

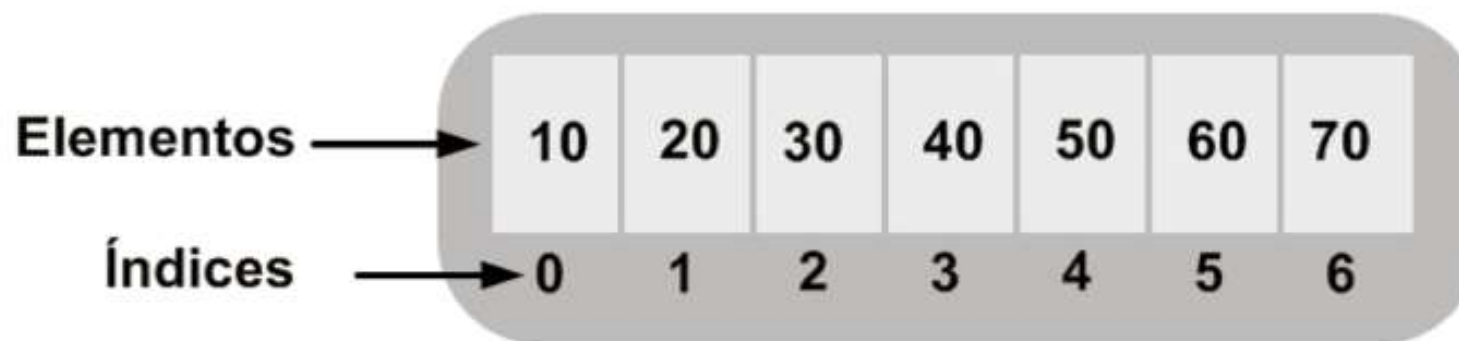
IPE – Introdução a Programação Estruturada

Aula 06 – Listas

Profº Ms Gustavo Molina

Listas em Python

Uma lista é um conjunto ordenado de valores (como um vetor em outras linguagens), onde cada valor é identificado por um índice. Os índices são iniciados em zero e atribuídos sequencialmente a partir deste. Os valores que compõem uma lista são chamados elementos.



Listas em Python

- As listas são similares a strings, que são conjuntos ordenados de caracteres, com a diferença de que os elementos de uma lista podem possuir qualquer tipo, ou seja, a lista pode conter quaisquer valores, incluindo valores de tipos mistos e até outras listas.

Lista_01.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_01.py > ...

```
1  a= [] #lista vazia
2  b = [10,20,30] #lista de elementos inteiros
3  c = [30,4.5,"CC"] #lista mista
4  d = [b,c] # lista contendo outras listas
5
6  print(a)
7  print(b)
8  print (c)
9  print (d)
```

Acessando Valores de uma Lista

- Para acessar um elemento específico de uma lista, usamos o nome da lista seguido do índice entre colchetes.

```
Lista_02.py X
E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_02.py > ...
1  a= [] #lista vazia
2  b = [10,20,30] #lista de elementos inteiros
3  c = [30,4.5,"CC"] #lista mista
4  d = [b,c] # lista contendo outras listas
5
6  print(b)
7  print (b[0])
8  print (c[2])
```

```
[10, 20, 30]
10
CC
```

Modificando Valores de uma Lista

- Para modificar o valor de uma lista, basta atribuir ao índice o novo valor.

Lista_03.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_03.py > ...

```
1  a= [] #lista vazia
2  b = [10,20,30] #lista de elementos inteiros
3  c = [30,4.5,"CC"] #lista mista
4  d = [b,c] # lista contendo outras listas
5
6  print(b)
7
8  b [0] = 40
9  b [1] = 50
10 b [2] = 60
11
12 print(b)
```

Adição de Elementos - Append

- Uma das principais vantagens de trabalharmos com listas é poder adicionar novos elementos durante a execução do programa. Para adicionarmos **um elemento no fim** da lista, utilizamos o método **append**.

