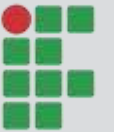




# Vetor e Matriz



**INSTITUTO  
FEDERAL**

Paraíba

---

Campus  
João Pessoa



## Problema

- Em diversas situações é necessário armazenar um grande volume de informações, o que torna impraticável a declaração de variáveis suficientes para armazenar esse grande volume.

### **Por exemplo:**

- Armazenar nomes de 30 pessoas;
- Armazenar a descrição de 1000 produtos;
- Armazenar a idade de 2.000.000 de pessoas;
- Armazenar os números (50) de um aposta da lotomania.

## ≡ Cenário para Análise

- Vamos partir de um programa, em Python, para ler 20 (vinte) números inteiros, calcular e exibir a média dos números lidos.
- Seria algo do tipo ...

```
soma = 0

for i in range(20):
    numero = int(input("Informe o " + str(i + 1) + " número: "))
    soma += numero

print("Média = %.2f" %(soma/20))
```



Ainda no mesmo cenário ...

E se eu quiser saber quais dos números lidos estavam acima da média?!



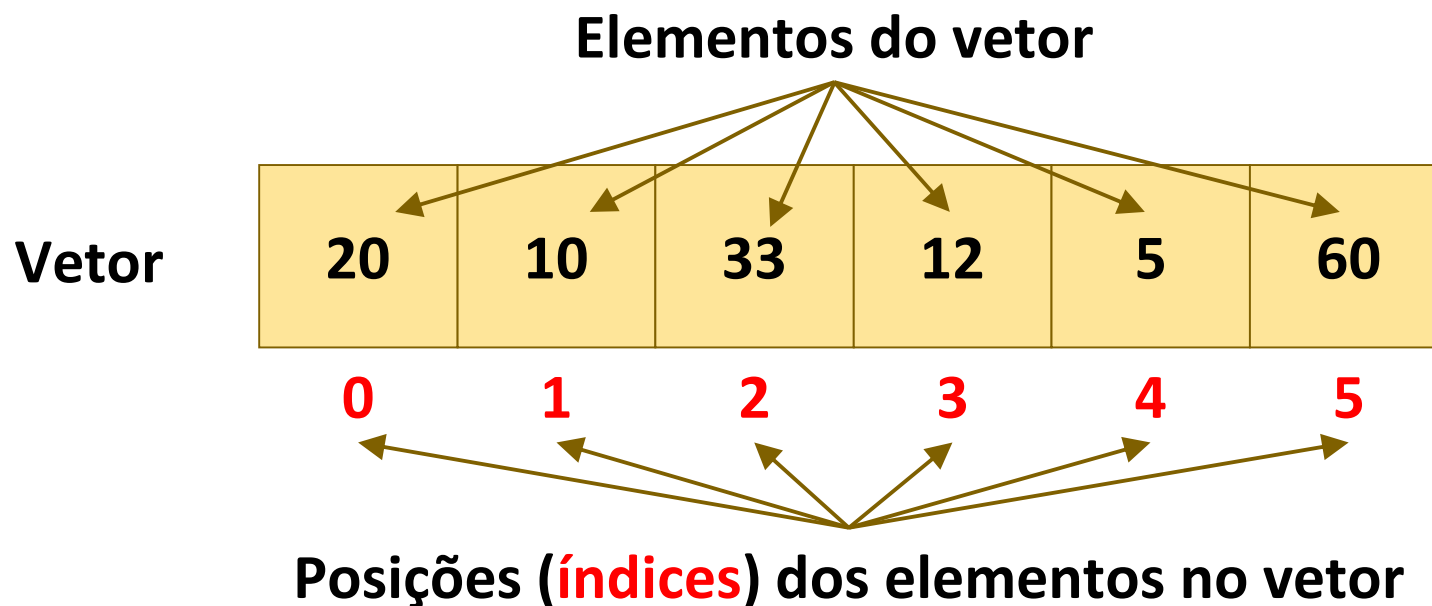
**Preciso declarar 20 variáveis?!?!?!?**

## ≡ Solução!

É possível definir uma estrutura de dados para armazenamento de vários valores com mesmo tipo.

