

ALGORITMIA E ESTRUTURAS DE DADOS

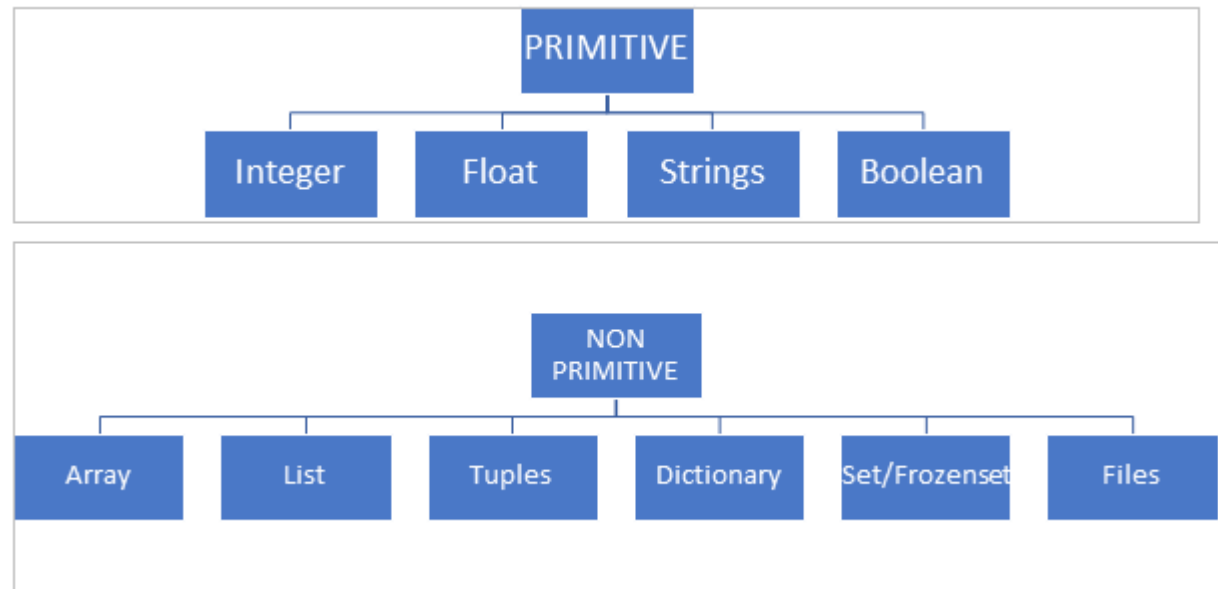
MÓDULO IV LISTAS

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

- ☐ Listas bidimensionais
- ☐ Listas tridimensionais

❖ Coleções

- ❑ Estruturas de dados **não primitivas**, compostas
- ❑ São compostas por uma conjunto de items (dados), onde cada um ocupa uma determinada posição na coleção de dados



❖ Coleções

❑ A Linguagem Python implementa 4 tipos de coleções de dados:

- **LIST**

Coleção de dados ordenada e editável. Permite dados duplicados

- **TUPLE**

Coleção de dados ordenada e não editável. Permite dados duplicados

- **SET**

Coleção de dados não ordenada e não indexada. Não permite dados duplicados

- **DICTIONARY**

Coleção de dados não ordenada, indexada e editável. Não permite dados duplicados

❖ Listas | Conceito

- ❑ Estrutura de dados **não primitiva**, composta
- ❑ Consiste numa coleção de items, onde cada um ocupa uma determinada posição na estrutura de dados
- ❑ Os items de uma lista são separados por vírgula e colocados entre []
- ❑ Os items de uma lista são geralmente do mesmo tipo, mas podem ser de tipos de dados diferentes

```
list = [value1, value2, value3,...valueN]
```

❖ Listas | Conceito

❑ Exemplos de listas

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]
```

```
numeros = [1,2,3,4,5,6,7,8,9,10]
```

```
dados = [1, "António", "1º ano", 18]
```

```
print(nomes)
```

```
print(numeros)
```

```
print(dados)
```

C:\WINDOWS\py.exe

```
['António', 'Carlos', 'Fátima', 'Raquel']  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
[1, 'António', '1º ano', 18]
```

Uma lista pode conter dados de mais do que um tipo

❖ Índice de uma lista

- ❑ Posso aceder individualmente a cada uma das posições de uma lista, através de um **índice**, em que a **primeira posição** é a **0**

```
Exemplo_listas.py > ...  
1  
2  
3  nomes = ["António", "Carlos", "Fátima", "Raquel"]  
4  
5  print(nomes[0])  
6  print(nomes[1])  
7  
8  
9  
10  
11
```

C:\WINDOWS\py.exe
António
Carlos
-

❖ Índice de uma lista

- ❑ Posso aceder a um subconjunto de uma lista, especificando a posição inicial e a posição final da lista

Posição inicial Posição final (é excluída)

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]
```

```
print(nomes[0:2])
```

```
print(nomes[1:3])
```

```
print(nomes[:3])
```

```
print(nomes[1:])
```