Lista_Append.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Append.py > ...

```
1  b =[10,20,30]
2  print (b)
3  b.append(40)
4  print (b)
```

```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Programs/Python/Python38/python.exe "e:/UNIP/2020/2º Semestre/IPE/Aula 6/Lista_Append.py"
[10, 20, 30]
[10, 20, 30, 40]
```

Adição de Elementos - Extend

- Outra maneira de adicionar um elemento a uma lista é usando o método **extend**. Esse método **adiciona vários elementos ao final da lista**.

Lista_Extend.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Extend.py > ...

```
1 g =[10,20,30]
2 print (g)
3 g.extend([50,60,70,80,90,100])
4 print (g)
```

```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Programs/Python/Python38/python.exe "e:/UNIP/2020/2º Semestre/IPE/Aula 6/Lista_Extend.py"
[10, 20, 30]
[10, 20, 30, 50, 60, 70, 80, 90, 100]
```

Adição de Elementos - Insert

Outra maneira de adicionar um elemento a uma lista é utilizando o método `insert(P,V)`, onde:

P -> posição onde será inserido o elemento;

V -> valor que será inserido.

Lista_Insert.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Insert.py > ...

```
1  vingadores=["Capitão América", "Homem de Ferro", "Hulk"]
2  print (vingadores)
3  vingadores.insert(3, "Doutor Estranho")
4  vingadores.insert(0, "Thor")
5  print (vingadores)
```

```
['Capitão América', 'Homem de Ferro', 'Hulk']
['Thor', 'Capitão América', 'Homem de Ferro', 'Hulk', 'Doutor Estranho']
```


Remoção de Elementos - Remove

- Para remover um elemento da lista, basta utilizar o método **remove**. Esse método **remove a primeira ocorrência** de um elemento na lista. **Resulta em erro, caso não exista o elemento.**

Lista_Remove.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Remove.py > ...

```
1  vingadores =["Capitão América", "Homem de Ferro", "Hulk", "Hulk"]
2  print (vingadores)
3  vingadores.remove("Hulk")
4  vingadores.remove ("Capitão América")
5  print (vingadores)
```

```
['Capitão América', 'Homem de Ferro', 'Hulk', 'Hulk']
['Homem de Ferro', 'Hulk']
```

Remoção de Elementos - Pop

Outro método utilizado para remover um elemento da lista é o `pop(P)`, onde `P` é a posição na qual se deseja excluir o item. Se usado sem o valor `pop()`, remove o último elemento da lista.

Lista_Pop.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Pop.py > ...

```
1 vingadores = ["Capitão América", "Homem de Ferro", "Hulk", "Thor", "Viúva Negra", "Homem Aranha"]
2 print (vingadores)
3 vingadores.pop()
4 vingadores.pop(0)
5 print (vingadores)
```

```
['Capitão América', 'Homem de Ferro', 'Hulk', 'Thor', 'Viúva Negra', 'Homem Aranha']
['Homem de Ferro', 'Hulk', 'Thor', 'Viúva Negra']
```

Remoção de Elementos - Del

Outra maneira de remover um elemento da lista é utilizando o método `del<nome da lista>[P]`, onde P é a posição na qual se deseja excluir o item. A Figura 54 mostra o exemplo de utilização desse método.

Lista_Pop.py •

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Pop.py > ...

```
1  vingadores = ["Capitão América", "Homem de Ferro", "Hulk", "Thor", "Viúva Negra", "Homem Aranha"]
2  print (vingadores)
3  vingadores.pop()
4  vingadores.pop(0)
5  print (vingadores)
```

```
['Capitão América', 'Homem de Ferro', 'Hulk', 'Thor', 'Viúva Negra', 'Homem Aranha']
['Homem de Ferro', 'Hulk', 'Thor', 'Viúva Negra']
```

Tamanho da Lista

Para definirmos o tamanho de uma lista, basta utilizarmos a função `len`. O valor retornado é igual ao número de elementos da lista, conforme é apresentado na Figura 48.

