



AP1 - Algoritmos e Programação 1

Listas

Conteúdo 10a



Antes.... alguns lembretes....

- Sobre EAD3 e EAD4:
 - Enviem email para carlao2005@gmail.com com a formação dos grupos (trios ou duplas)
 - colocar no assunto: "grupo trabalho disciplina AP1"
 - Prazo: **20 de maio**
 - após esse prazo, vou assumir que o trabalho será individual
 - os temas serão distribuídos para os grupos/alunos através de sorteio
- Sobre EAD1:
 - Sobre Strings
 - Prazo de entrega da atividade: 23:55 do dia **9/maio**
 - Próximo Domingo!!

Introdução

- As variáveis de um programa podem ser **simples** ou **compostas**
- Uma **variável simples** é capaz de armazenar **apenas um valor** em um dado momento
- Exemplos de variáveis simples:
 - idade
 - peso
 - altura
 - data_nascimento
 - nome, etc...

Exemplo

```
declare NOTA1, NOTA2, media Reais
```

```
inicio
```

```
  escreva("Digite a nota1: ")
```

```
  leia(NOTA1)
```

```
  escreva("Digite a nota2: ")
```

```
  leia(NOTA2)
```

```
  media ← (NOTA1 + NOTA2) / 2
```

```
  escreva("A média é: ", media)
```

Digite a nota1:

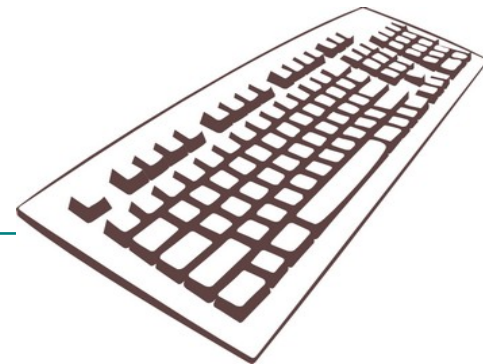
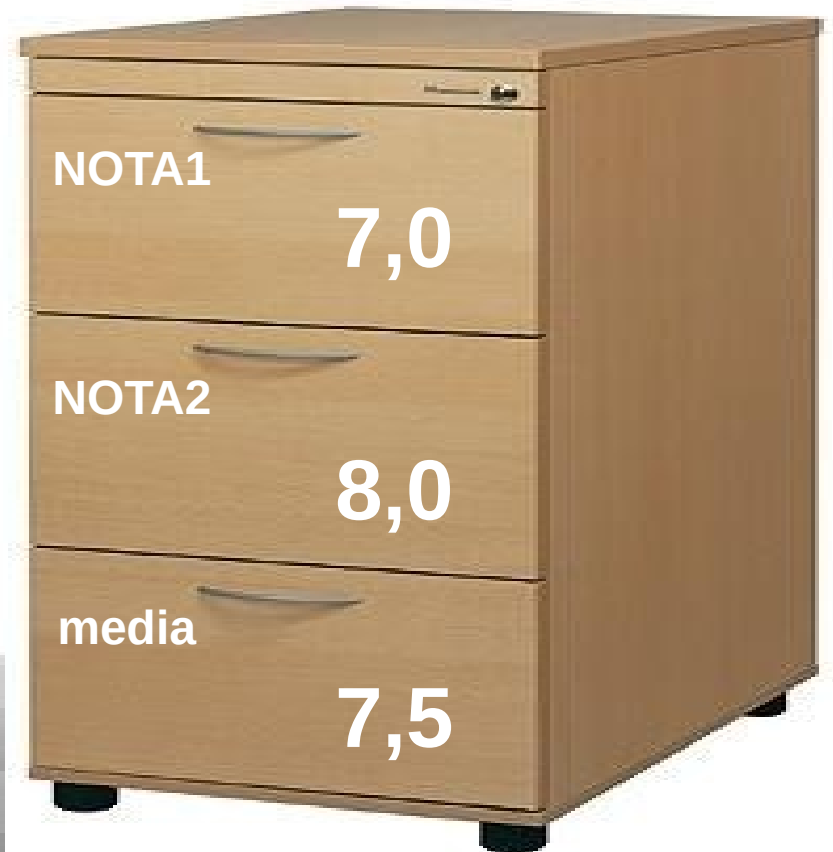
7.0

Digite a nota2:

8,0

A média é: 7,5

Memória



Exemplo

```
declare NOTA1, NOTA2, media Reais
```

```
inicio
```

```
  escreva("Digite a nota1: ")
```

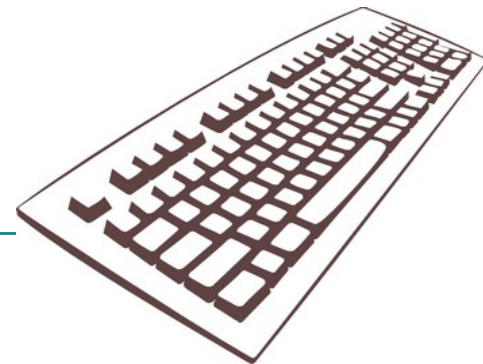
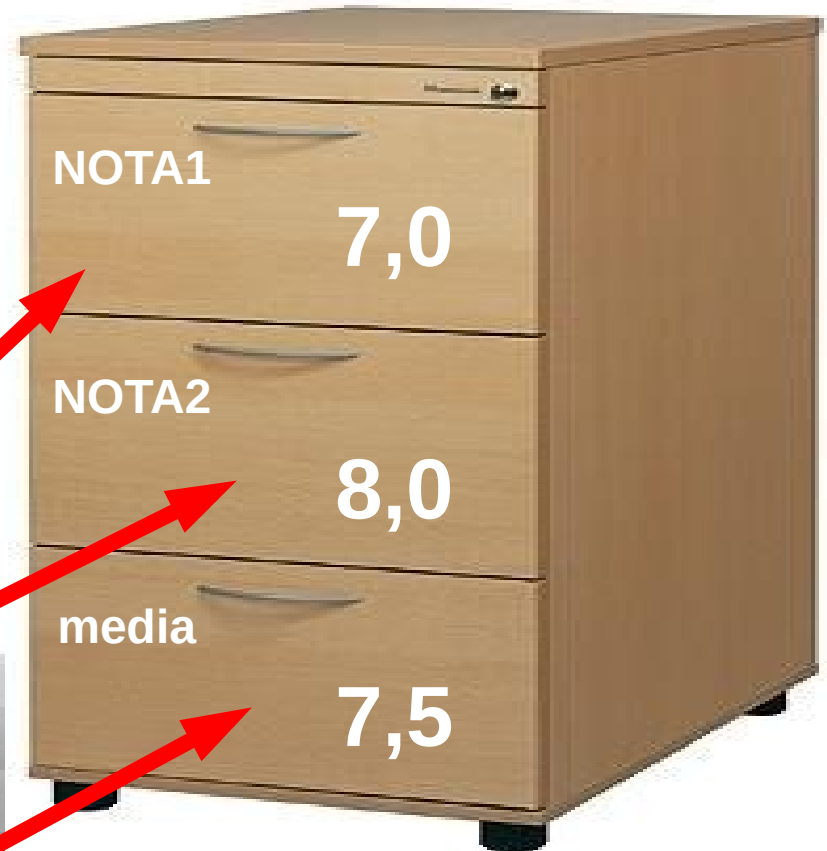
```
  leia(NOTA1)
```

```
  escreva("Digite a nota2: ")
```

```
  leia(NOTA2)
```

**Uma variável simples
é capaz de armazenar
apenas um valor em
um dado momento**

Memória



O que é uma lista?

- Uma **variável composta** que pode armazenar **vários valores** ao mesmo tempo
- Por exemplo, uma **variável composta** é capaz de armazenar o **cpf, nome, endereço e salário** de um funcionário de uma empresa
 - Tudo em uma **única** variável!
- Tipos:
 - Variável composta **homogênea**:
 - todos os valores são do mesmo tipo (arrays de outras ling.)
 - Variável composta **heterogênea**:
 - os valores podem ser de tipos diferentes (a lista do Python)

O que é uma lista?

- Os valores que compõem uma lista são chamados de **elementos**
- Por ser heterogênea, os elementos de uma lista podem ser de **tipos diferentes**
 - inclusive pode ser outra lista 🤪
- Cada elemento em uma Lista é identificado através de um **índice**

Exemplos de listas:

- Criando uma variável do tipo lista, vazia
 - Isto é, sem nenhum elemento dentro

```
L = []
```

```
>>>  
>>> L = []  
>>> print(L)  
[]  
>>>
```


Exemplos de listas:

- Criando uma variável do tipo lista, com vários elementos:
 - **strings**, **inteiro**, **float**

```
L = ['123456789-11', 'Maria', 'Rua xxx', 38, 2510.60]
```

```
>>>
>>> L = ['123456789-11', 'Maria', 'Rua xxx', 38, 2510.60]
>>> print(L)
['123456789-11', 'Maria', 'Rua xxx', 38, 2510.6]
>>>
>>> |
```

```
L = ['123456789-11', 'Maria', 'Rua xxx', 38, 2510.60]
```



Acessando os elementos

- Para acessar um elemento qualquer da lista, usamos o **índice** que indica a posição do elemento na lista
- Em Python, o primeiro elemento de uma lista está na posição **0 (zero)**
 - Lembre-se!! Programadores contam a partir do zero!!!
- Para acessar um elemento qualquer de uma lista, usamos o nome da lista e o valor do índice (em que se encontra o elemento) entre colchetes
 - Do mesmo jeito que acessamos caracteres individuais numa String!
 - Já estudou o EAD #1 ????

```
>>> L = ['123456789-11', 'Maria', 'Rua xxx', 38, 2510.60]
>>>
>>> print( L[0] )
123456789-11
>>> print( type( L[0] ) )
<class 'str'>
>>>
>>> print( L[3] )
38
>>> print( type( L[3] ) )
<class 'int'>
>>>
>>> print( L[4] )
2510.6
>>> print( type( L[4] ) )
<class 'float'>
>>>
>>> print( L[5] )
Traceback (most recent call last):
  File "<pyshell#43>", line 1, in <module>
    print( L[5] )
IndexError: list index out of range
>>>
```