C:\WINDOWS\py.exe

['António', 'Carlos']

['Carlos', 'Fátima']

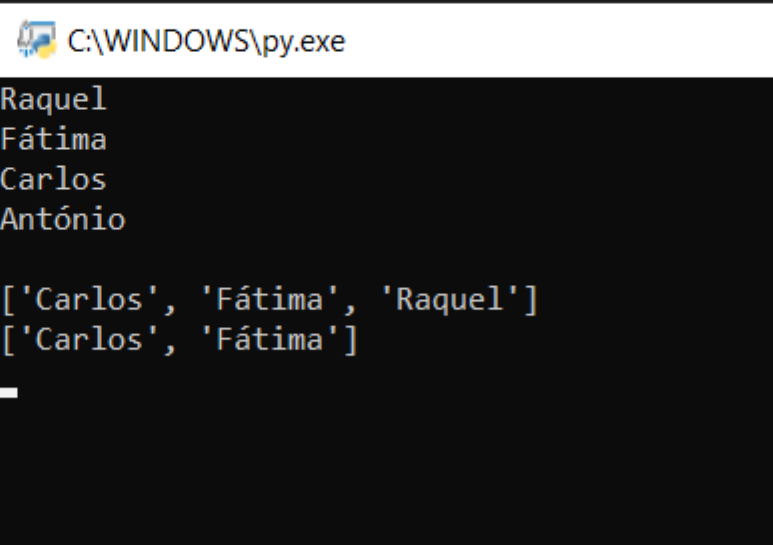
['António', 'Carlos', 'Fátima']

['Carlos', 'Fátima', 'Raquel']

❖ Índice de uma lista

- ❑ Posições **negativas** numa lista: identificam posições a partir **do final da lista**
- ❑ Assim, posição -1 refere-se à última posição da lista

```
exemplo_listas.py 7 ...  
1  
2  
3 nomes = ["António", "Carlos", "Fátima", "Raquel"]  
4  
5 print(nomes[-1])  
6 print(nomes[-2])  
7 print(nomes[-3])  
8 print(nomes[-4])  
9  
10 print()  
11  
12 print(nomes[-3:])  
13 print(nomes[-3:-1])  
14  
15  
16
```



C:\WINDOWS\py.exe

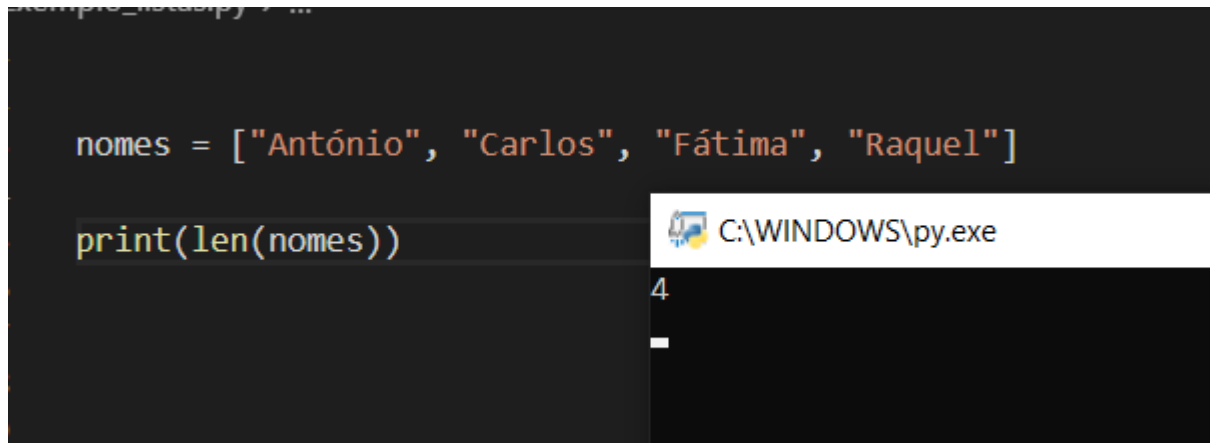
Raquel
Fátima
Carlos
António

['Carlos', 'Fátima', 'Raquel']
['Carlos', 'Fátima']
-

❖ Comprimento de uma lista

- ❑ Comprimento (tamanho) de uma lista: **len**
- ❑ Função **len()** devolve o comprimento (nº de items) de uma lista

