



Listas e Estruturas de Repetição em Python

Agenda

Tópicos:

- **Operações Básicas em Listas**
- **Fatias de Listas (Slices)**
- **Operador del**
- **Laços de Repetição Simples**
- **Iterando por Itens em Sequências**
- **Operador in**
- **Textos São Sequências!**
- **Procurando Ajuda**



Operações Básicas em Listas

Criando listas vazias

```
>>> lista = []  
>>> lista = list()
```

Operações Básicas em Listas

Criando listas preenchidas

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista = [3, 'ABC', 5]
```

**Listas são
heterogêneas**

Operações Básicas em Listas

Obtendo a quantidade de elementos da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> len(lista)
```

```
4
```

Operações Básicas em Listas

Obtendo um elemento da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista[0]
```

```
8
```

```
>>> lista[1]
```

```
9
```

```
>>> lista[4]
```

```
>>> ???
```

Operações Básicas em Listas

Obtendo um elemento da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista[0]
```

```
8
```

```
>>> lista[1]
```

```
9
```

```
>>> lista[4]
```

```
>>> ???
```

Erro! O primeiro índice é 0 e o último é 3

Operações Básicas em Listas

Obtendo um elemento da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista[-1]
```

```
15
```

```
>>> lista[-2]
```

```
1
```



Operações Básicas em Listas

Obtendo um elemento da lista indiretamente

```
>>> lista = [8, 9, 1, 15]
```

```
>>> a = 3
```

```
>>> lista[a]
```

```
15
```

Operações Básicas em Listas

Alterando um elemento da lista

```
>>> lista = [8, 9, 1, 15]
>>> lista[0] = 14
>>> lista
[14, 9, 1, 15]
```

Operações Básicas em Listas

Alterando um elemento da lista indiretamente

```
>>> lista = [8, 9, 1, 15]
```

```
>>> a = 2
```

```
>>> lista[a] = 14
```

```
>>> lista
```

```
[8, 9, 14, 15]
```

Operações Básicas em Listas

Acrescentando um novo elemento ao final da lista

```
>>> lista = [8, 9, 1, 15]
>>> lista.append(12)
>>> lista
[8, 9, 1, 15, 12]
```

Operações Básicas em Listas

Acrescentando vários novos elementos ao final da lista

```
>>> lista = [8, 9, 1, 15]
>>> lista2 = [4, 5, 6]
>>> lista.extend(lista2)
>>> lista
[8, 9, 1, 15, 4, 5, 6]
```

Operações Básicas em Listas

Acrescentando um novo elemento a uma posição qualquer da lista

```
>>> lista = [8, 9, 1, 15]
>>> lista.insert(1, 12)
>>> lista
[8, 12, 9, 1, 15]
```

Operações Básicas em Listas

Obtendo e removendo o último elemento da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista.pop()
```

```
15
```

```
>>> lista
```

```
[8, 9, 1]
```


Operações Básicas em Listas

Obtendo e removendo o último elemento da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> a = lista.pop()
```

```
>>> a
```

```
15
```

```
>>> lista
```

```
[8, 9, 1]
```

Operações Básicas em Listas

Obtendo e removendo um elemento da lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista.pop(2)
```

```
1
```

```
>>> lista
```

```
[8, 9, 15]
```

Operações Básicas em Listas

Removendo todos os elementos da lista

```
>>> lista = [8, 9, 1, 15]
>>> lista.clear()
>>> lista
[]
```

Operações Básicas em Listas

Contando aparições de elementos na lista

```
>>> lista = [8, 9, 1, 8, 8, 15]
```

```
>>> lista.count(8)
```

```
3
```

```
>>> lista.count(10)
```

```
0
```

Operações Básicas em Listas

Ordenando todos os elementos da lista, de forma crescente

```
>>> lista = [8, 9, 1, 15]
>>> lista.sort()
>>> lista
[1, 8, 9, 15]
```

Operações Básicas em Listas

Ordenando todos os elementos da lista, de forma decrescente

```
>>> lista = [8, 9, 1, 15]
>>> lista.sort(reverse=True)
>>> lista
[15, 9, 8, 1]
```

Operações Básicas em Listas

Procurando por um elemento na lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista.index(15)
```

```
3
```

```
>>> lista.index(10)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ValueError: 10 is not in list
```

Operações Básicas em Listas

Procurando por um elemento na lista

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista.index(15)
```

```
3
```

```
>>> lista.index(10)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ValueError: 10 is not in list
```

**Mais para frente
veremos como
tratar erros!**

Fatias de Listas (Slices)

A notação de fatia (slice) : tem inúmeras aplicações, apesar de parecer um pouco confusa a princípio...