Lista_Tamanho.py

E: > UNIP > 2020 > 2º Semestre > IPE > Aula 6 > Lista_Tamanho.py > ...

```
1 vingadores = ["Capitão América", "Homem de Ferro", "Hulk", "Thor", "Viúva Negra", "Homem Aranha"]
2 print (len (vingadores))
```

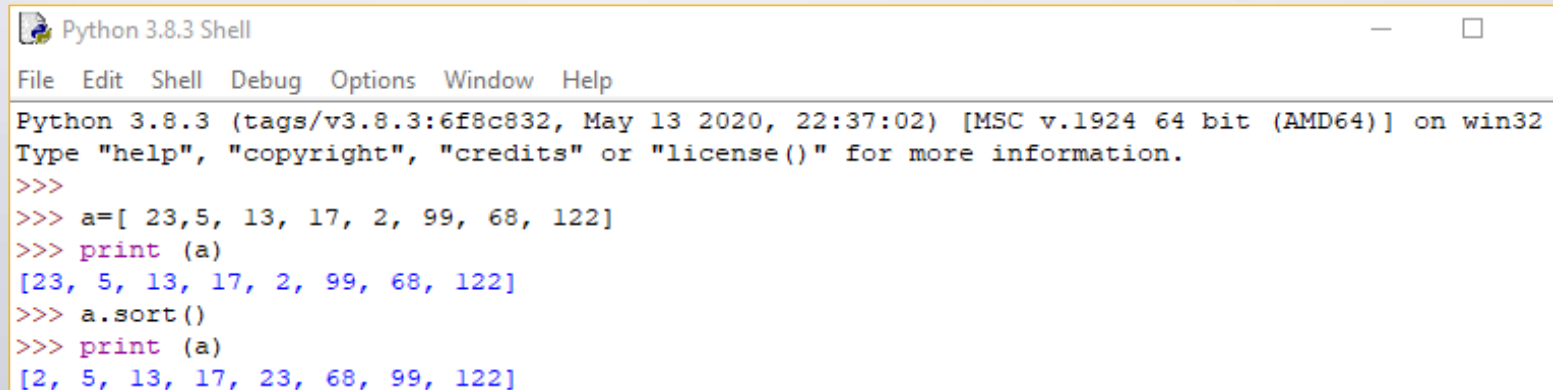
```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Programs/Python/Python38/python.exe "e:/UNIP/2020/2º Semestre/IPE/Aula 6/Lista_Tamanho.py"
```

6

Ordenando uma Lista

Para ordenar os elementos de uma lista de maneira crescente, basta utilizar o método `Sort()`. O método `Sort` utiliza o algoritmo de ordenação Timsort.

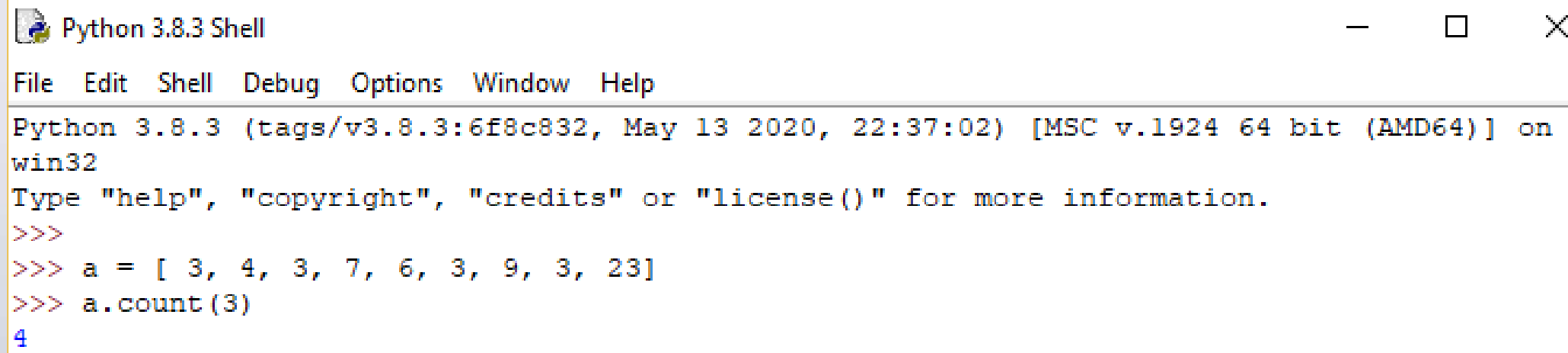
Timsort é um algoritmo de ordenação híbrido derivado do merge sort e do insertion sort, projetado para ter boa performance em vários tipos de dados do mundo real. Foi inventado por Tim Peters, em 2002, para ser usado na linguagem de programação Python, e tem sido o algoritmo de ordenação padrão do Python desde a versão 2.3. A Figura 55 mostra o exemplo de utilização desse método.