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
print(len(nomes))
```



C:\WINDOWS\py.exe
4

A lista nome tem **4** posições, identificadas do índice **0** ao índice **3**

❖ Percorrer uma lista

- ❑ Percorrer lista com um ciclo for: cláusula **range**

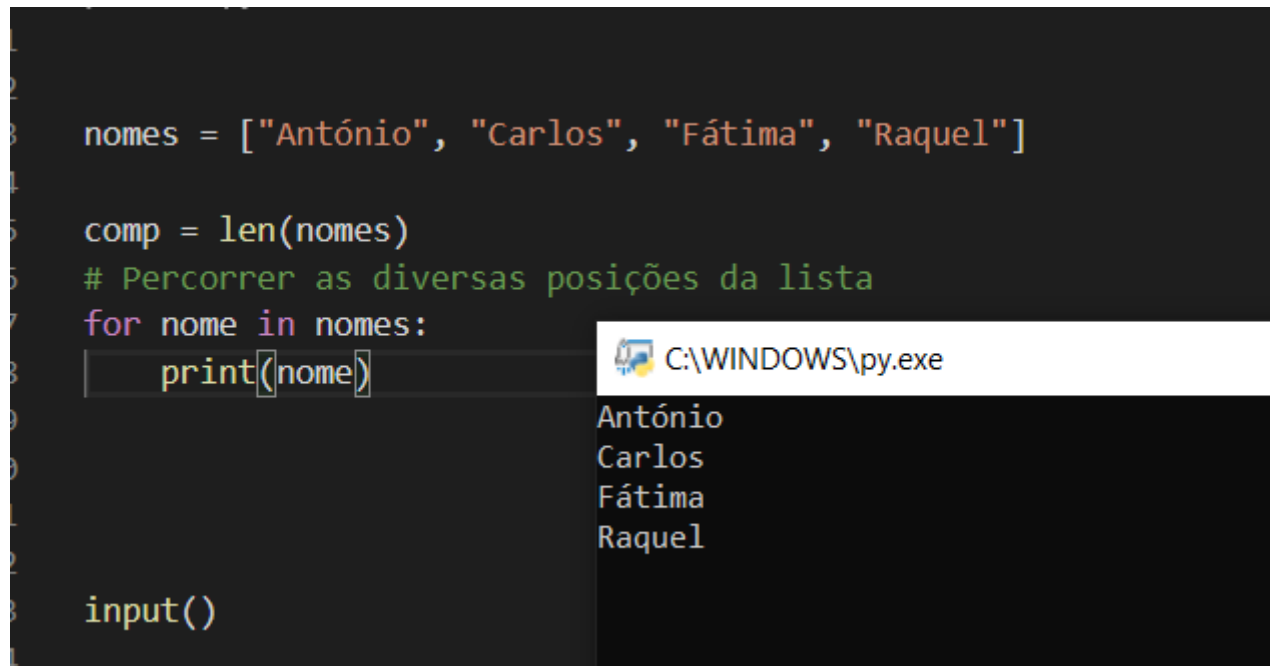
```
Exemplo_listas.py > ...  
1  
2  
3  nomes = ["António", "Carlos", "Fátima", "Raquel"]  
4  
5  comp = len(nomes)  
6  # Percorrer as diversas posições da lista  
7  for i in range (comp):  
8      print(nomes[i])  
9  
10  
11  
12
```

C:\WINDOWS\py.exe
António
Carlos
Fátima
Raquel

❖ Percorrer uma lista

- ❑ Percorrer lista com um ciclo for:

```
1 nomes = ["António", "Carlos", "Fátima", "Raquel"]
2
3
4
5 comp = len(nomes)
6 # Percorrer as diversas posições da lista
7 for nome in nomes:
8     print(nome)
9
10
11
12
13 input()
```



The image shows a screenshot of a Python script being executed. The script defines a list named 'nomes' with four elements: "António", "Carlos", "Fátima", and "Raquel". It then calculates the length of the list and uses a 'for' loop to iterate over each element, printing it. The output of the script is displayed in a separate window titled 'C:\WINDOWS\py.exe', showing the names 'António', 'Carlos', 'Fátima', and 'Raquel' on separate lines.

❖ Operadores in e not in 1. Se estiver em | V
2. Se não estiver em | A ou R L

❑ Operadores in e not in devolvem True ou False

```
nomes = ["Antônio", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
```

```
# converte uma string (ou um tuplo) numa lista  
name = input("Nome: ")
```

```
# Operador in 1  
if name in nomes:  
    print("O nome existe na lista")
```

