

Laboratório de Programação

Aula 04 - Listas

Prof.: Robson Gomes

Variáveis “Mutáveis” e “Imutáveis”

```
x = 1
```

```
y = x
```

```
print("Valor de x antes da mudança:", x)
```

```
print("Valor de y antes da mudança:", y)
```

```
x = 3
```

```
print("Valor de x depois da mudança :", x)
```

```
print("Valor de y depois da mudança:", y)
```

Valor de x antes da mudança: 1

Valor de y antes da mudança: 1

Valor de x depois da mudança: 3

Valor de y depois da mudança: 1

```
x = 1  
print(id(x))
```

```
x = x + 1  
print(id(x))
```

Posição na memória de x antes da mudança: 2305958346992
Posição na memória de x depois da mudança: 2305958347024

```
x = [1, 2, 3]
```

```
y = x
```

```
print("Lista x antes do append:", x)
```

```
print("Lista y antes do append:", y)
```

```
x.append(4)
```

```
print("Lista x depois do append :", x)
```

```
print("Lista y depois do append:", y)
```

Lista x antes do append: [1, 2, 3]

Lista y antes do append: [1, 2, 3]

Lista x depois do append : [1, 2, 3, 4]

Lista y depois do append: [1, 2, 3, 4]

Variáveis “Mutáveis” e “Imutáveis”

Aula 04

```
a = [1,2,3]
```

```
b = a
```

```
print("Lista a antes da mudança:", a)
```

```
print("Lista b antes da mudança:", b)
```

```
b[1] = "x"
```

```
print("Lista a depois da mudança:", a)
```

```
print("Lista b depois da mudança:", b)
```

Lista a antes da mudança: [1, 2, 3]

Lista b antes da mudança: [1, 2, 3]

Lista a depois da mudança: [1, 'x', 3]

Lista b depois da mudança: [1, 'x', 3]

```
def ordenaLista(lista):  
    lista.sort()  
    return lista
```

Lista antes da função: [3, 2, 1]
Lista depois da função: [1, 2, 3]

```
x = [3, 2, 1]  
print("Lista antes da função: ", x)  
ordenaLista(x)  
print("Lista depois da função: ", x)
```



**Como saber quais
variáveis são
“Mutáveis” ou
“Imutáveis” em Python?**

Em relação à mutabilidade, Python pode ser classificado em:

Mutáveis: os conteúdos podem ser alterados (lists, dict, set, file)

Imutáveis: os conteúdos não podem ser alterados (str, bool, float, int, long, complex, tuple)

Listas em

Python

LISTAS

- Como uma string, uma lista é uma sequência de valores. Em uma string, os valores são caracteres; em uma lista, eles podem ser de qualquer tipo.. Para definir uma lista são usados colchetes []

```
listaDeDados = ['0', 1 , '2', 3 , '4', 5]
```

```
print (listaDeDados)
```

```
print (listaDeDados[1])
```

```
print (listaDeDados[0:2]) #imprime de onde até onde?
```

```
print (listaDeDados[0:14]) #e se extrapolar pra mais?
```

```
print (listaDeDados[:2]) #os dois primeiros
```

```
print (listaDeDados[2:]) #tudo menos os dois primeiros
```

```
print (listaDeDados[-1]) #o último
```

```
print (listaDeDados[-2]) #o penúltimo
```

```
print (listaDeDados[-2:]) #os dois últimos
```

```
print (listaDeDados[:-2]) #todos menos os dois últimos
```



LISTAS

- Agora tem outras possibilidades. Podemos manipular os dados.

```
listaDeDados = ['0', 1, '2', 3, '4', 5]
```

```
print (listaDeDados[1])
```

```
listaDeDados[1] = "numero"
```

```
print (listaDeDados[1])
```



LISTAS

- Mas... Nada de extrapolar!

```
listaDeDados = ['0', 1, '2', 3, '4', 5]
```

```
print (listaDeDados[6]) #também não pode extrapolar
```



LISTAS — FATIAMENTO (SLICING)

```
lista = ["CEULP", 5, "23", 100]
```

```
nova = lista[:2]
```

```
print ("Mudança 1:", nova)
```

```
nova = lista[2:]+["a",3*10]
```

```
print ("Mudança 2:", nova)
```

```
nova = 3*lista[:3]
```

```
print ("Mudança 3:", nova)
```

Mudança 1: ['CEULP', 5]

Mudança 2: ['23', 100, 'a', 30]