Modificando os elementos de uma lista

- As listas no Python são **mutáveis**, podendo ser alteradas a qualquer momento
- Por exemplo, se quisermos alterar o elemento da posição de índice 1 da lista L (slide anterior), podemos fazer a seguinte atribuição:

```
L[1] = 'Asdrubal'
```

```
>>>
>>> L[1] = 'Asdrubal'
>>>
>>> print( L )
['123456789-11', 'Asdrubal', 'Rua xxx', 38, 2510.6]
>>>
>>> |
```

Percorrendo uma lista

- Para percorrer (fala-se “varrer”) uma lista pode-se utilizar qualquer uma das estruturas de repetição (**for** ou **while**)
 - Qual é a melhor?
 - Depende do que você quer fazer! Sempre!!
- Percorrendo uma lista com **for** acessando diretamente cada elemento:

```
lista_notas=[7.5, 9.7, 10.0, 5.5, 8.3]  
for nota in lista_notas:  
    print(nota)
```

File Edit Format Run Options Window Help

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
```

```
for nota in lista_notas:  
    print(nota)
```

===== RESTART:

7.5

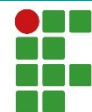
9.7

10.0

5.5

8.3

>>> |

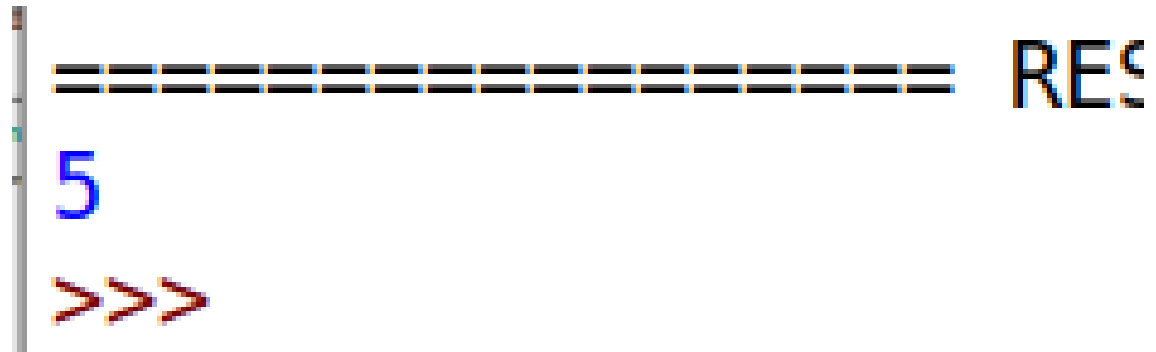


Verificar o comprimento de uma lista

- A função **len()** retorna o comprimento de uma lista.

```
lista_notas=[7.5, 9.7, 10.0, 5.5, 8.3]
```

```
print( len( lista_notas ) )
```



A terminal window showing the execution of the Python code. The prompt is a vertical bar. The first line shows the list `lista_notas` with its elements `7.5, 9.7, 10.0, 5.5, 8.3` and the result `5` in blue. The second line shows the prompt `>>>` followed by the code `print(len(lista_notas))` and the result `5` in blue.

Percorrendo uma lista

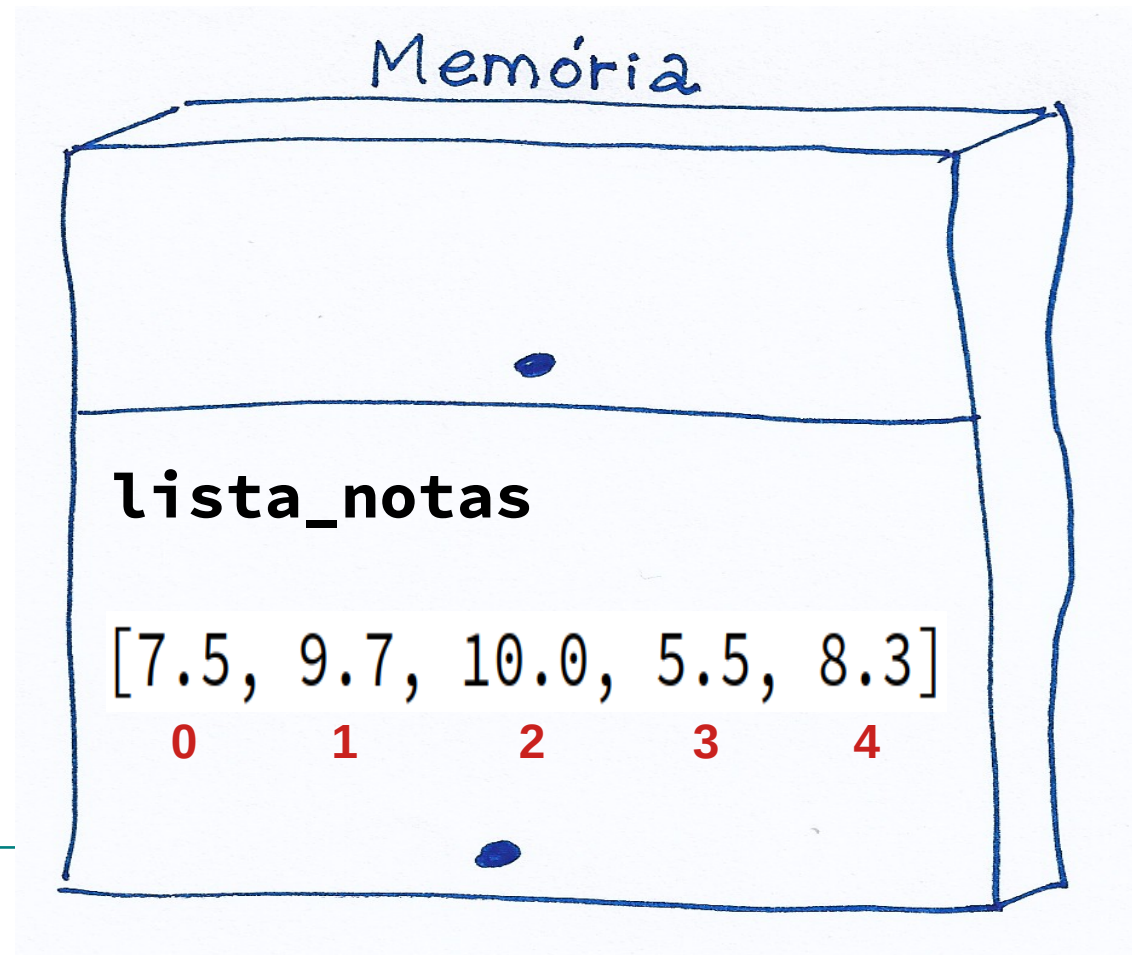
- Percorrendo uma lista com **while** e **índice**:

```
índice=0
while índice < len(lista_notas):
    print(lista_notas[índice])
    índice=índice+1
```

- Percorrendo uma lista com **for** e **índice**:

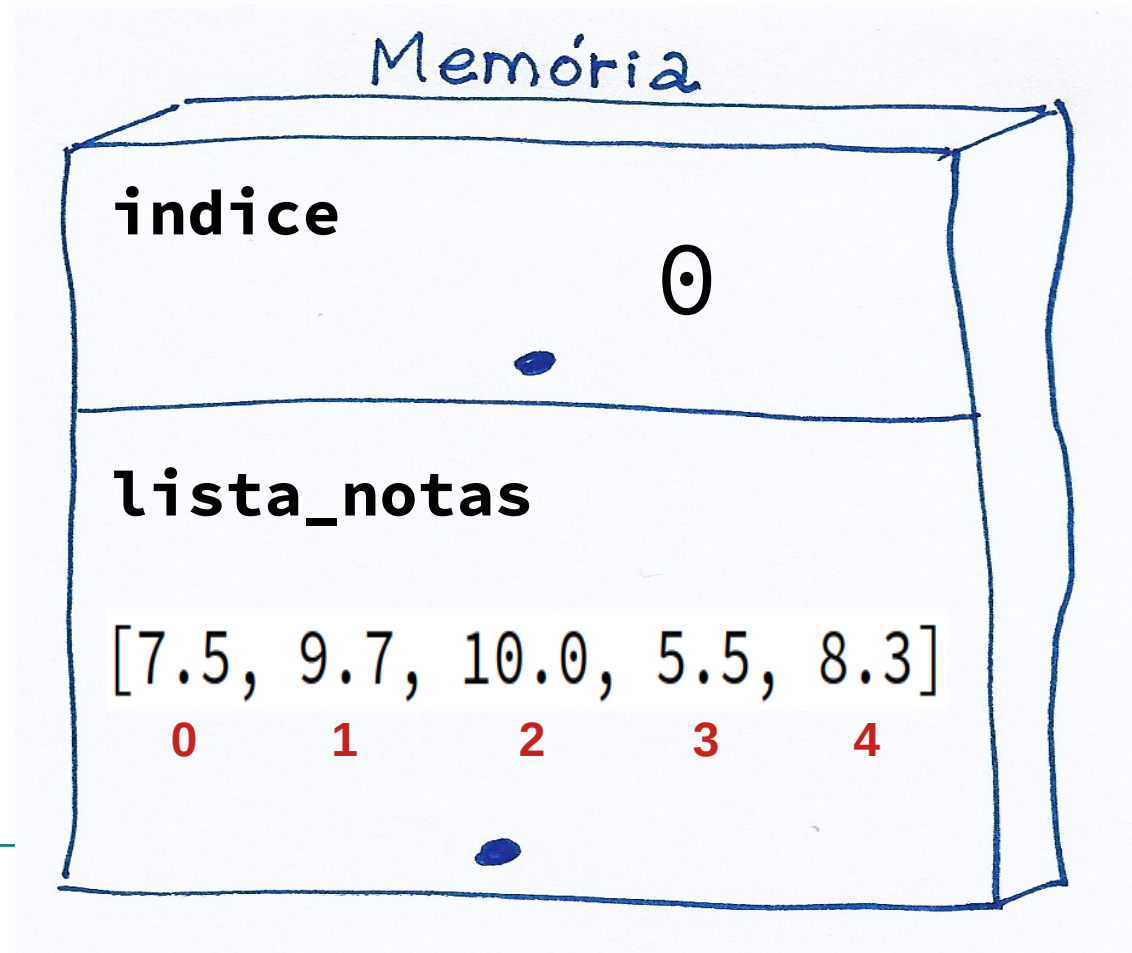
```
for índice in range(0, len(lista_notas)):
    print(lista_notas[índice])
```