```
# Operador not in 2  
if name not in nomes:  
    print("O nome não existe na lista")
```

C:\WINDOWS\py.exe

Nome: Raquel
O nome existe na lista

Importante

dica: caso seja necessario remover duplicados de uma lista

& não esquecer

❖ Listas | Métodos *built-in python*

❑ Alguns métodos que manipulam listas:

Método	Descrição
append	Acrescenta um elemento no final da lista
insert	Insere um elemento numa determinada posição (índice) na lista
remove	Remove um determinado elemento da lista (dado)
del	Remove o elemento em determinada posição (índice)
clear	Remove todos os elementos de uma lista
pop	Remove e devolve o último elemento da lista
count	Devolve o nº de elementos com um determinado valor
index	Devolve o índice do primeiro elemento com determinado valor

https://www.w3schools.com/python/python_lists.asp

<https://www.tutorialsteacher.com/python/python-list>

❖ Listas | Métodos *built-in python*

❑ Alguns métodos que manipulam listas:

Método	Descrição
copy	Devolve uma cópia da lista
list	Converte uma string numa lista
sort	Ordena a lista por ordem crescente
reverse	Inverte a ordenação da lista
max	Devolve o maior valor contido na lista
min	Devolve o menos valor contido na lista
sum	Devolve a soma dos elementos contidos numa lista

https://www.w3schools.com/python/python_lists.asp

<https://www.tutorialsteacher.com/python/python-list>

❖ Listas | Métodos *built-in python*

❑ Append(*value*)

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 15, 25, 40, 35, 12, 7]
```

```
# Acrescenta nome no final da lista  
name = input("Nome: ")  
nomes.append(name)  
  
# acrescenta numero no final da lista  
num = int(input("Número: "))  
numeros.append(num)  
  
print()  
print(nomes)  
print(numeros)
```

C:\WINDOWS\py.exe

Nome: Ernesto
Número: 27

```
['António', 'Carlos', 'Fátima', 'Raquel', 'Ernesto']  
[10, 20, 30, 5, 15, 25, 40, 35, 12, 7, 27]
```

Tip !

Nota: insere um elemento **no final** da lista

❖ Listas | Métodos *built-in python*

❑ `Insert(pos, value)`

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 15, 25, 40, 35, 12, 7]
```

```
# Acrescenta nome numa determinada posição  
name = input("Nome: ")  
nomes.insert(0, name)
```

```
# acrescenta numero numa determinada posição  
num = int(input("Número: "))  
numeros.insert(1, num)
```

```
print()  
print(nomes)  
print(numeros)
```

C:\WINDOWS\py.exe

Nome: Ernesto
Número: 27

```
['Ernesto', 'António', 'Carlos', 'Fátima', 'Raquel']  
[10, 27, 20, 30, 5, 15, 25, 40, 35, 12, 7]
```

Tip !

Nota: insere um elemento na lista, na **posição** indicada

❖ Listas | Métodos *built-in python*

❑ Remove(*value*)

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 15, 25, 40, 35, 12, 7]
```

```
# remove determinado elemento da lista  
name = input("Nome: ")  
nomes.remove(name)
```

```
# remove determinado elemento da lista  
num = int(input("Número: "))  
numeros.remove(num)
```

```
print()  
print(nomes)  
print(numeros)
```

C:\WINDOWS\py.exe

Nome: Carlos
Número: 20

```
['António', 'Fátima', 'Raquel']  
[10, 30, 5, 15, 25, 40, 35, 12, 7]
```

Tip !

Nota: remove a **primeira ocorrência de determinado valor**, no caso de existirem dados repetidos

❖ Listas | Métodos *built-in python*

❑ Del(*pos*)

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 15, 25, 40, 35, 12, 7]
```

```
# remove elemento em determinada posição
```

```
del nomes[0]  
del numeros[9]
```

```
print()  
print(nomes)  
print(numeros)
```

C:\WINDOWS\py.exe

```
['Carlos', 'Fátima', 'Raquel']  
[10, 20, 30, 5, 15, 25, 40, 35, 12]
```

Tip !

Nota: remove **determinada posição** na lista

❖ Listas | Métodos *built-in python*

❑ Clear()

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]
numeros = [10, 20, 30, 5, 15, 25, 40, 35, 12, 7]

# remove TODOS elementos da lista
nomes.clear()
numeros.clear()

print()
print(nomes)
print(numeros)
```

C:\WINDOWS\py.exe

```
[]
[]
```

Tip !

Nota: remove **todos** os elementos da lista

❖ Listas | Métodos *built-in python*

❑ Pop()

```
1
2
3  nomes = ["António", "Carlos", "Fátima", "Raquel"]
4  numeros = [10, 20, 30, 5, 15, 25, 40, 35, 12, 7]
5
6  # REMOVE e DEVOLVE o último elemento da lista
7  name = nomes.pop()
8  num = numeros.pop()
9
10 print()
11 print(name)
12 print(num)
13
14
15
```

C:\WINDOWS\py.exe

Raquel
7

Tip !