A screenshot of a Python 3.8.3 Shell window. The window has a title bar with the text 'Python 3.8.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a text editor with the following Python code:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> a=[ 23,5, 13, 17, 2, 99, 68, 122]
>>> print (a)
[23, 5, 13, 17, 2, 99, 68, 122]
>>> a.sort()
>>> print (a)
[2, 5, 13, 17, 23, 68, 99, 122]
```

Número de Ocorrência de Elementos em uma Lista

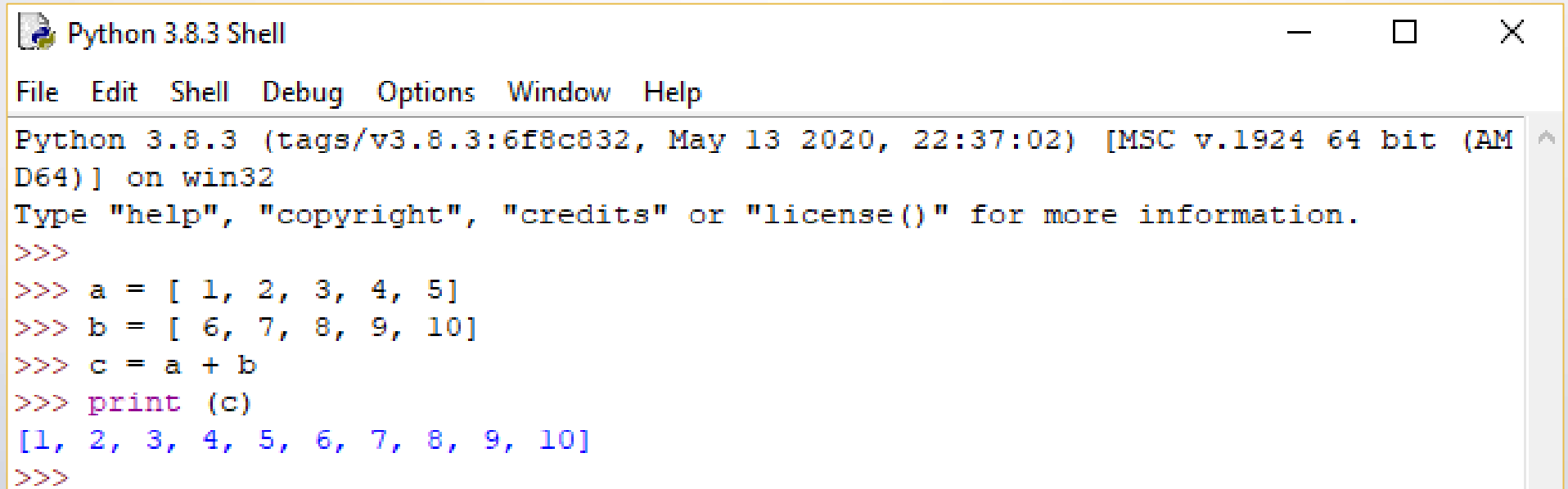
O método utilizado para retornar a quantidade de vezes que um elemento aparece na lista é `Count()`. A Figura 56 mostra o

A screenshot of a Python 3.8.3 Shell window. The window has a title bar with the text 'Python 3.8.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains the following text:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on  
win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
>>> a = [ 3, 4, 3, 7, 6, 3, 9, 3, 23]  
>>> a.count(3)  
4
```

