



- Criando Listas
- Acessando por Index
- Adicionando Itens
- Remover Itens
- Imprimir Itens da Lista
- Métodos Extra

Python

Lógica de Programação

Estrutura de Dados Lista

01 Básico sobre Listas

- Imagine que você queira criar uma lista de todos os itens de uma feira. A primeira vista, você criaria uma variável para cada uma dos itens, não é mesmo? Mas, para isso você tem as listas, que aprenderemos agora. No exemplo a seguir, temos uma lista de *strings*, contendo 3 elementos.

```
1 listaFeira = ['maca', 'banana', 'uva']
```

- A lista acima, é uma lista com 3 elementos, na qual a primeira posição temos uma maçã, na segunda uma banana e na terceira uma uva. Entretanto, **em Python, a primeira posição é zero**, de modo que, a maçã está na posição 0, banana na posição 1 e uva na posição 2. A analogia é apresentada na imagem a seguir.

0	1	2
		

- A lista acima, é uma lista com 3 elementos, na qual a primeira posição temos uma maçã, na segunda uma banana e na terceira uma uva. Entretanto, **em Python, a primeira posição é zero**, de modo que, a maçã está na posição 0, banana na posição 1 e uva na posição 2. A analogia é apresentada na imagem a seguir.

```
1 listaFeira = ['maca', 'banana', 'uva']
2
3 print(listaFeira[0])
4 print(listaFeira[1])
5 print(listaFeira[2])
6
7 primeiroItemLista = listaFeira[0]
8 print('o primeiro item
   é', primeiroItemLista)
```

Saída

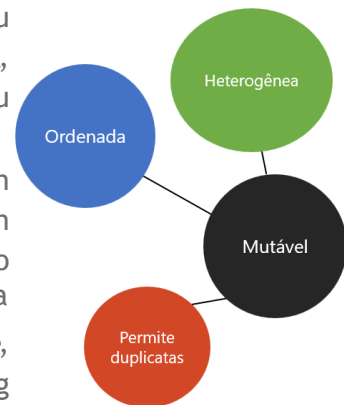
```
maca
banana
uva

o primeiro item é maca
```

```
listaFeira = ['maca', 'banana', 'uva']
               ↑       ↑       ↑
            listaFeira[0] listaFeira[1] listaFeira[2]
```

02 Propriedades de Listas

- **Mutável** significa “mudável”, ou seja, depois de criada a lista, podemos adicionar, remover ou modificar os elementos
- **Ordenada** significa que, cada item tem um índice (index) e seguem uma ordem (os novos são adicionados no final)
- **Permite duplicatas** significa que, se eu adicionar uma String ‘Daniel’, eu posso adicionar outra String ‘Daniel’ na mesma lista.
- **Heterogênea** significa que, uma lista pode ter ao mesmo tempo, elementos String, int e float, como o exemplo a seguir.
 - A primeira posição (0) é uma String (‘maca’)
 - A segunda posição (1) é um int
 - A terceira posição (2) é um float



```
1 lista = ['maca', 1, 1.0]
2
3 print(lista[0])
4 print(lista[1])
5 print(lista[2])
```

03 Criando Listas

- Anteriormente, nos antecipamos um pouco e criamos as nossas primeiras listas, não é mesmo? Agora, retomamos aos estágios preliminares de lista, que é a criação, que pode ser feita de **duas** formas, sendo elas:
 - Usando o construtor chamado list()
 - Utilizando colchetes [], como fizemos anteriormente

```

1 lista1 = []
2 lista2 = list()
3
4 lista3 = ['A', 'B']
5 lista4 = list( ('A', 'B') )

```

- No exemplo acima, criamos as lista1 e lista2 vazias, isto é, sem elementos. Complementarmente, as lista3 e lista4 possuem duas Strings ('A' na posição 0 e 'B' na posição 1).
 - Atenção ao modo em que a lista4 é criada, mais especificamente aos parênteses. Os parênteses indicados pela cor **azul** são relacionados ao construtor list(), enquanto que, os parênteses indicados pela cor **vermelha** estão associadas aos elementos, que por sua vez estão separados por vírgulas.

04 Acessando Elementos por Índice

- Os elementos da lista podem ser acessados por índice (em inglês, *index*). Como vimos anteriormente, os elementos em uma lista com Python, iniciam no índice 0 até tamanho - 1. Ou seja, em uma lista 3 elementos, vai de 0 a 2.

```
1 listaFeira = ['maca', 'banana', 'uva']
```



- Entretanto, o que não falamos é que, podemos ter índices negativos. Loucura, não?

```

1 lista1 = ['A', 'B', 'E', 'L', 'L', 'A']
2
3 print(lista1[1])
4 print(lista1[-5])

```

- No exemplo acima, nas linhas 3 e 4 será apresentado a String 'B'. O entendimento dos índices negativos é apresentado na imagem a seguir.

A	B	E	L	L	A
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

- Os elementos da lista podem ser acessados da direita para a esquerda usando índices negativos. O valor negativo começa de -1 a - tamanho da lista. Ou seja, indica que em Python podemos trabalhar de trás para frente.

05 Tamanho da Lista

- Em inglês, comprimento significa *length*. E vai ser com o método len() que vamos descobrir o tamanho de uma lista ou String.