```
indice=0  
while indice < len(lista_notas):  
    print(lista_notas[indice])  
    indice=indice+1
```



indice=0 ←

```
while indice < len(lista_notas):  
    print(lista_notas[indice])  
    indice=indice+1
```



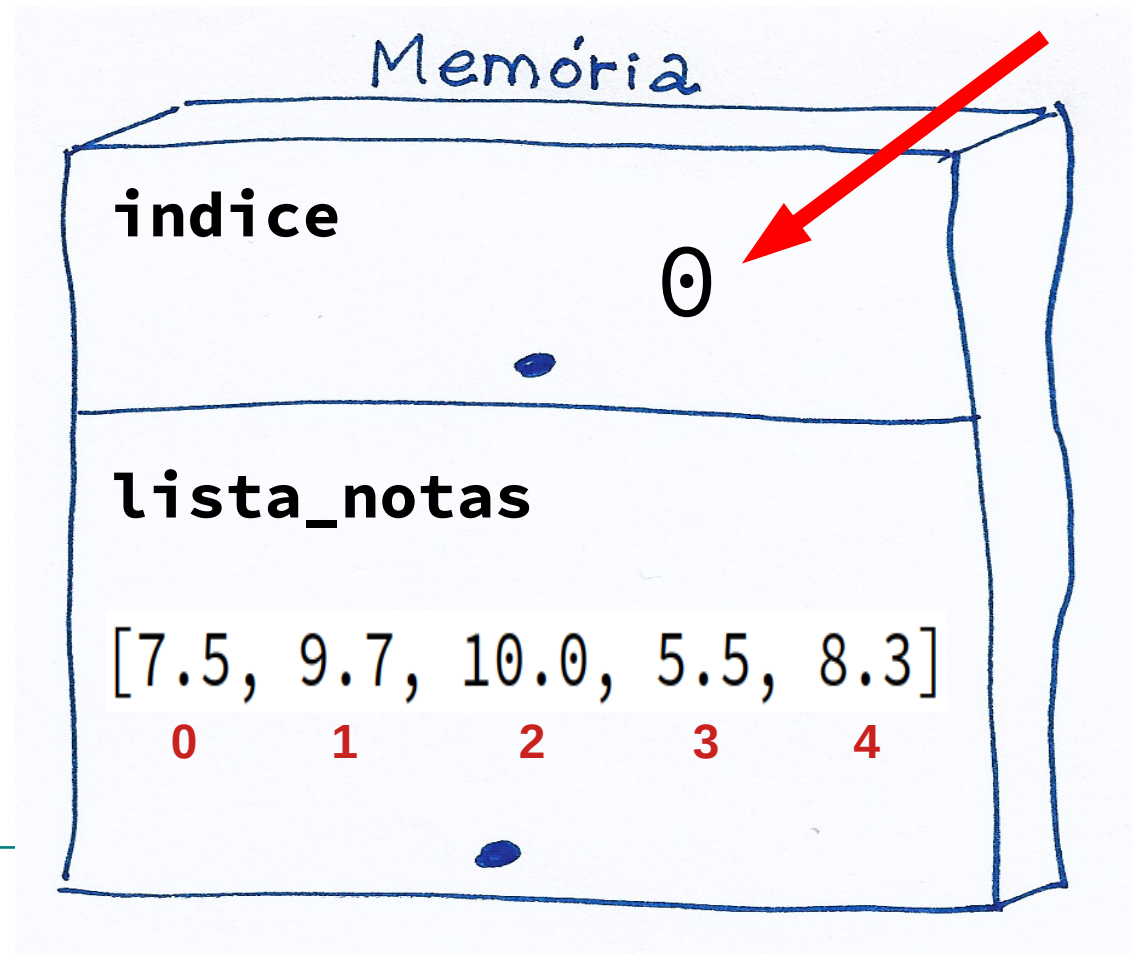
```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5



```
indice=0
```

```
while indice < len(lista_notas):
```

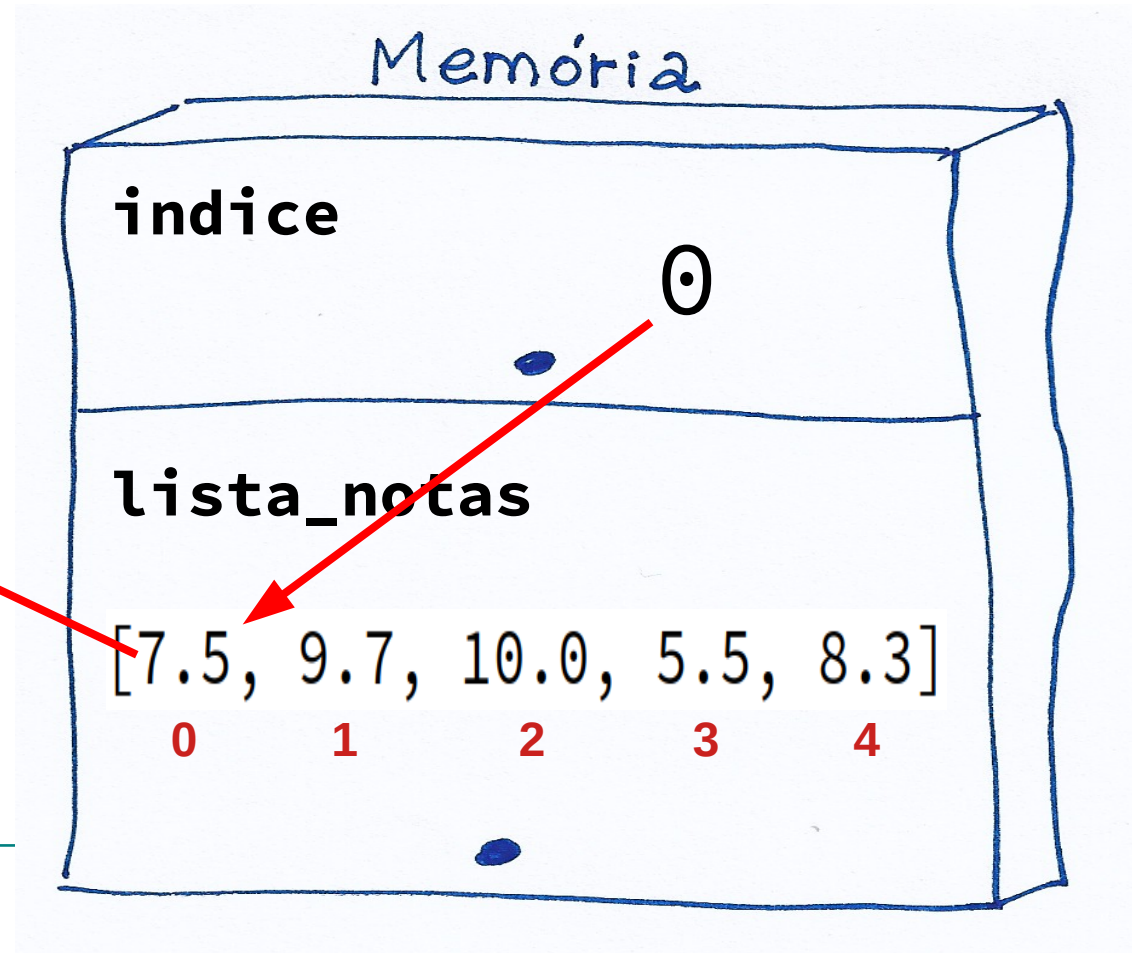
```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5

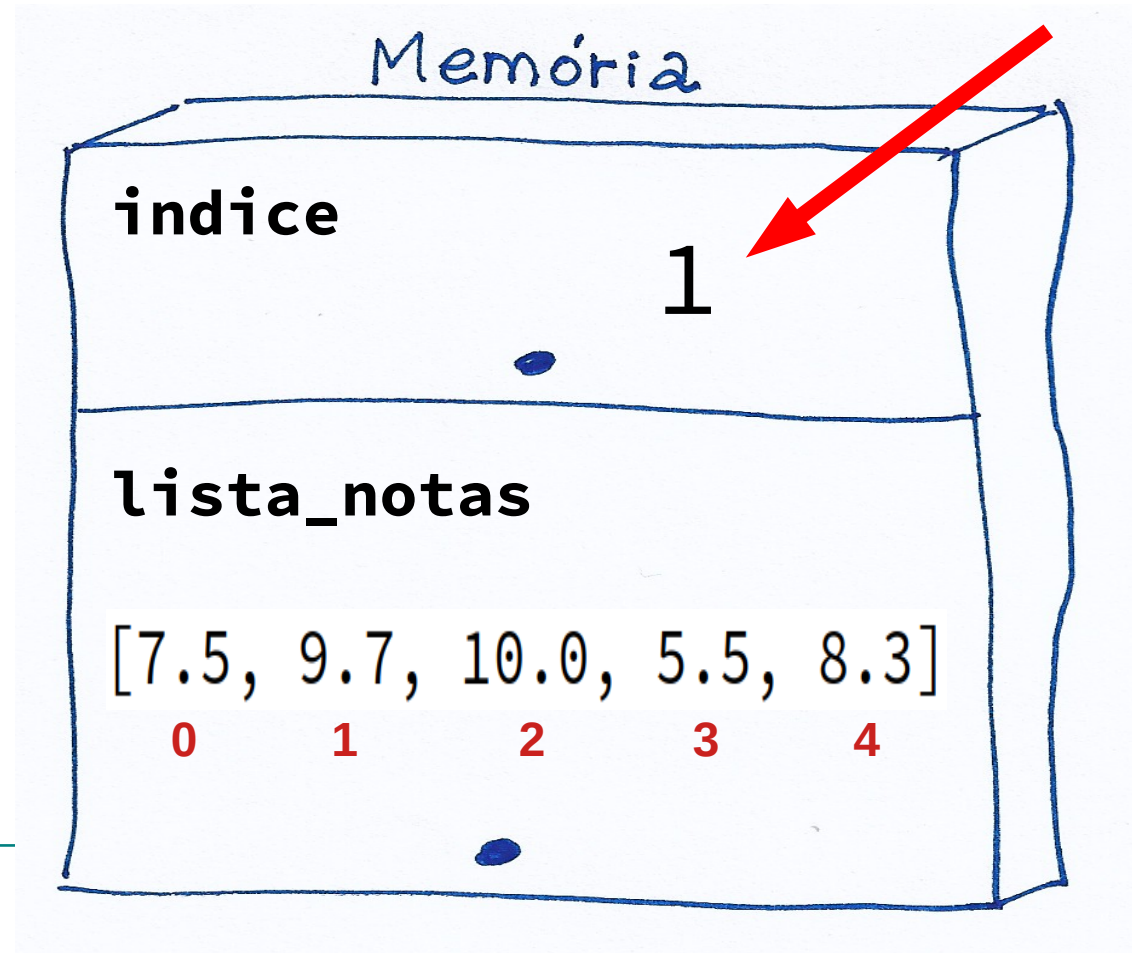