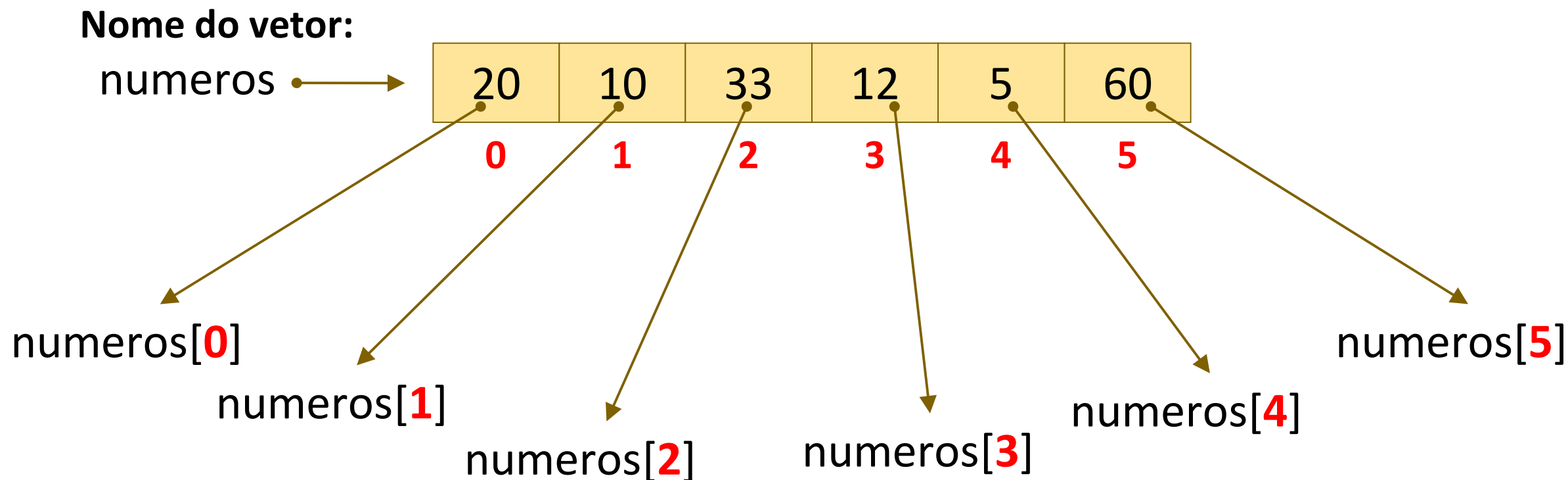
**Vetor**

Graficamente, representamos assim:



## ≡ Solução!

O **vetor** possui **identificador único** e cada elemento que o compõe pode ser **individualizado/referenciado por meio de um índice**



## ≡ Em Python...

Existem módulos que nos permitem manipular de forma eficiente vetores ... mas, nesse curso, **queremos aprender a representar um vetor** e a **definir operações** de manipulação sobre ele.

Em Python .... podemos implementar um vetor utilizando o conceito de listas : tipo *list()*