Operações em Listas - Concatenação

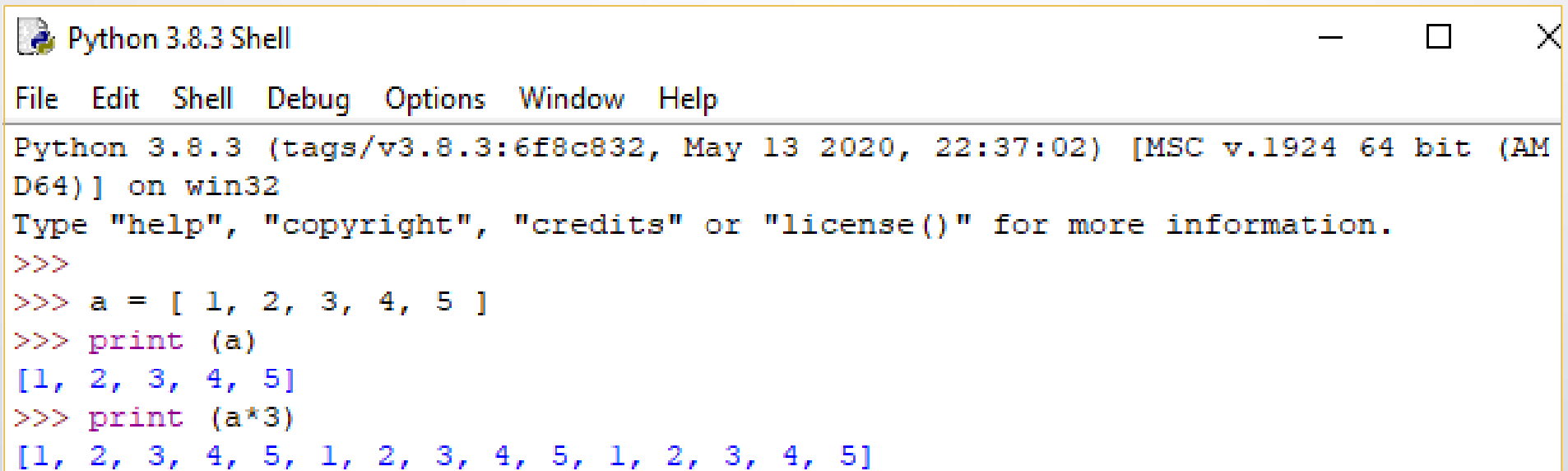
- O operador (+) é utilizado para **concatenar** listas.

A screenshot of a Python 3.8.3 Shell window. The window has a title bar with the text 'Python 3.8.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window displays the following text:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> a = [ 1, 2, 3, 4, 5]
>>> b = [ 6, 7, 8, 9, 10]
>>> c = a + b
>>> print (c)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>
```

Operações em Listas - Concatenação

- O operador (*) é utilizado para **repetir uma lista X vezes**.

A screenshot of a Python 3.8.3 Shell window. The window has a title bar with the text 'Python 3.8.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains the following text:

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> a = [ 1, 2, 3, 4, 5 ]
>>> print (a)
[1, 2, 3, 4, 5]
>>> print (a*3)
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```


Operador In

- Para saber se um elemento **pertence** a uma lista, utilizamos o **operador in**.

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> a = [1,2,3,4,5]
>>> 2 in a
True
>>> 10 in a
False
.
```

Listas Aninhadas

- Uma lista aninhada é uma lista que aparece como elemento de outra lista. No exemplo abaixo, o terceiro elemento é uma lista aninhada.

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> a = [10,20,[30,40]]
>>> print (a)
[10, 20, [30, 40]]
>>> print (a[2][0]) #extraindo um elemento da lista aninhada
30
```

Dúvidas???