Nota: remove e devolve o **último** elemento da lista

❖ Listas | Métodos *built-in python*

❑ Count(*value*)

```
2
3  nomes = ["António", "Carlos", "Fátima", "Raquel"]
4  numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
5
6  name = input("Nome: ")
7  num = int(input("Número: "))
8
9  # Devolve o nº elementos com determinado valor
10 cont = nomes.count(name)
11 cont1 = numeros.count(num)
12
13 print()
14 print(name, "ocorre", cont, "vezes")
15 print(num, "ocorre", cont1, "vezes")
16
17
18
```

C:\WINDOWS\py.exe

Nome: Carlos
Número: 30

Carlos ocorre 1 vezes
30 ocorre 3 vezes

❖ Listas | Métodos *built-in python*

❑ Index (*value, start, end*)

```
2
3  nomes = ["António", "Carlos", "Fátima", "Raquel"]
4  numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
5
6  name = input("Nome: ")
7  num = int(input("Número: "))
8
9  # Devolve o índice da primeira ocorrência de determinado valor
0  pos = nomes.index(name)
1  pos1 = numeros.index(num)
2
3  print()
4  print(name, "ocorre na posição", pos)
5  print(num, "ocorre na posição", pos1)
6
7
8
```

C:\WINDOWS\py.exe

Nome: Carlos
Número: 30

Carlos ocorre na posição 1
30 ocorre na posição 2

Tip !

Nota: devolve a **posição da primeira ocorrência** do objeto de pesquisa

❖ Listas | Métodos *built-in python*

❑ Index (*value, start, end*)

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
```

```
num = int(input("Número: "))
```

```
# Devolve o índice da ocorrência de determinado valor numa lista
```

```
pos1 = numeros.index(num)
```

```
pos2 = numeros.index(num, pos1+1) # encontra a 2ª ocorrência de num
```

```
pos3 = numeros.index(num, pos2+1) # encontra a 3ª ocorrência de num
```

```
print()
```

```
print(num, "ocorre na posição", pos1, ",", pos2, " e ", pos3)
```

C:\WINDOWS\py.exe

Número: 30

30 ocorre na posição 2 , 4 e 7



No caso do objeto de pesquisa não existir, **devolve um erro!**

o método `find()` não existe nas listas!

❖ Listas | Métodos *built-in python*

❑ Copy()

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
```

```
# cria uma lista cópia da inicial  
nomes_backup = nomes.copy()
```

```
print()  
print(nomes_backup)
```

C:\WINDOWS\py.exe

```
['António', 'Carlos', 'Fátima', 'Raquel']
```

Tip !

Nota: método copy devolve uma lista idêntica à inicial. Não tem argumentos.

❖ Listas | Métodos *built-in python*

❑ List(*string*)

```
5
6
7 # converte uma string (ou um tuplo) numa lista
8 name = input("Nome: ")
9 letras = list(name)
10
11 print(letras)
```

C:\WINDOWS\py.exe

Nome: Margarida

['M', 'a', 'r', 'g', 'a', 'r', 'i', 'd', 'a']

Tip !

Nota: converte uma string numa lista

❖ Listas | Métodos *built-in python*

❑ sort | reverse

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
```

```
# ordena lista  
nomes.sort()  
numeros.sort()  
print(nomes)  
print(numeros)
```

```
# Inverte ordem da lista  
print()  
nomes.reverse()  
numeros.reverse()  
print(nomes)  
print(numeros)
```

C:\WINDOWS\py.exe

```
['António', 'Carlos', 'Fátima', 'Raquel']  
[5, 7, 10, 12, 20, 25, 30, 30, 30, 40]  
  
['Raquel', 'Fátima', 'Carlos', 'António']  
[40, 30, 30, 30, 25, 20, 12, 10, 7, 5]
```

Tip !

`Lista.sort(reverse=True)` => ordena por ordem decedente

❖ Listas | Métodos *built-in python*

❑ `max(lista)` | `min(lista)`

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
```

```
# Maior e menor valor de uma lista
```

```
print("maior valor")
```

```
print(max(nomes))
```

```
print(max(numeros))
```

```
print("\n\nmenor valor")
```

```
print(min(nomes))
```

```
print(min(numeros))
```

C:\WINDOWS\py.exe

maior valor

Raquel

40

menor valor

António

5

❖ Listas | Métodos *built-in python*

❑ `sum(lista)`

```
nomes = ["António", "Carlos", "Fátima", "Raquel"]  
numeros = [10, 20, 30, 5, 30, 25, 40, 30, 12, 7]
```

```
# Maior e menor valor de uma lista
```

```
print("maior valor")
```

```
print(max(nomes))
```

```
print(max(numeros))
```

```
print("\n\nmenor valor")
```

```
print(min(nomes))
```

```
print(min(numeros))
```

```
print("soma da lista:", sum(numeros))
```