Convém se acostumar com sua leitura, pois essa notação é responsável por parte do poder do Python!



Fatias de Listas (Slices)

Copiando um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista2 = lista[3:5]
```

```
>>> lista2
```

```
[15, 20]
```

Fatias de Listas (Slices)

Copiando um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista2 = lista[3:5]
```

```
>>> lista2
```

```
[15, 20]
```

Inclusivo

Exclusivo

Fatias de Listas (Slices)

Copiando um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista2 = lista[3:]
```

```
>>> lista2
```

```
[15, 20, 30, 2, 10]
```

Fatias de Listas (Slices)

Copiando um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista2 = lista[:5]
```

```
>>> lista2
```

```
[8, 9, 1, 15, 20]
```

Fatias de Listas (Slices)

Copiando um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista2 = lista[:]
```

```
>>> lista2
```

```
[8, 9, 1, 15, 20, 30, 2, 10]
```

Fatias de Listas (Slices)

Copiando um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista2 = lista[3:-2]
```

```
>>> lista2
```

```
[15, 20, 30]
```



Fatias de Listas (Slices)

Substituindo um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista[3:5] = [90, 91, 92, 93]
```

```
>>> lista
```

```
[8, 9, 1, 90, 91, 92, 93, 30, 2, 10]
```


Fatias de Listas (Slices)

Substituindo um trecho da lista

0, 1, 2, 3, 4, 5, 6, 7

```
>>> lista = [8, 9, 1, 15, 20, 30, 2, 10]
```

```
>>> lista[2:] = []
```

```
>>> lista  
[8, 9]
```

Operador del

O operador del é utilizado para excluir elementos ou fatias de listas

```
>>> lista = [8, 9, 1, 15]
>>> del lista[1]
>>> lista
[8, 1, 15]
```


Operador del

O operador del é utilizado para excluir elementos ou fatias de listas

```
>>> lista = [8, 9, 1, 15]
>>> del lista[1:3]
>>> lista
[8, 15]
```

Laços de Repetição Simples

Assim como as estruturas de tomada de decisão, laços de repetição precisam da identificação correta, e de uma pergunta válida



```
a = 0
while a < 5:
    print(a)
    a = a + 1
```

Saída na tela:

0
1
2
3
4

Laços de Repetição Simples

Assim como as estruturas de tomada de decisão, laços de repetição precisam da identificação correta, e de uma pergunta válida

```
lista = [8, 9, 1, 15]
a = 0
while a < len(lista):
    print(lista[a])
    a = a + 1
```

Saída na tela:

**8
9
1
15**

Laços de Repetição Simples

Assim como as estruturas de tomada de decisão, laços de repetição precisam da identificação correta, e de uma pergunta válida

```
lista = [8, 9, 1, 15]
a = 0
while a < len(lista):
    print(lista[a])
    a = a + 1
```

Saída na tela:

8
9
1
15

Apesar de funcionar, essa forma de percorrer uma lista não é a mais utilizada em Python... Veremos outra forma em breve...


Laços de Repetição Simples

A instrução break aborta o laço de repetição

```
bomba = int(input('Digite o número bomba: '))
lista = [8, 9, 1, 15]
a = 0
while a < len(lista):
    if bomba == lista[a]:
        print('BUUUMMMM!')
        break
    print(lista[a])
    a = a + 1
```

Operador in

O operador in é utilizado para trabalhar com o conteúdo de listas e sequências (como veremos em breve)



```
>>> lista = [8, 9, 1, 15]
>>> 9 in lista
>>> True
>>> 10 in lista
>>> False
```


Iterando por Itens em Sequências

A estrutura for das outras linguagens é parecida com o while. Mas em Python, o for serve apenas para iterar sobre itens de sequências