*Obs: os elementos de uma lista podem possuir qualquer tipo.*



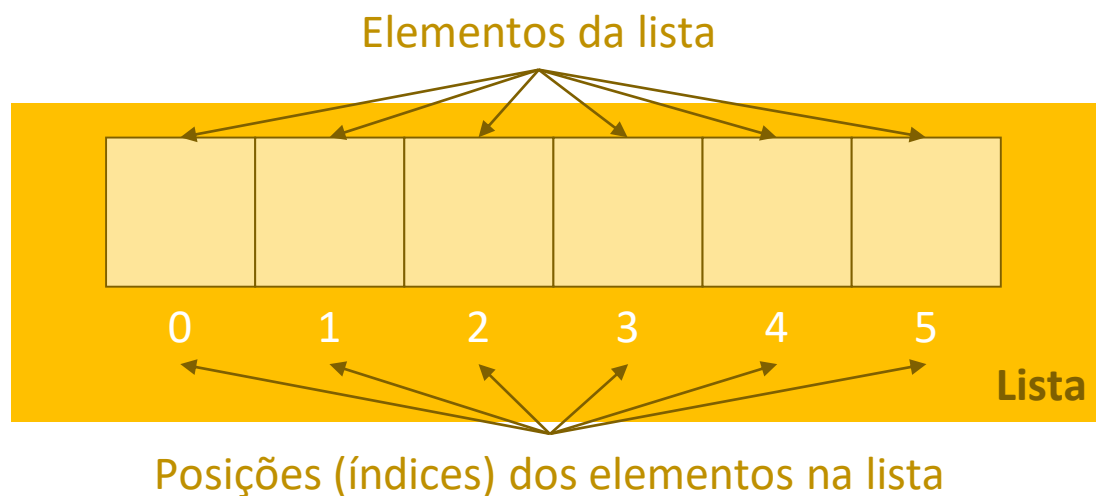
Lista



## ≡ Definição

- É um tipo especial que permite armazenar vários valores ao mesmo tempo!

**Graficamente, representamos assim:**



# ≡ Declaração

```
lista0 = [] # Lista vazia
lista1 = [1] # Lista com um elemento
lista2 = [1, 2] # Lista com dois elementos
lista3 = list()
lista4 = list(range(2, 11, 2))

# len - função para calcular a quantidade
# de elementos contidos na lista

print("%s - %d" %(lista0, len(lista0)))
print("%s - %d" %(lista1, len(lista1)))
print("%s - %d" %(lista2, len(lista2)))
print("%s - %d" %(lista3, len(lista3)))
print("%s - %d" %(lista4, len(lista4)))
```





Manipulação Básica



[ ]

Acessa (leitura ou escrita) os elementos da lista, através do índice (index).

Sintaxe:

`<id_lista>[<index>:[index]]`  
                                  ↑  
                                  opcional



Informar um índice  
inválido

## Exemplo

```
lista = [10, 20, 30, 40]
print (lista)
print (lista[0])
print (lista[2:])
print (lista[:2])
print (lista[1:3])
```

## Saída



# ≡ insert

Adiciona elemento na lista, informando a posição.

Sintaxe:

```
<id_list>.insert(<index>, <elemento>)
```

## Exemplo

```
lista = []  
  
lista.insert(0, 10)  
lista.insert(0, 20)  
lista.insert(0, 30)  
lista.insert(0, 40)  
  
print (lista)
```

## Saída



# ≡ append

Adiciona elemento no final da lista.

Sintaxe:

```
<id_list>.append(<elemento>)
```

## Exemplo

```
lista = []  
  
lista.append(10)  
lista.append(20)  
lista.append(30)  
lista.append(40)  
  
print (lista)
```

## Saída



# ≡ extend

Adiciona o(s) elemento(s) de uma lista “l2” no final de outra lista “l1”.

Sintaxe:

```
<id_list>.extend(<id_list>)
```

## Exemplo

```
lista1 = [10, 20]  
lista2 = [30, 40]  
  
print(lista1)  
print(lista2)  
lista1.extend(lista2)  
print(lista1)  
print(lista2)
```

## Saída





Concatena listas.

Sintaxe:

```
<id_list> += <id_list>
```

## Exemplo

```
lista1 = [10, 20]  
lista2 = [30, 40]  
  
print (lista1)  
print (lista2)  
lista1 += lista2  
print (lista1)
```

## Saída







Multiplica a lista por um inteiro N, gerando N cópias dos seus elementos.

Sintaxe:

```
<id_list> *= <id_list>
```

### Exemplo

```
lista=[10, 20, 30, 40]  
lista*=2  
print(lista)
```

### Saída



# ≡ del

Remove determinado elemento (ou elementos) da lista.

