de Python - [Aula 47] – Iterando listas.** Disponível em <https://www.youtube.com/watch?v=nOiySJIpwp0>. Acessado em 30 jan. 2019.
- 4) EXCRIPT. **Curso de Python - [Aula 48] – Fatiando listas.** Disponível em <https://www.youtube.com/watch?v=5Ctc5rIBouI>. Acessado em 30 jan. 2019.
- 5) EXCRIPT. **Curso de Python - [Aula 49] – Incluindo, alterando e excluindo elementos.** Disponível em <https://www.youtube.com/watch?v=AwEC1ptLWs8>. Acessado em 30 jan. 2019.
- 6) EXCRIPT. **Curso de Python - [Aula 50] – Ordenamento de listas.** Disponível em <https://www.youtube.com/watch?v=UslqFh1HAM8>. Acessado em 30 jan. 2019.
- 7) EXCRIPT. **Curso de Python - [Aula 51] – lista – quantidade de elementos.** Disponível em <https://www.youtube.com/watch?v=npsdj0kepTU>. Acessado em 30 jan. 2019.