C:\WINDOWS\py.exe

maior valor

Raquel

40

menor valor

António

5

soma da lista: 209

❖ Listas | bidimensionais

- ❑ Uma lista geralmente contém dados numa ordem linear, ou numa única dimensão.
- ❑ No entanto, há muitos sistemas no “mundo real” que são multidimensionais. Para visualizar esses dados precisamos de uma estrutura de dados multidimensional
- ❑ Uma lista bidimensional poder servir para representar uma imagem digital, um jogo de tabuleiro, etc.
- ❑ Uma lista bidimensional nada mais é do que uma lista de listas (uma forma de *Nested list*)
- ❑ Uma lista bidimensional pode também ser vista como uma tabela de 2 dimensões: linhas e colunas

```
myList = [0,1,2,3]
```

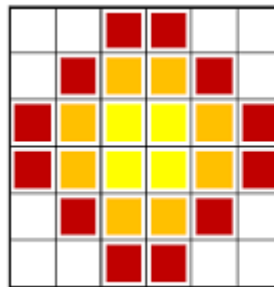
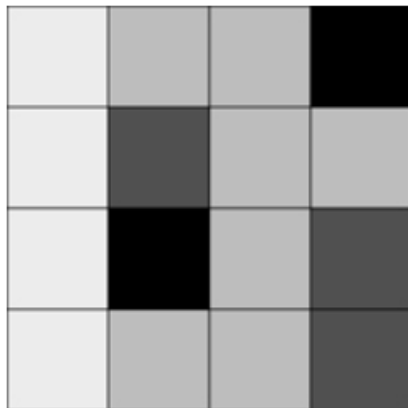
Lista unidimensional

```
myList = [ [0, 1, 2, 3],  
           [3, 2, 1, 0],  
           [3, 5, 6, 1],  
           [3, 8, 3, 4] ]
```

Lista bidimensional

❖ Listas | bidimensionais

❑ Representação de uma lista bidimensional:



11 11 00 00 11 11
11 00 01 01 00 11
00 01 10 10 01 00
00 01 10 10 01 00
11 00 01 01 00 11
11 11 00 00 11 11

```
myList = [ [236, 189, 189, 0],  
            [236, 80, 189, 189],  
            [236, 0, 189, 80],  
            [236, 189, 189, 80] ]
```



RGB de cada pixel, p.e.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad \text{Original matrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^T \Rightarrow \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

❖ Listas | bidimensionais

☐ Representação de uma lista bidimensional

Architecture of Multidimensional Collections in Java

	C0	C1	C2	C3	C4	C5	C6	C7	C8
R0	11	12	13	14	15	16	17	18	
R1	21	22	23	24	25	26			
R2	31	32	33	34	35	36	37		
R3	41	42	43	44	45				
R4	51	52	53	54	55	56	57	58	59
R5	61	62	63	64	65	66	67	68	

Each row can have different number of objects

Exemplo de representação de uma lista bidimensional

4 linhas	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	5 colunas				

Nº de elementos da list:
linhas x colunas = 20 elementos

❖ Listas | bidimensionais

- ❑ Iterar uma lista bidimensional como um objeto único

```
Exemplo1.py > ...  
1  # Listas bidimensionais  
2  
3  
4  lista = [[1,2,3], [4,5,6], [7,8,9]]  
5  print(lista)  
6  
7  
8  
9  
10
```

```
C:\WINDOWS\py.exe  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
_
```

❖ Listas | bidimensionais

- ❑ Iterar uma lista bidimensional linha a linha

```
Exemplo1.py > ...  
1  # Listas bidimensionais  
2  
3  
4  lista = [[1,2,3], [4,5,6], [7,8,9]]  
5  
6  # Iterar uma lista linha a linha  
7  for linha in lista:  
8      print(linha)  
9  
10  
11  
12  
13
```

C:\WINDOWS\py.exe

```
[1, 2, 3]  
[4, 5, 6]  
[7, 8, 9]  
_
```

❖ Listas | bidimensionais

- ❑ Iterar uma lista bidimensional através dos índices

Exemplo1.py > ...

```
1  # Listas bidimensionais
2
3
4  lista = [[1,2,3], [4,5,6], [7,8,9]]
5
6  # Iterar uma lista através dos índices
7  for i in range(len(lista)):      # len(lista) dá-nos o nº de linhas da lista
8      for j in range(len(lista[i])): # len(lista[i]) dá-nos o nº de elementos da linha i da lista
9          print(lista[i][j], end=" ")
10         print()
11
12
13
14
15
```