Sintaxe:

```
del(<id_list>[index:index])
```



Informar um índice  
inválido

## Exemplo

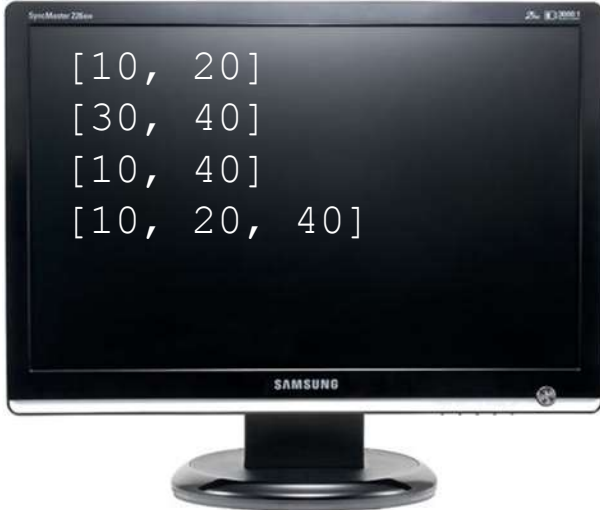
```
lista = [10, 20, 30, 40]
del(lista[2:])
print(lista)
```

```
lista = [10, 20, 30, 40]
del(lista[:2])
print(lista)
```

```
lista = [10, 20, 30, 40]
del(lista[1:3])
print(lista)
```

```
lista = [10, 20, 30, 40]
del(lista[2])
print(lista)
```

## Saída



```
[10, 20]
[30, 40]
[10, 40]
[10, 20, 40]
```



# remove

Remove determinado elemento da lista.

Sintaxe:

```
<id_list>.remove(elemento)
```



Informar um elemento  
que não existe na lista

## Exemplo

```
lista = [10, 10, 20, 20]  
print(lista)  
lista.remove(20)  
print(lista)
```

## Saída



# ≡ pop

Remove o último elemento da lista.

Sintaxe:

```
<id_list>.pop()
```



Tentar remover em uma  
lista que está vazia

## Exemplo

```
lista = [10, 20, 30, 40]  
print (lista.pop())  
print (lista.pop())  
print (lista.pop())  
print (lista.pop())
```

## Saída



# ≡ clear

Remove todos os elementos da lista.

Sintaxe:

```
<id_list>.clear()
```

## Exemplo

```
lista = [10, 20, 30, 40]
print (lista)
lista.clear()
print (lista)
```

## Saída





# len

Retorna o comprimento de uma lista.

Sintaxe:

```
len(<id_lista>)
```

## Exemplo

```
lista=[10,20,30,40]  
tamanho=len(lista)  
print(tamanho)
```

## Saída





# min

Retorna o menor valor em uma lista

Sintaxe:

```
min(<id_lista>)
```

## Exemplo

```
lista=[20,10,40,30]  
menor=min(lista)  
print(menor)
```

## Saída





# max

Retorna o maior elemento de uma lista.

Sintaxe:

```
max(<id_lista>)
```

## Exemplo

```
lista=[20,10,40,30]  
maior=max(lista)  
print(maior)
```

## Saída







# sum

Retorna a soma dos elementos de uma lista.

Sintaxe:

```
sum(<id_lista>)
```

## Exemplo

```
lista=[20,10,40,30]  
soma=sum(lista)  
print(soma)
```

## Saída





Manipulação Avançada

# ≡ count

Retorna a quantidade de determinado elemento na lista.

Sintaxe:

```
<id_list>.count(elemento)
```

## Exemplo

```
lista = [10, 20, 30, 30, 40]
print(lista)
print(lista.count(10))
print(lista.count(30))
print(lista.count(50))
```

## Saída



Verifica se um elemento está contido na lista. Retorna o index, se encontrado.

Sintaxe:

`<id_list>.index(<elemento>)`

Informar elemento que  
não existe na lista

## Exemplo

```
lista = [10, 20, 30, 40]
print (lista)
print (lista.index(10))
print (lista.index(20))
print (lista.index(30))
print (lista.index(40))
```

## Saída



# ≡ reverse

Inverte a ordem dos elementos na lista.

Sintaxe:

```
<id_list>.reverse()
```

## Exemplo

```
lista = [10, 20, 30, 40]
print(lista)
lista.reverse()
print(lista)
```

## Saída



# ≡ sort

Ordena os elementos da lista.