```
indice=0
while indice < len(lista_notas):
    print(lista_notas[indice])
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5



```
indice=0
```

```
while indice < len(lista_notas):
```

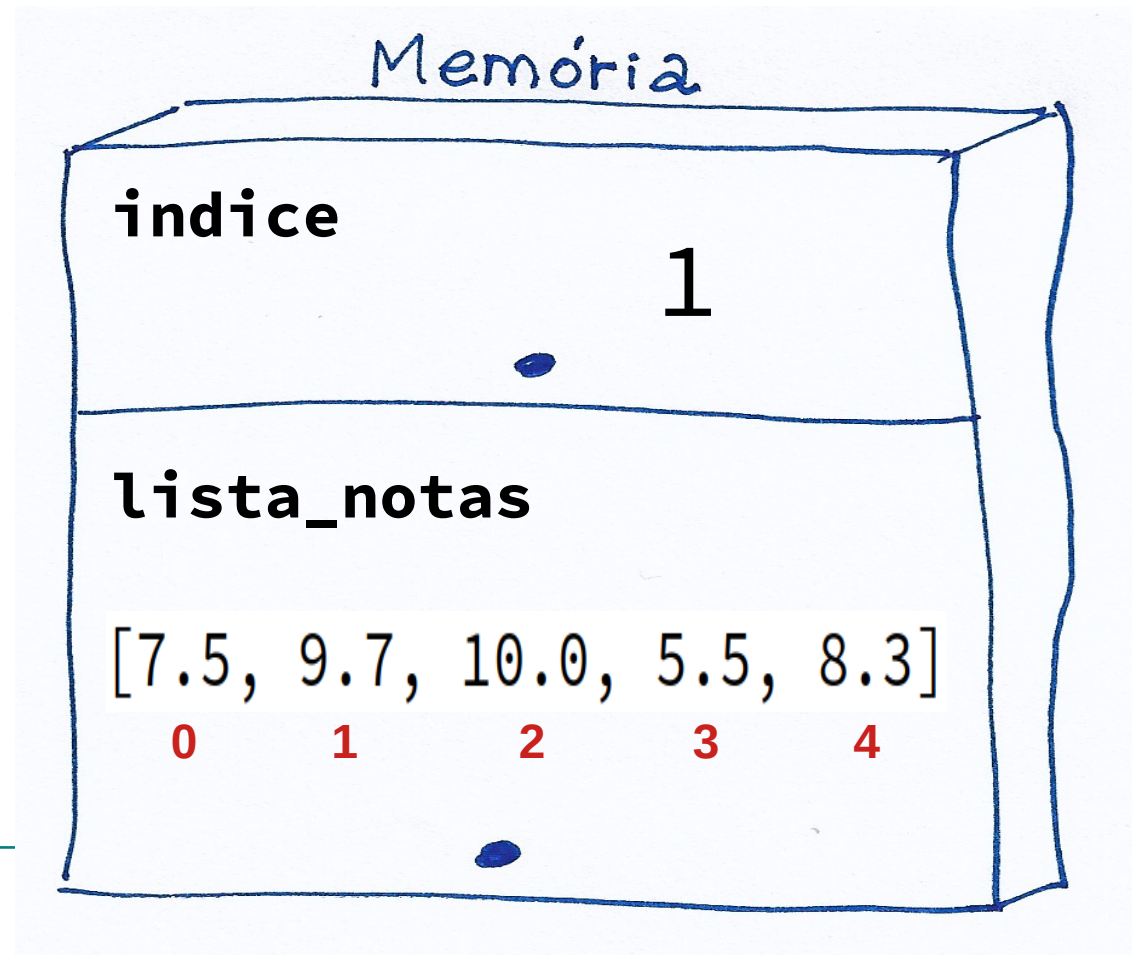
```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

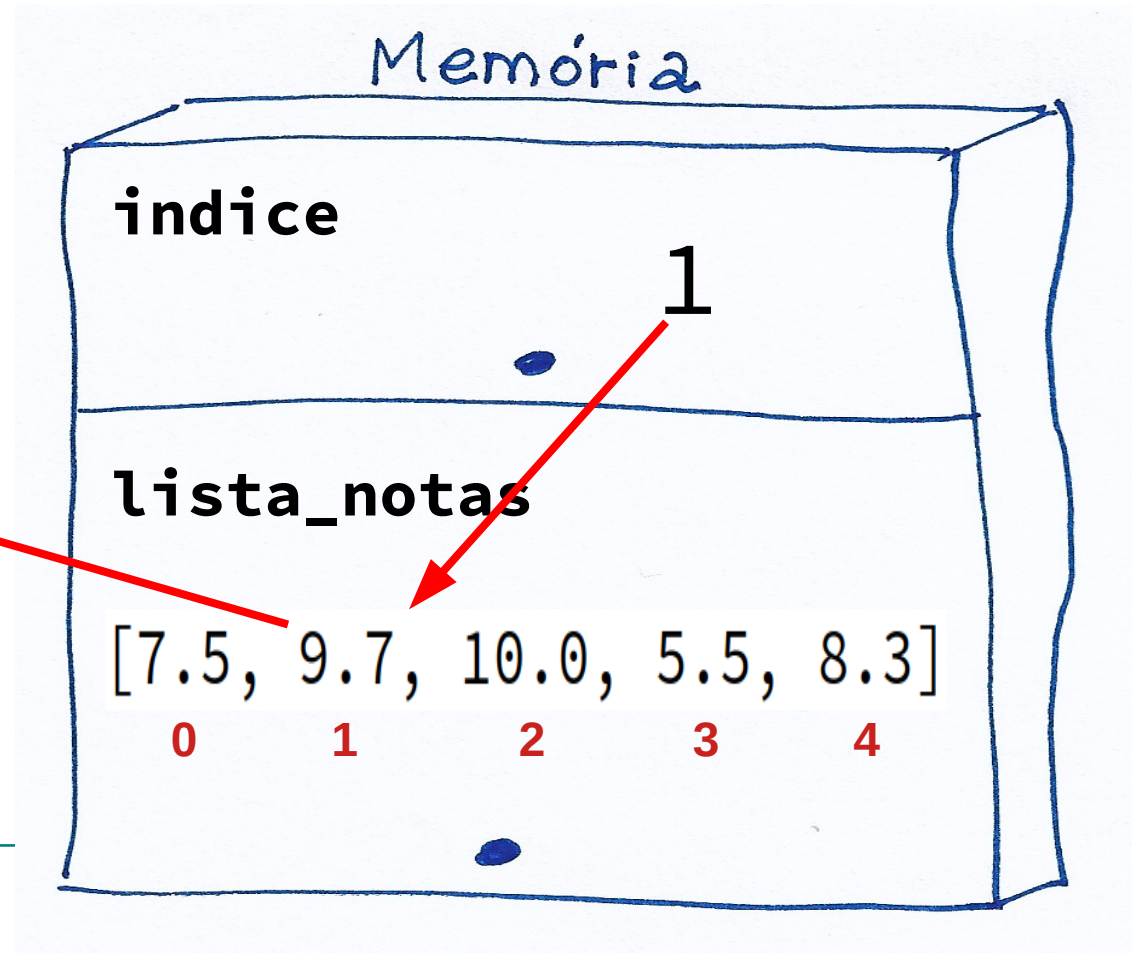
```
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5

