
Listas e Repetições

cubos
//academy//

As variáveis que vimos até agora
são do tipo **simples**. Vamos
conhecer uma variável do tipo
composta

Listas

**Listas permitem o
armazenamento de vários
valores em uma só variável**

Listas



Caixa de uvas



Caixa de abacaxis

Exemplo

Suponha que você precisa armazenar os dados de precipitação de chuva em uma cidade

```
● ● ●  
  
#utilizando variáveis simples  
chuva1 = 7.3  
chuva2 = 0.5  
chuva3 = 9.0  
chuva4 = 1.3  
#utilizando lista  
chuvas = [7.3, 0.5, 9.0, 1.3]
```

Listas



Caixa de frutas

Exemplo

É possível criar uma lista com elementos de **tipos diferentes!**



```
infos_aluno = [1, "Cubos Academy",  
"Joaozinho", [10, 9, 8.9], 8.7, True]
```

Indices

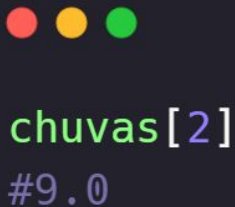
Os elementos individuais de uma lista podem ser acessados através de seus índices (posições).



```
chuvas = [7.3, 0.5, 9.0, 1.3]
```

The image shows a terminal window with a dark background. At the top left, there are three colored circles (red, yellow, green). Below them, the text `chuvas = [7.3, 0.5, 9.0, 1.3]` is displayed. Underneath each element in the list, there is a white curly bracket pointing to an index below the list: `0` for 7.3, `1` for 0.5, `2` for 9.0, and `3` for 1.3.

Se quero acessar o item 9.0:



```
chuvas[2]  
#9.0
```

The image shows a terminal window with a dark background. At the top left, there are three colored circles (red, yellow, green). Below them, the text `chuvas[2]` is displayed. Below that, the output `#9.0` is shown.

Métodos em Listas

append()

Adiciona um item no final da lista alterando a variável.



```
notas = [8, 9, 10, 8.7]
notas.append(9.7)
notas
#[8, 9, 10, 8.7, 9.7]
```

count()

Retorna a **quantidade de itens da lista** que correspondem ao **elemento desejado**.



```
notas = [8, 7, 9, 10, 10, 7, 10, 8, 10]  
notas.count(10)  
#4
```

insert()

Insere um item em **uma posição** específica desejada na lista.



```
#lista de vogais
vogais = ["a", "e", "i", "u"]
#inserir no índice 3 a letra "o"
vogais.insert(3, "o")
vogais
#['a', 'e', 'i', 'o', 'u']
```

index()

Retorna **a posição** onde um **elemento** se encontra na lista.

Atenção: Retorna somente o índice da primeira aparição do elemento na lista.



```
vogais = ["a", "e", "i", "o", "u"]  
vogais.index("i")  
#2
```



```
notas = [8, 7, 9, 10, 10, 7, 10, 8, 10]  
notas.index(10)  
#3
```

len()

Retorna o **tamanho da lista**, ou seja, a **quantidade de elementos** presentes nela.

Strings são listas de caracteres 😊



```
infos_aluno = [1, "Cubos Academy", [10, 9, 8.9], 8.7, True]
```

#importante! só funciona desta forma:

```
len(infos_aluno)
```

```
#5
```

Strings e Listas

Strings e Listas

Strings e Listas possuem algumas **similaridades**:

- São sequências, a lista de itens e a string de caracteres;
- É possível acessar as partes através de índices;
- Ambos tem tamanho;
- É possível converter strings em listas.



```
txt = 'Programação Python'
txt[0]
#'P'
```


split()

Recorta uma string em várias utilizando um **caractere** para definir o lugar do corte.

Retorna uma lista de strings.



```
txt = 'Programação+Python'  
txt.split("+")  
#['Programação', 'Python']
```

find()

Procura o **índice** dos **pedaços de texto** específicos dentro de uma string.

Retorna a primeira posição que corresponde ao texto passado.



```
txt = 'Programação em Python é tudo  
de bão'
```

```
txt2 = 'Python é tudo de bão'
```

```
print(txt.find('ão')) #9
```

```
print(txt2.find('ão')) #18
```

Vamos ver como efetuar
comparações

Na prática, no Colab!

cubos
//academy//

Repetições

cubos
//academy//

A programação é uma grande aliada na automatização de tarefas repetitivas

cubos
//academy//

Loops ou Laços

Estruturas de repetição que permitem executar mais de uma vez um mesmo trecho de código

Podem ser úteis para:

- Repetir uma mesma operação sobre todos os elementos de uma lista
- Executar um comando quantas vezes for necessário

Loops / Laços

Tipos de Laços:

- com quantidade determinada de repetições
- com quantidade indeterminada de repetições

Aplicações

- Imprimir individualmente todos os itens de uma lista
- Testar quais itens de uma lista satisfazem a uma determinada condição
- Imprimir uma mensagem até que o usuário faça a ação desejada

For

cubos
//academy//

Laço for

- Laço com **quantidade determinada** de repetições
- Repetição acontece **ao longo de uma lista**
- **Exemplo de Aplicação:**
Para todos os elementos de determinado conjunto, repita determinado comando



```
for item in lista:  
    comando a ser executado sobre cada  
    item
```

Exemplo

Imprimir cada item de uma lista

```


vogais = ["a", "e", "i", "o", "u"]
for letra in vogais:
    print(letra)
```

```

#a
#e
#i
#o
#u
```

Exemplo

Executar comandos dentro do loop



```
vogais = ["a", "e", "i", "o", "u"]  
for letra in vogais:  
    maiuscula = letra.upper()  
    print(maiuscula)
```

#A

#E

#I

#O

#U

Exemplo

Executar comandos dentro do loop

```
notas = [6.4, 7, 9, 10, 10, 5, 10, 8]
for nota in notas:
    passou = (nota >= 7)
    print(passou)

#False
#True
#True
#True
#True
#False
#True
#True
```

While

Laço while

- Laço com **quantidade indeterminada** de repetições
- Repetição acontece quando uma **condição é satisfeita**
- **Exemplo de Aplicação:**
Enquanto determinada condição é satisfeita, repita determinado comando

```
while condição:  
    comando a ser executado enquanto a  
    condição é satisfeita
```

Exemplo

Imprimir cada item de um contador

```
repetir = True
contador = 0

while (repetir):
    print(contador)
    if (contador >= 5):
        repetir = False
    contador += 1

# 0
# 1
# 2
# 3
# 4
# 5
```


Exemplo

Executar comandos dentro do loop

```
repetir = True
alvo = 5

while (repetir):
    digitado = input('Digite um número de 1 a 10: ')
    print('Digitou: ' + digitado)
    if (int(digitado) == alvo):
        print('Acertou!')
        repetir = False
    else:
        print('Errou, tente novamente :/')

# Digite um número de 1 a 10: 3
# Digitou: 3
# Errou, tente novamente :/

# Digite um número de 1 a 10: 5
# Digitou: 5
# Acertou!
```

Exemplo

Imprimir cada **item** de uma **lista**

```
● ● ●  
  
vogais = ["a", "e", "i", "o", "u"]  
tamanho = len(vogais)  
i = 0  
  
while i <= tamanho-1:  
    print(vogais[i])  
    i+=1
```



Helena Guimarães
Professora Cubos Academy

cubos
// academy //

www.cubos.academy