Sintaxe:

```
<id_list>.sort(reverse=<boolean>)
```

↑  
opcional

## Exemplo

```
lista = [20, 40, 10, 30]

print(lista)
lista.sort()
print(lista)
```

## Saída





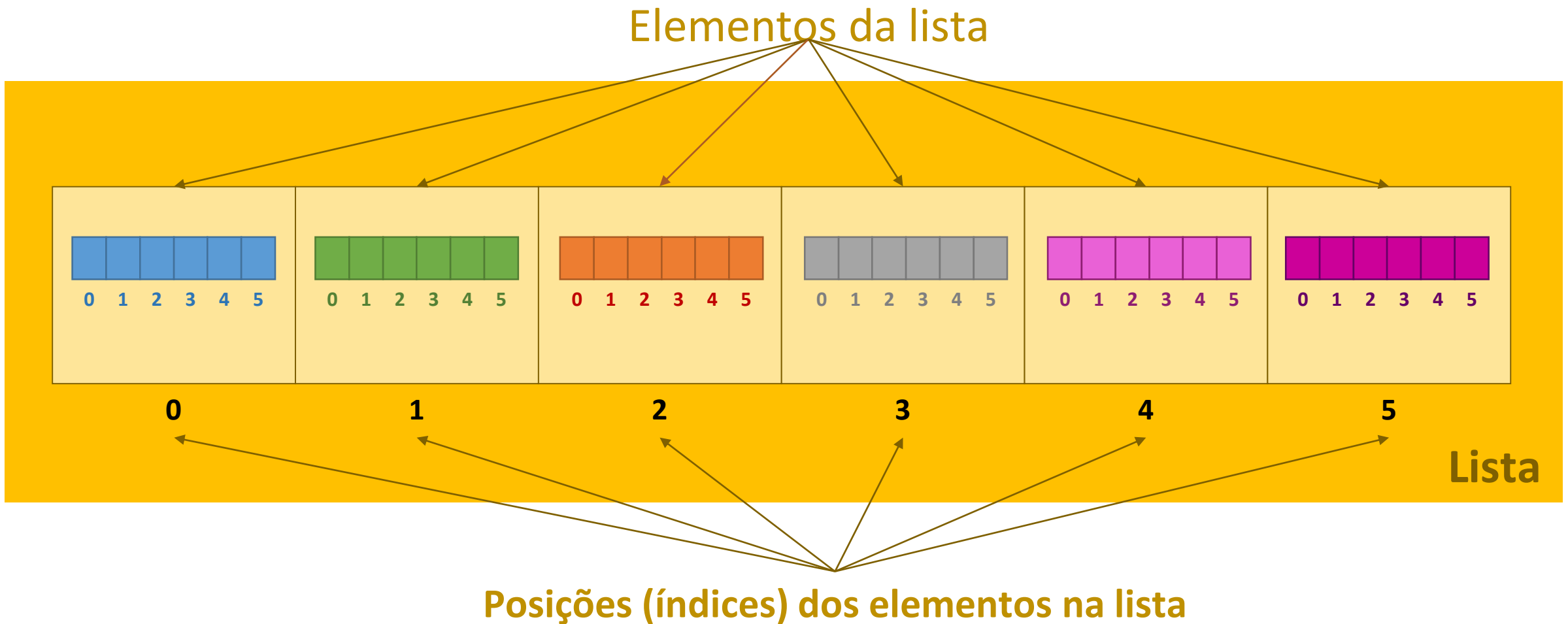
## Listas de Listas



## Listas Multidimensionais

- Uma lista, em Python, pode ser formada por elementos individuais heterogêneos, ou seja, de tipos diferentes;
- Também pode ser formada uma lista de listas, também conhecida como **matriz!**







# Matriz

Nome da Matriz:  
numeros

Elementos da Matriz

10	22	2	43	1	28
0	1	2	3	4	5
15	51	4	7	67	90
0	1	2	3	4	5
1	57	22	35	8	21
0	1	2	3	4	5
11	26	9	87	31	13
0	1	2	3	4	5
7	32	29	4	71	44
0	1	2	3	4	5
16	97	23	78	18	29
0	1	2	3	4	5
0	1	2	3	4	5

1. Qual a ordem dessa matriz?
2. Qual a sua representação em termos de linhas e colunas?
3. Como representar em Python?

## ≡ Exemplo

```
A = []  
for i in range(5):  
    A.append( [0] * 5 )  
A[1][1] = 2  
for i in range(5):  
    print(A[i])
```