C:\WINDOWS\py.exe

```
1 2 3
4 5 6
7 8 9
```

❖ Listas | bidimensionais

Exemplo1.py > ...

```
1  # Listas bidimensionais
2
3
4  lista = []
5
6  lin = 3      # nº de linhas
7  col = 3      # nº de colunas ou nº de elementos em cada linha
8
9  # Iterar as diversas linhas da lista
10 for i in range(lin):
11     lista.append([])      # acrescenta uma lista vazia para cada linha
12     for j in range(col) :
13         numero = int(input("Número: "))
14         lista[i].append(numero) # acrescenta à lista o numero lido
15
16 print(lista)
17 # ou:
18 for linha in lista:
19     print(linha)
20
21
22
23
24 input()
25
```

C:\WINDOWS\py.exe

```
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
Número: 6
Número: 7
Número: 8
Número: 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

❖ Listas | bidimensionais

```
Exemplo1.py > cria_lista
1  import random
2
3  # Listas bidimensionais
4
5  def cria_lista(nlin, ncol):
6  # função que cria uma lista de valores aleatórios, com nlin linhas, e ncol elementos por cada linha
7      lista = []
8      for i in range(nlin):
9          lista.append([])          # acrescenta uma lista vazia para cada linha
10         for j in range(ncol) :
11             numero = random.randint(0,50)
12             lista[i].append(numero) # acrescenta à lista o numero aleatório
13     return lista
14
15
16
17 lista = cria_lista(3,3)
18 for linha in lista:
19     print(linha)
20
21
22
```

```
C:\WINDOWS\py.exe
[16, 6, 33]
[16, 21, 25]
[13, 16, 31]
```

❖ Listas | bidimensionais

❑ Funções *built-in Python* funcionam em lista de uma dimensão

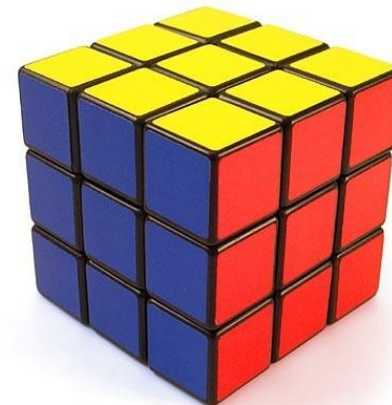
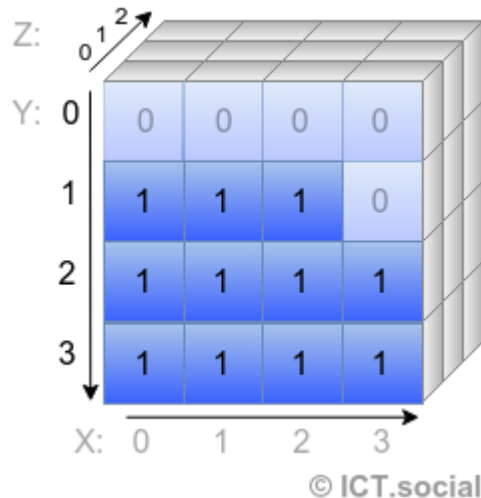
```
Exemplo1.py > ...
12         lista[i].append(numero) # acrescenta à lista o numero aleatório
13     return lista
14
15
16 lista = cria_lista(3,3)
17
18 # Funções para minipular listas funcionam em listas unidimensionais!
19 lista[0].sort()      # ordena a 1ª linha ordem ascendente
20
21 lista[1].sort()
22 lista[1].reverse()   # ordena a 2ª linha ordem descendente
23
24 for linha in lista:
25     print(linha)
26
27 numero = 2
28 pos = lista[0].index(numero)
29 print("{0} aparece na posição {1} ".format(numero, pos))
30
31 print("{0} aparece {1} vezes ".format(numero, lista[0].count(numero)))
32
33
34
```

C:\WINDOWS\py.exe

Numero: 1
Numero: 2
Numero: 2
Numero: 3
Numero: 4
Numero: 4
Numero: 5
Numero: 6
Numero: 6
[1, 2, 2]
[4, 4, 3]
[5, 6, 6]
2 aparece na posição 1
2 aparece 2 vezes

❖ Listas | tridimensionais

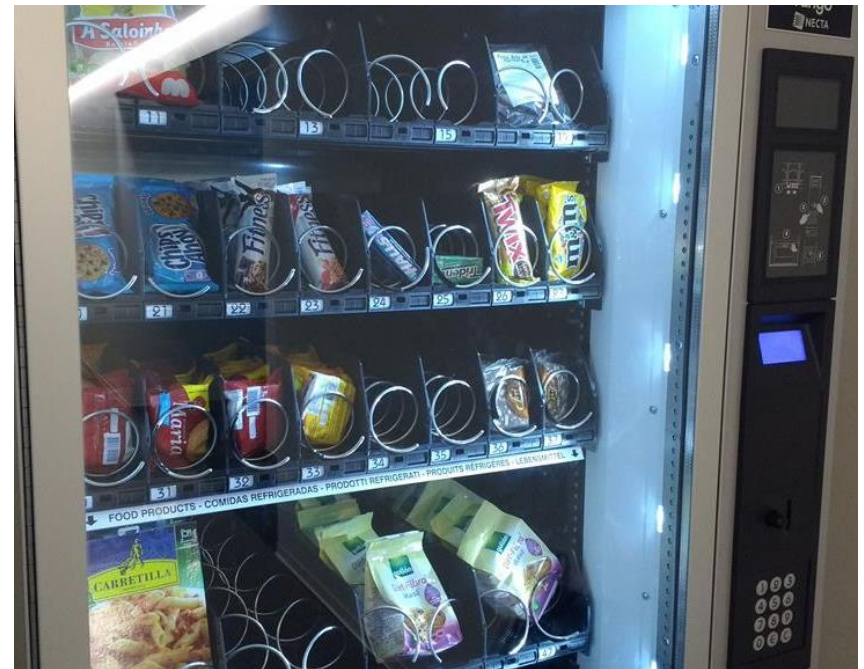
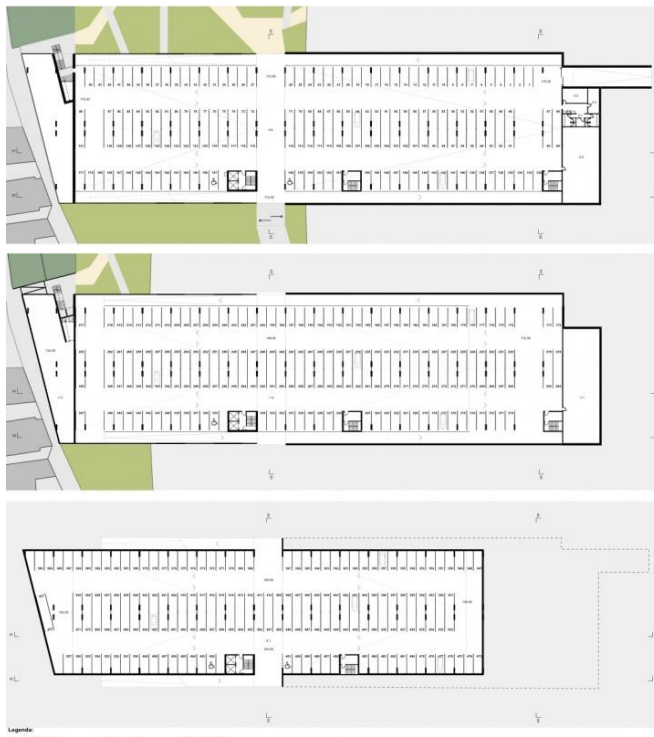
- ❑ Às vezes, pode ser útil criar uma lista com mais dimensões
- ❑ Todos nós podemos pelo menos imaginar uma lista 3D
- ❑ Uma lista tridimensional pode ser vista como uma lista contendo linhas, colunas e profundidade



❖ Listas | tridimensionais

❑ Exemplos de aplicabilidade:

- ❑ Parque de estacionamento com vários pisos, filas e lugares
- ❑ Representação de produtos numa máquina de *vending*
- ❑ Menus com 3 (ou mais) níveis de profundidade
- ❑ Etc.



❖ Listas | tridimensionais