Mudança 3: ['CEULP', 5, '23', 'CEULP', 5, '23', 'CEULP', 5, '23']



LISTAS — UM POUCO MAIS

- Trabalhando com a Lista
- Criando Lista vazia
`listaVazia = []`
- Criando Lista com somente um valor
`listaComUm = [um]`
- Tamanho da Lista: `len`
`print (len (listaVazia))`
`print (len (listaComUm))`



LISTAS ANINHADAS

```
a = ["p", "q", "r"]  
b = ["t", a, "u"]
```

```
print ("Tamanho de b:", len(b))
```

```
print ("b:", b)
```

```
print ("b[1]:", b[1])
```

```
print ("b[1][0]:", b[1][0])
```

Tamanho de b: 3

b: ['t', ['p', 'q', 'r'], 'u']

b[1]: ['p', 'q', 'r']

b[1][0]: p



LISTAS - COMANDOS

- **append(x)** -- insere x no fim da lista

```
lista1 = ["a", "b", "c"]
```

```
print ("lista1 antes do append:", lista1)
```

```
lista1.append("f")
```

```
print ("lista1 depois do append:", lista1)
```

lista1 antes do append: ['a', 'b', 'c']

lista1 depois do append: ['a', 'b', 'c', 'f']



LISTAS - COMANDOS

- **append(x)** -- e se eu inserir uma lista x no fim da lista?

```
lista1 = ["a", "b", "c"]
```

```
lista2 = ["d", "e"]
```

```
print ("lista1 antes do append:", lista1)
```

```
lista1.append(lista2)
```

```
print ("lista1 depois do append:", lista1)
```

lista1 antes do append: ['a', 'b', 'c']

lista1 depois do append: ['a', 'b', 'c', ['d', 'e']]



LISTAS - COMANDOS

- **extend (L)** -- insere a **lista** L no fim da lista original

```
lista1 = ["a", "b", "c"]
```

```
lista2 = ["d", "e"]
```

```
print ("lista1 antes do extend:", lista1)
```

```
lista1.extend(lista2)
```

```
print ("lista1 depois do extend:", lista1)
```

lista1 antes do extend: ['a', 'b', 'c']

lista1 depois do extend: ['a', 'b', 'c', 'd', 'e']



LISTAS - COMANDOS

- **insert(i, x)** -- insere x na posição i
- O primeiro argumento é o índice do elemento antes do qual será feita a inserção

```
lista1 = ["a", "b", "c"]
```

```
print ("lista1 antes do insert:", lista1)
```

```
lista1.insert(2,"k")
```

```
print ("lista1 depois do insert:", lista1)
```

lista1 antes do insert: ['a', 'b', 'c']

lista1 depois do insert: ['a', 'b', 'k', 'c']



LISTAS - COMANDOS

- **remove(x)** -- remove o primeiro item encontrado na lista cujo valor é igual a x. Se não existir valor igual, uma exceção ValueError é levantada.

```
lista1 = ["a", "b", "c"]
```

```
print ("lista1 antes do remove:", lista1)
```

```
lista1.remove("b")
```

```
print ("lista1 depois do remove:", lista1)
```

lista1 antes do remove: ['a', 'b', 'c']

lista1 depois do remove: ['a', 'c']



LISTAS - COMANDOS

- `remove(x)` -- mostrando erro

```
lista1 = ["a", "b", "c"]  
  
print ("lista1 antes do remove:", lista1)  
  
lista1.remove("f")  
  
print ("lista1 depois do remove:", lista1)
```

Traceback (most recent call last):
File "tuplas.py", line 95, in <module>
lista1.remove("f")
ValueError: list.remove(x): x not in list



LISTAS - COMANDOS

- **pop(i)** -- Remove o item na posição dada e o retorna. Se nenhum índice for especificado, o comando remove e retorna o último item na lista. **i é opcional**

```
lista1 = ["a", "b", "c"]
```

```
print ("lista1 antes do pop:", lista1)
```

```
x = lista1.pop(1)
```

```
print ("lista1 depois do pop:", lista1)
```

```
print ("Valor de x:", x)
```

lista1 antes do pop: ['a', 'b', 'c']

lista1 depois do pop: ['a', 'c']

Valor de x: b



LISTAS - COMANDOS

- **pop(i)** -- testando sem o i

```
lista1 = ["a", "b", "c"]
```

```
print ("lista1 antes do pop:", lista1)
```

```
x = lista1.pop()
```

```
print ("lista1 depois do pop:", lista1)
```

```
print ("Valor de x:", x)
```

lista1 antes do pop: ['a', 'b', 'c']

lista1 depois do pop: ['a', 'b']