```
lista = [8, 9, 1, 15]  
for a in lista:  
    print(a)
```

Saída na tela:

8
9
1
15

Iterando por Itens em Sequências

A estrutura for das outras linguagens é parecida com o while. Mas em Python, o for serve apenas para iterar sobre itens de sequências

Exclusivo

```
for a in range(5):  
    print(a)
```

Saída na tela:

0

1

2

3

4

Iterando por Itens em Sequências

A estrutura for das outras linguagens é parecida com o while. Mas em Python, o for serve apenas para iterar sobre itens de sequências

Inclusivo

Exclusivo

```
for a in range(4, 9):  
    print(a)
```

Saída na tela:

4

5

6

7

8

Iterando por Itens em Sequências

A estrutura for das outras linguagens é parecida com o while. Mas em Python, o for serve apenas para iterar sobre itens de sequências

Inclusivo

Exclusivo

```
for a in range(4, 9, 2):  
    print(a)
```

Saída na tela:

4

6

8

Iterando por Itens em Sequências

A estrutura for das outras linguagens é parecida com o while. Mas em Python, o for serve apenas para iterar sobre itens de sequências

```
lista = [8, 9, 1, 15]  
for a in range(len(lista)):  
    print(lista[a])
```

Saída na tela:

**8
9
1
15**

Iterando por Itens em Sequências

Outra utilidade do for e das sequências é criar listas preenchidas de forma procedural

```
>>> lista = list(range(8))  
>>> lista  
[0, 1, 2, 3, 4, 5, 6, 7]
```

Iterando por Itens em Sequências

Outra utilidade do for e das sequências é criar listas preenchidas de forma procedural

```
>>> lista = list(range(4, 8))  
>>> lista  
[4, 5, 6, 7]
```

Iterando por Itens em Sequências

Outra utilidade do for e das sequências é criar listas preenchidas de forma procedural

```
>>> lista = list(range(4, 8))
```

```
>>> lista
```

```
[4, 5, 6, 7]
```

```
>>> lista = list(2 * x for x in range(4, 8))
```

```
>>> lista
```

```
[8, 10, 12, 14]
```


Iterando por Itens em Sequências

Outra utilidade do for e das sequências é criar listas preenchidas de forma procedural

```
>>> lista = list(0 for x in range(4))
```

```
>>> lista
```

```
[0, 0, 0, 0]
```

```
>>> lista = list(1 for x in range(3))
```

```
>>> lista
```

```
[1, 1, 1]
```



**Toda vez?! Ele deve
gostar desse slide...**




Sim!!! Mas...

Vamos praticar!!!

Textos São Sequências!


Vimos alguns exemplos com listas, mas algumas operações de listas também podem ser utilizadas com textos...



```
>>> texto = 'Abcd'
>>> texto[2]
'c'
```

Textos São Sequências!

Vimos alguns exemplos com listas, mas algumas operações de listas também podem ser utilizadas com textos...



```
>>> texto = 'Abcd'
>>> texto[1:]
'bcd'
```

Textos São Sequências!

Vimos alguns exemplos com listas, mas algumas operações de listas também podem ser utilizadas com textos...

Diferente de listas, textos são imutáveis!!!

```
>>> texto = 'Abcd'
```

```
>>> texto[1] = 'X'
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'str' object does not support item assignment

Textos São Sequências!

Vimos alguns exemplos com listas, mas algumas operações de listas também podem ser utilizadas com textos...

```
texto = 'Abcd'  
for a in texto:  
    print(a)
```

Saída na tela:

**A
b
c
d**

Procurando Ajuda

**Para mais informações sobre listas, sequências,
tuplas, conjuntos e dicionários:**

docs.python.org/3/tutorial/datastructures.html

Procurando Ajuda

[**www.python.org/doc**](http://www.python.org/doc)

[**docs.python.org/3/reference**](http://docs.python.org/3/reference)

[**docs.python.org/release/3.7.3/reference**](http://docs.python.org/release/3.7.3/reference)

[**docs.python.org/3/library**](http://docs.python.org/3/library)

[**docs.python.org/release/3.7.3/library**](http://docs.python.org/release/3.7.3/library)

[**docs.python.org/3/tutorial/introduction.html**](http://docs.python.org/3/tutorial/introduction.html)

[**stackoverflow.com**](http://stackoverflow.com)

[**pt.stackoverflow.com**](http://pt.stackoverflow.com)



Ficamos por aqui!