```
Exemplo1.py > ...
4
5 def cria_lista(nlin, ncol, nprof):
6     # função que cria uma lista de valores aleatórios, com nlin linhas, e ncol elementos por cada linha
7     lista = []
8     for i in range(nlin):
9         lista.append([])          # acrescenta uma lista vazia para cada linha
10        for j in range(ncol) :
11            lista[i].append([])    # em cada linha, acrescenta uma coluna à lista
12            for k in range(nprof):
13                numero = input("Numero: ")
14                lista[i][j].append(numero) # Para cada linha / coluna, acrescenta dados à lista (profundidade)
15        return lista
16
17
18 lista = cria_lista(3,3, 3)
19
20 for linha in lista:
21     print("linha :", linha)
22
23
24
25
26 input()
27
```

C:\WINDOWS\py.exe

```
Numero: 12
Numero: 13
Numero: 14
Numero: 15
Numero: 16
Numero: 17
Numero: 18
Numero: 19
Numero: 20
Numero: 21
Numero: 22
Numero: 23
Numero: 24
Numero: 25
Numero: 26
Numero: 27
linha : [['01', '02', '03'], ['04', '05', '06'], ['07', '08', '09']]
linha : [['10', '11', '12'], ['13', '14', '15'], ['16', '17', '18']]
linha : [['19', '20', '21'], ['22', '23', '24'], ['25', '26', '27']]
```