- No exemplo a seguir, apresentamos como usar o método len.
 - Na linha 3 exibimos diretamente o tamanho
 - Nas linhas 5 e 6 também exibimos o tamanho, mas antes atribuímos o valor a uma variável

```

1 lista = ['A', 'B', 'E', 'L', 'L', 'A']
2
3 print(len(lista))
4
5 tamanhoLista = len(lista)
6 print(tamanhoLista)

```

- E agora, vamos mostrar o uso do método len() com Strings, que basicamente conta a quantidade de caracteres da String.

```

1 nome = 'Daniel'
2
3 print(len(nome))
4
5 tamanhoNome = len(nome)
6 print(tamanhoNome)

```

- Sim, mas quando vou usar esse método? No exemplo a seguir, usamos os métodos:

- Sum()** que soma todos os elementos da lista
- Len()** que retorna o tamanho (comprimento) da lista

```

1 notas = [10, 8, 9]
2
3 somaNotas = sum(notas)
4 media = somaNotas / len(notas)
5 print(media)

```

06 Imprimir Elementos da Lista

- Para apresentar os elementos de uma lista, chamamos comumente de iterar. Na sequência, apresentaremos duas formas de iterar uma lista.

```

1 lista = ['maçã', 'uva', 'banana']
2
3 ▼ for item in lista:
4     print(item)

```

```

1 lista = ['maçã', 'uva', 'banana']
2
3 ▼ for i in range(len(lista)):
4     print(lista[i])

```

07 Adicionar Elementos à Lista

- Para adicionar elementos à lista, temos 3 opções: **append()**, **insert()** e **extend()**.
- Vamos de append()**? O método *append()* adiciona no fim da lista, como no exemplo a seguir, na qual na linha 2 adicionamos a String 'Nathaly' à lista.

```

1 lista = ['Daniel','Arthur']
2 lista.append('Nathaly')
3
4 ▼ for item in lista:
5     print(item)

```

- O método `insert()` possibilita a inserção de um novo elemento em uma dada posição da lista.

```

1 lista = ['Daniel','Arthur']
2 lista.insert(2,'Nathaly')
3 lista.insert(0,'Joselita')

```

- No exemplo acima, na linha 2, inserimos 'Nathaly' na posição 2, que não estava ocupada. Veja a ilustração a seguir.

- Antes da execução da linha 2:

Valores →	'Daniel'	'Arthur'
Índices →	0	1

- Depois da execução da linha 2:

Valores →	'Daniel'	'Arthur'	'Nathaly'
Índices →	0	1	2

08 Exclusão de um Elemento da Lista

- Como vimos anteriormente, podemos excluir uma lista ou tupla INTEIRA. Entretanto, podemos excluir um elemento da lista ou um intervalo de elementos de uma lista. Este tipo de exclusão não funciona com tuplas, como veremos a seguir.

```

1 frutas = ['banana','uva','maçã']
2 del frutas[0]
3 print(frutas)
4
5 nomes = 'Daniel','Arthur','Nathaly'
6 del nomes[0]
7 print(nomes)

```



- No exemplo acima podemos verificar a exclusão de um elemento. Na linha 2, eliminamos o primeiro elemento (posição 0) de uma lista. Entretanto, a operação da linha 6 não é possível, vai dar erro, pois tuplas são imutáveis.
- Complementarmente, no exemplo abaixo, usamos a operação `del frutas[:2]`, na qual o dois pontos seguido do número 2 nos informa que, exclua da primeira posição, até a posição 2 (ou seja, não inclui a posição 2). Desta maneira, o resultado da linha 3 será `['maçã', 'melancia', 'kiwi']`

```

1 frutas = ['banana','uva','maçã','melancia','kiwi']
2 del frutas[:2] #exclui ATÉ a posição 2
3 print(frutas)

```

09 Métodos Filé de Listas

- Vamos aprender métodos bem úteis que podem ser aplicados a listas, sendo eles: `append()`, `insert()`, `remove()`, `pop()`, `clear()`, `sort()` e `reverse()`. São métodos *built-in*.
- A seguir temos um exemplo completo. Experimente por um print entre cada uma das linhas para entender melhor.
 - Na linha 2, adicionar um elemento ao fim da lista
 - Na linha 3, inserimos o valor 10 na posição 1
 - Na linha 4, removemos o elemento 10
 - Na linha 5, removemos o elemento da posição 2
 - Na linha 7, ordenamos os elementos de ordem crescente (do menor ao maior)
 - Na linha 9, usamos o mesmo método, porém ordenamos de maneira decrescente (do maior ao menor)

```

1 lista1 = [20, 30, 40]
2 lista1.append(50) #insere no final
3 lista1.insert(1,10) #insere 10 na posição 1
4 lista1.remove(10) #remove o elemento 10
5 lista1.pop(2) #remove o elemento da posição 2
6 lista1.insert(0,70) #insere 70 na posição 0
7 lista1.sort() #ordena de modo crescente
8 #vai ficar [20,30,50,70]
9 lista1.sort(reverse=True) #ordena de modo decrescente
10 #vai ficar [70,50,30,20]
11 print(len(lista1))

```

10 Resumo de Listas

```
listaFeira = ['maca','banana','uva']
```

`listaFeira[0]` `listaFeira[1]` `listaFeira[2]`

Método	Descrição	Exemplo
<code>append(elemento)</code>	Adiciona ao fim da lista	<code>lista.append(12)</code>
<code>clear()</code>	Remove todos elementos	<code>lista.clear()</code>
<code>count(x)</code>	Quantas 'x' tem na lista?	<code>lista.count('x')</code>
<code>index(x)</code>	Posição de X na lista?	<code>lista.index('x')</code>
<code>insert(i,elemento)</code>	Add 'X' na posição i	<code>lista.insert(0,'X')</code>
<code>pop()</code>	Remove o último elemento	<code>lista.pop()</code>
<code>remove(x)</code>	Remove o elemento 'x'	<code>lista.remove('x')</code>
<code>reverse()</code>	Inverte os elementos	<code>lista.reverse()</code>
<code>sort()</code>	Ordena de modo crescente	<code>lista.sort()</code>
<code>sort()</code>	Ordena de modo decrescente	<code>lista.sort(reverse=True)</code>