9.7



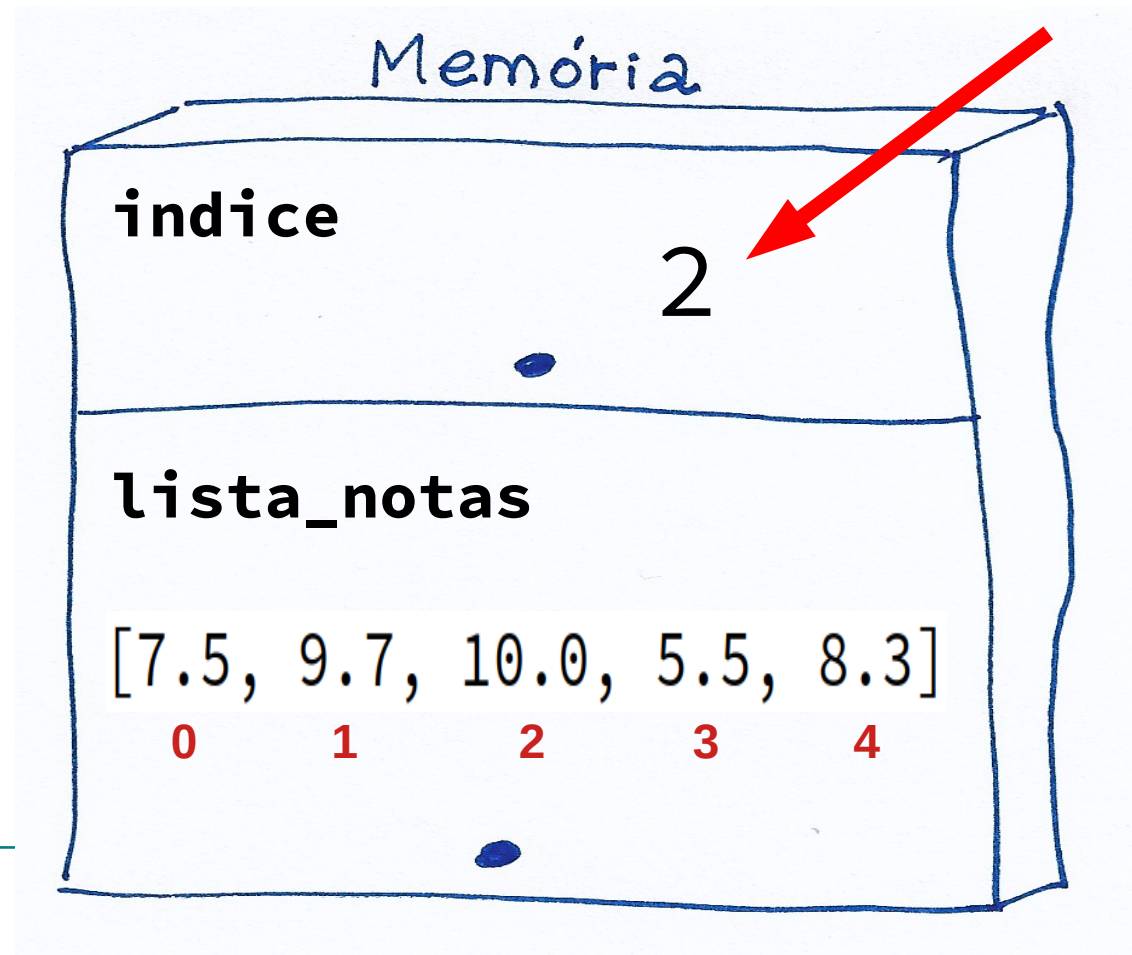

```
indice=0
while indice < len(lista_notas):
    print(lista_notas[indice])
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5

9.7



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

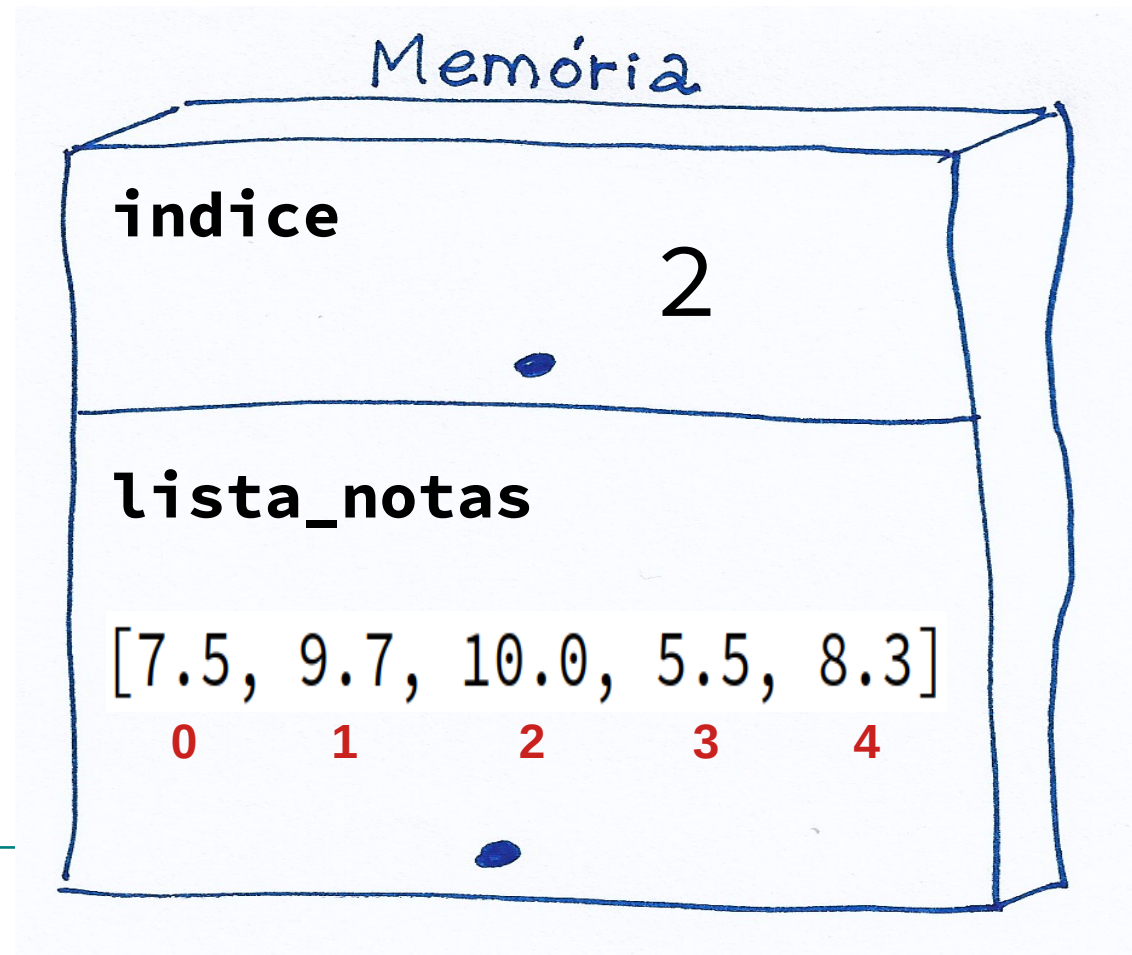
```
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5

9.7



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

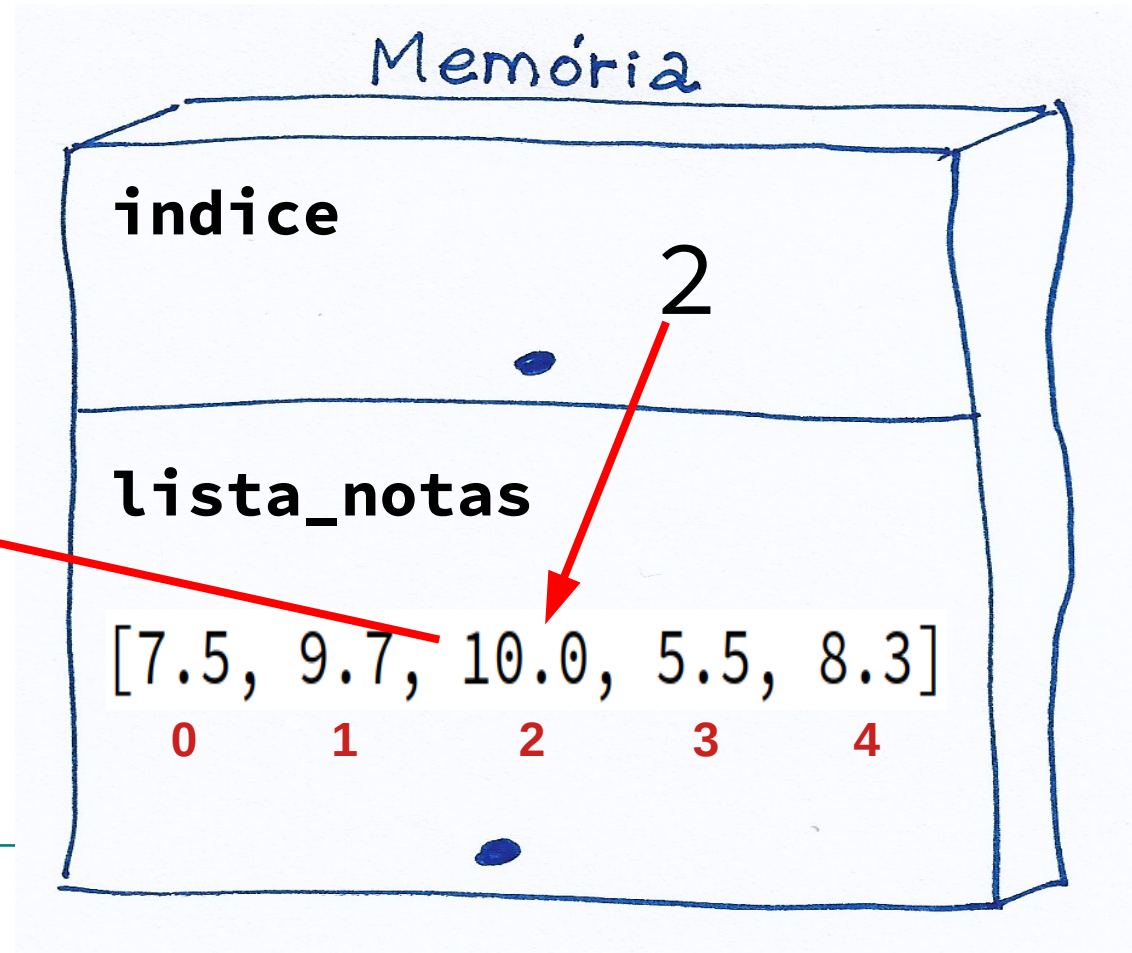
len(lista_notas) = 5

Tela:

7.5

9.7

10.0

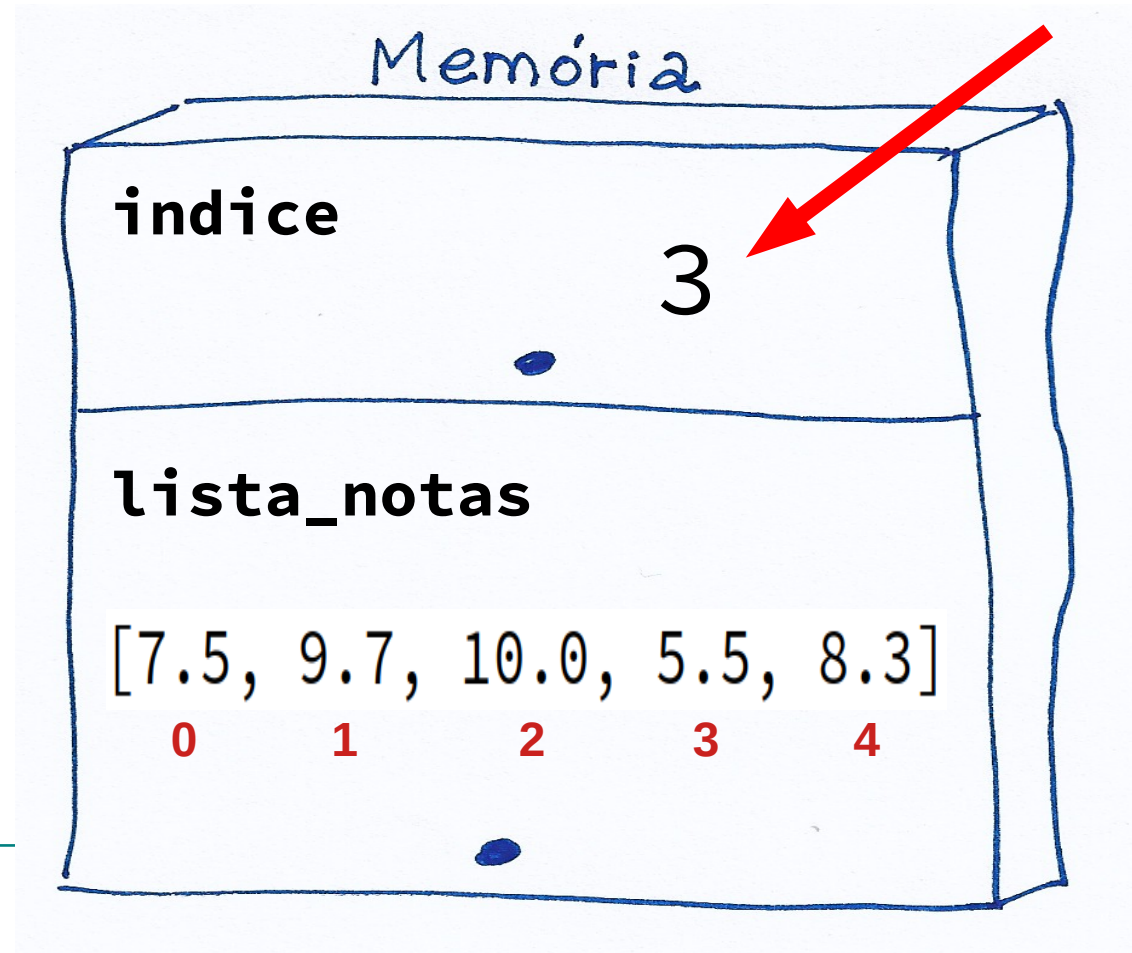


```
indice=0
while indice < len(lista_notas):
    print(lista_notas[indice])
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5
9.7
10.0