Valor de x: c



LISTAS - COMANDOS

- **index(x)** -- retorna o índice do primeiro item cujo valor é igual a x, gerando a exceção ValueError se o valor não existir

```
lista1 = ["a", "b", "c"]
```

```
indice = lista1.index("b")
```

```
print ("indice=", indice)
```

indice= 1



LISTAS - COMANDOS

- **count(x)** -- devolve o número de vezes que o valor *x* aparece na lista.

```
lista1 = ["a", "b", "c", "b"]
```

```
print ("Quantidade de \"b\" =", lista1.count("b"))
```

```
print ("Quantidade de \"t\" =", lista1.count("t"))
```

Quantidade de "b" = 2

Quantidade de "t" = 0



LISTAS - COMANDOS

- `sort()` -- ordena os itens na própria lista.

```
lista1 = ["x", "b", "v", "b"]
```

```
print ("lista1 antes do sort:", lista1)
```

```
lista1.sort()
```

```
print ("lista1 depois do sort:", lista1)
```

lista1 antes do sort: ['x', 'b', 'v', 'b']

lista1 depois do sort: ['b', 'b', 'v', 'x']



LISTAS - COMANDOS

- **reverse()** -- inverte a ordem dos elementos na lista.

```
lista1 = ["x", "b", "v", "b"]
```

```
print ("lista1 antes do reverse:", lista1)
```

```
lista1.reverse()
```

```
print ("lista1 depois do reverse:", lista1)
```

lista1 antes do reverse: ['x', 'b', 'v', 'b']

lista1 depois do reverse: ['b', 'v', 'b', 'x']



LISTAS - COMANDOS

- **del(i)** -- remove o elemento pelo índice sem retorná-lo. Atente-se que a sintaxe muda. Permite eliminar slices também

```
lista1 = ["x", "b", "v", "b"]
```

lista1 antes do del: ['x', 'b', 'v', 'b']

```
print ("lista1 antes do del:", lista1)
```

lista1 depois do primeiro del: ['x', 'v', 'b']

```
del lista1[1]
```

lista1 depois do segundo del: ['b']

```
print ("lista1 depois do primeiro del:", lista1)
```

```
del lista1[0:2]
```

```
print ("lista1 depois do segundo del:", lista1)
```



LISTAS - CUIDADOS

```
lista1 = ["a", "b", "c"]
```

```
lista2 = lista1
```

```
print ("lista1 antes do append:", lista1)
```

```
print ("lista2 antes do append", lista2)
```

```
lista1.append("*****")
```

```
print ("lista1 depois do append:", lista1)
```

```
print ("lista2 depois do append", lista2)
```

```
lista2[0]="Abracadabra"
```

```
print ("lista1 depois da atribuição:", lista1)
```

```
print ("lista2 depois da atribuição", lista2)
```

lista1 antes do append: ['a', 'b', 'c']

lista2 antes do append ['a', 'b', 'c']

lista1 depois do append: ['a', 'b', 'c', '***']**

lista2 depois do append ['a', 'b', 'c', '***']**

lista1 depois da atribuição: ['Abracadabra', 'b', 'c', '***']**

lista2 depois da atribuição ['Abracadabra', 'b', 'c', '***']**



UM POUCO MAIS DE LISTAS (NA VERDADE, É A DESCULPA PARA FALAR DO **FOR**)

- O **for** só funciona para estruturas em que você sabe quantas vezes haverá a repetição, tipo listas.

```
lista=["a", 1, "x+y", 1+2]
for i in lista:
    print (i)
```

```
a
1
x+y
3
```



FOR (COM A FUNÇÃO RANGE())

```
for i in range(10):  
    print(i)
```

0
1
2
3
4
5
6
7
8
9



FOR (COM A FUNÇÃO RANGE())

Testar no for:

`range(5,10)`

`range(0,10,2)`



FOR

```
lista=["uva", "pera", "maça"]  
for i in range(len(lista)):  
    print (i, "-", lista[i])
```

0 - uva
1 - pera
2 - maça



FOR EM LISTAS- CUIDADOS

```
lista=["1","2","3"]
```

```
for i in lista:  
    lista.append(i)  
    print(i)
```



Laboratório de Programação

Aula 04 - Listas

Prof.: Robson Gomes