```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

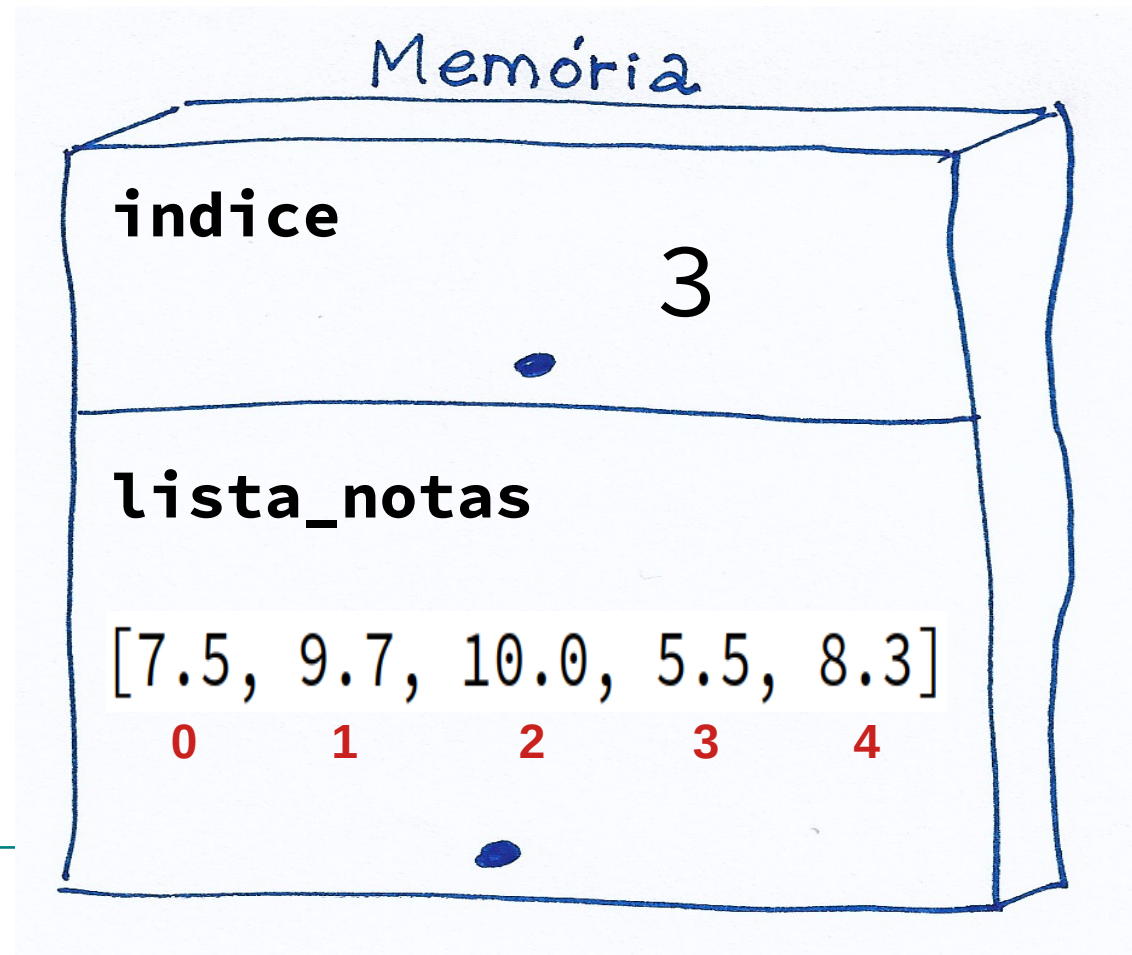
len(lista_notas) = 5

Tela:

7.5

9.7

10.0



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5

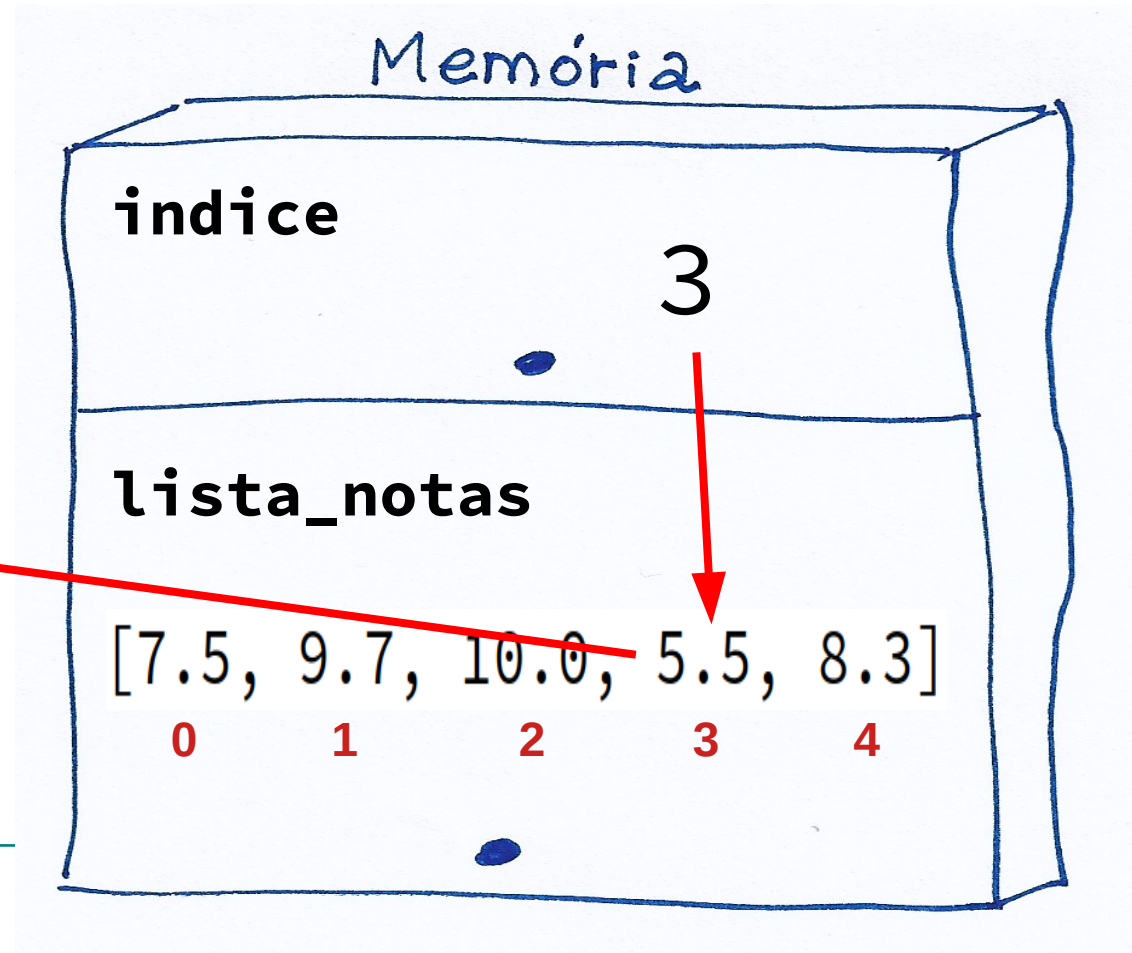
Tela:

7.5

9.7

10.0

5.5

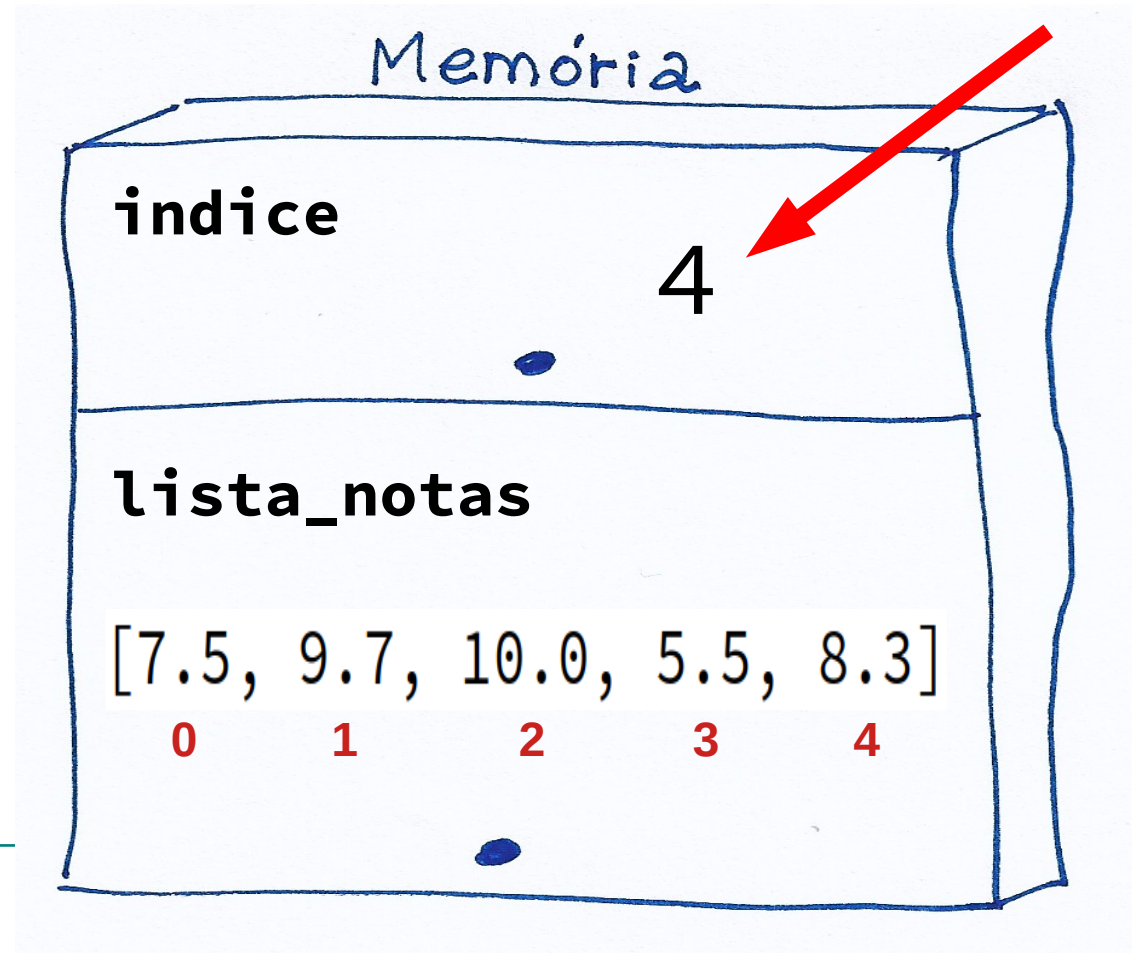


```
indice=0
while indice < len(lista_notas):
    print(lista_notas[indice])
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5
9.7
10.0
5.5



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5

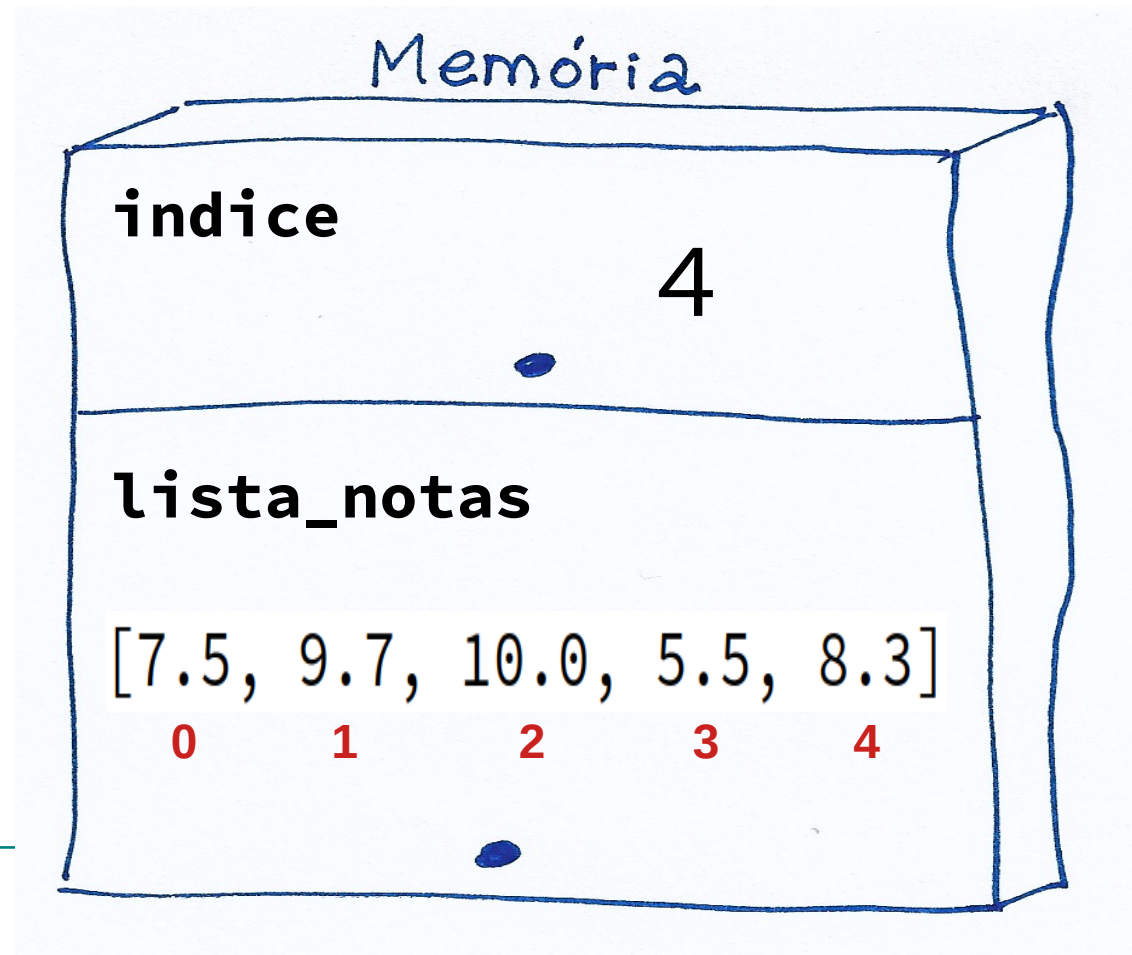
Tela:

7.5

9.7

10.0

5.5




```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5

Tela:

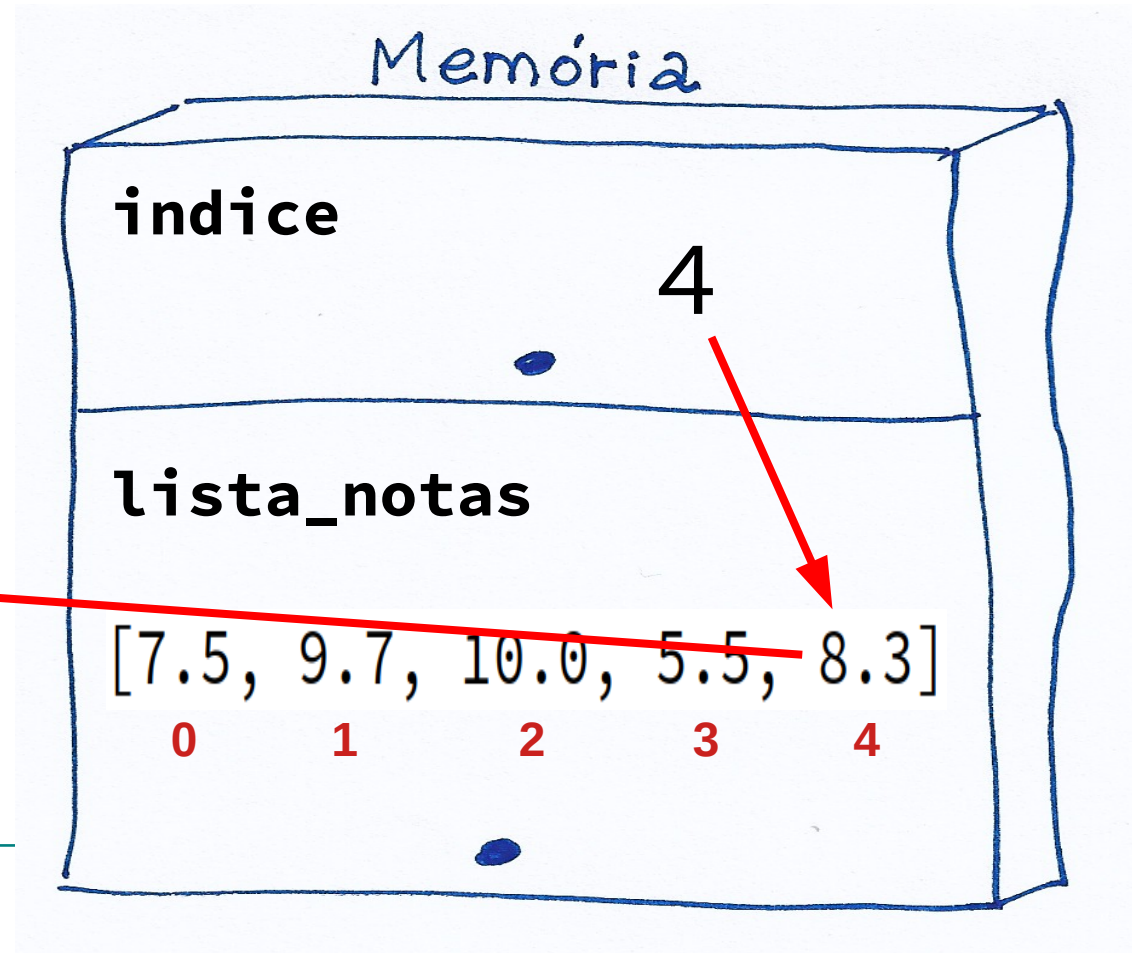
7.5

9.7

10.0

5.5

8.3

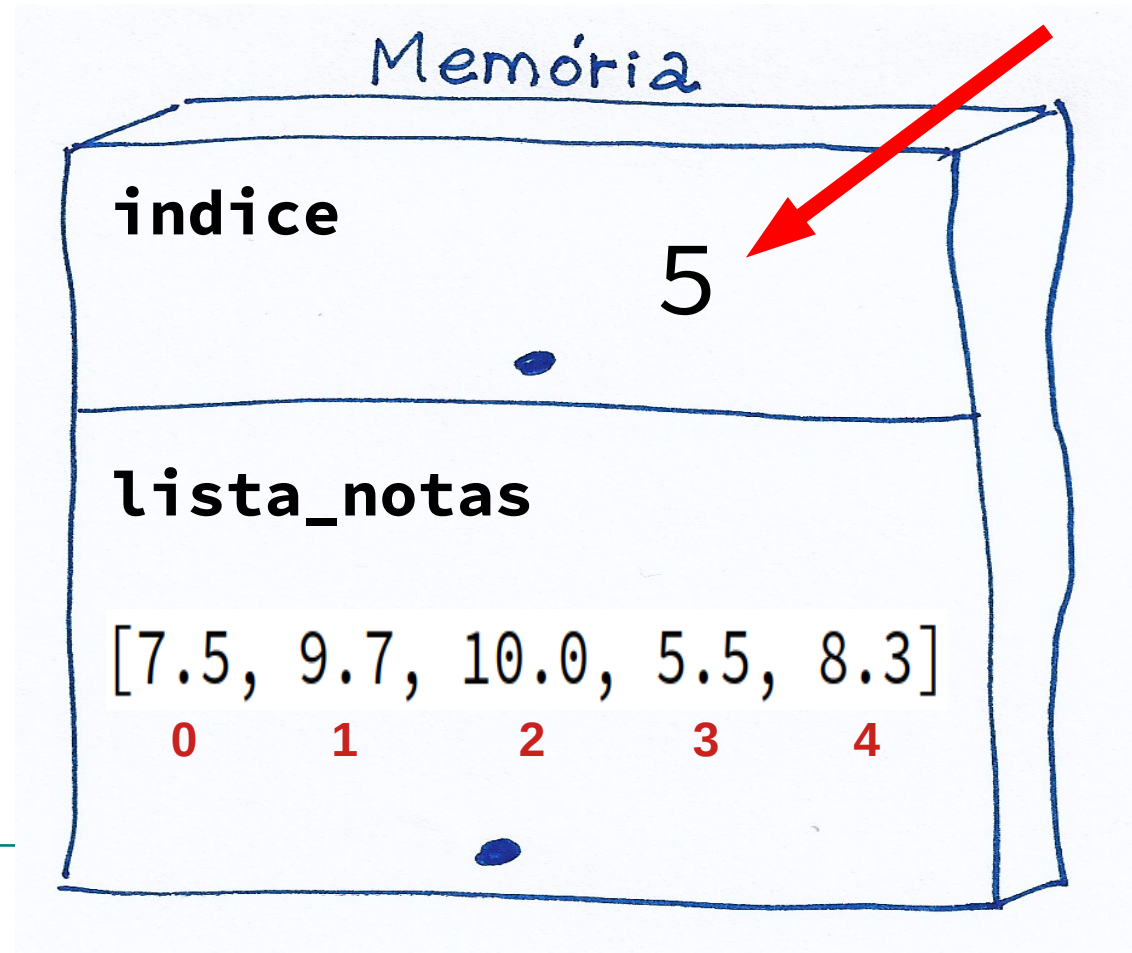


```
indice=0  
while indice < len(lista_notas):  
    print(lista_notas[indice])  
    indice=indice+1
```

len(lista_notas) = 5

Tela:

7.5
9.7
10.0
5.5
8.3



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

len(lista_notas) = 5

Tela:

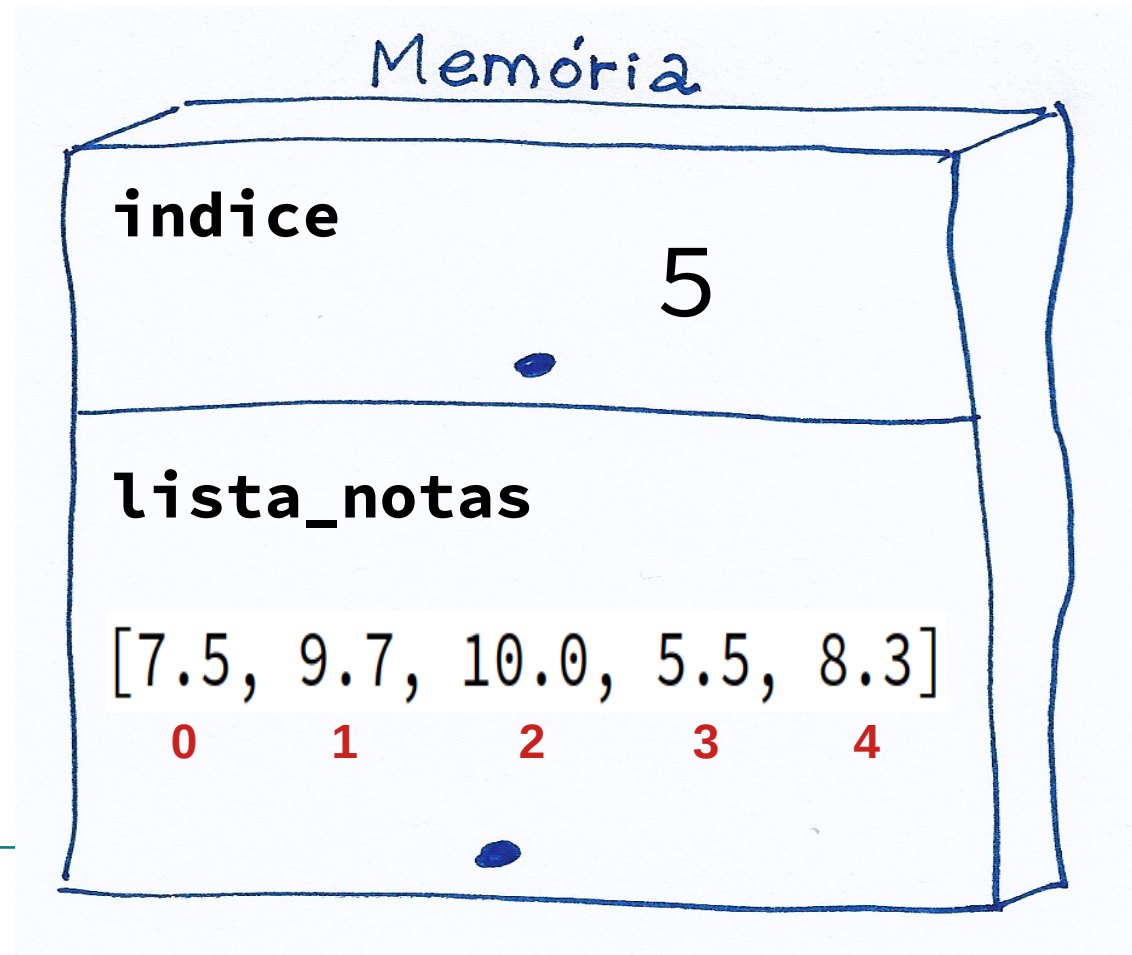
7.5

9.7

10.0

5.5

8.3



```
indice=0
```

```
while indice < len(lista_notas):
```

```
    print(lista_notas[indice])
```

```
    indice=indice+1
```

**fim do while,
fim do programa.**

len(lista_notas) = 5

Tela:

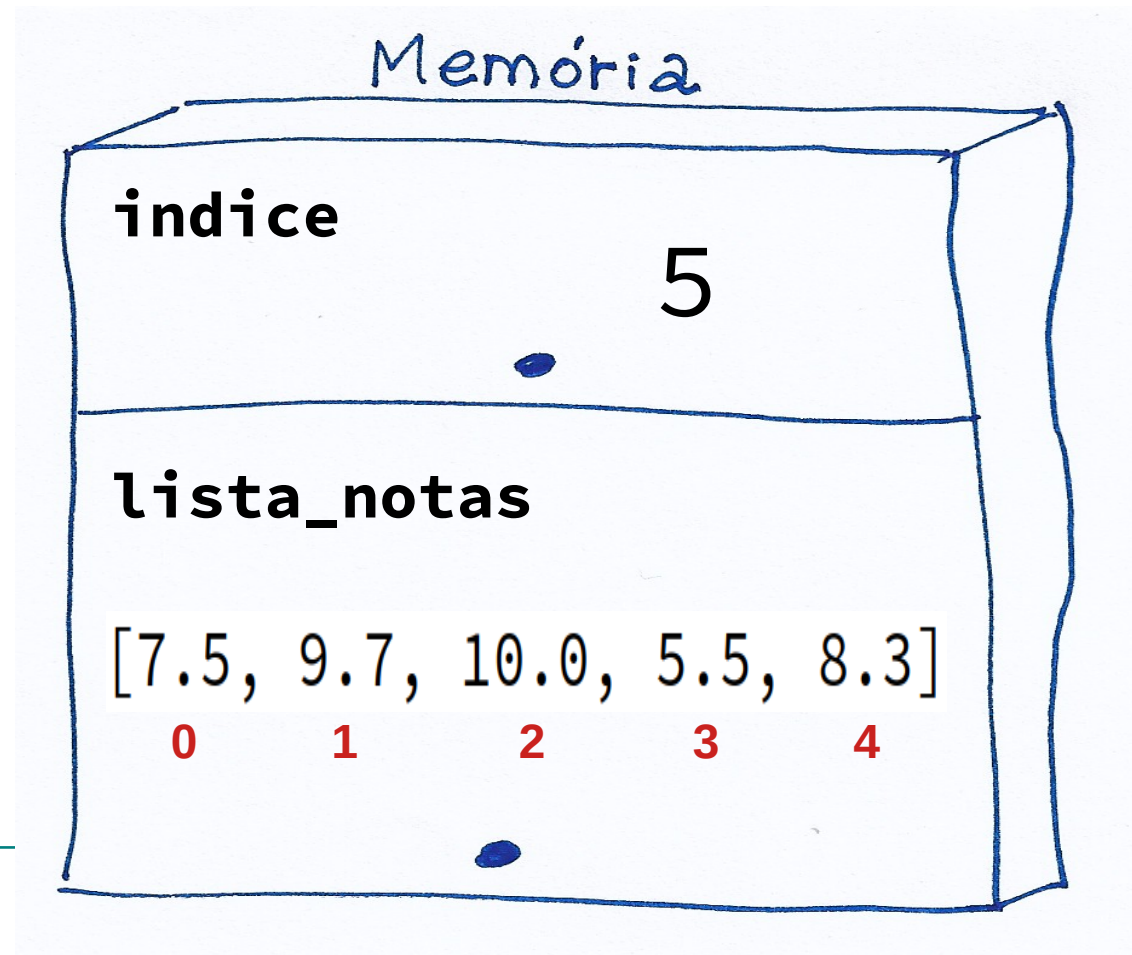
7.5

9.7

10.0

5.5

8.3




```
indice=0
while indice < len(lista_notas):
    print(lista_notas[indice])
    indice=indice+1
```

← fim do while,
fim do programa.

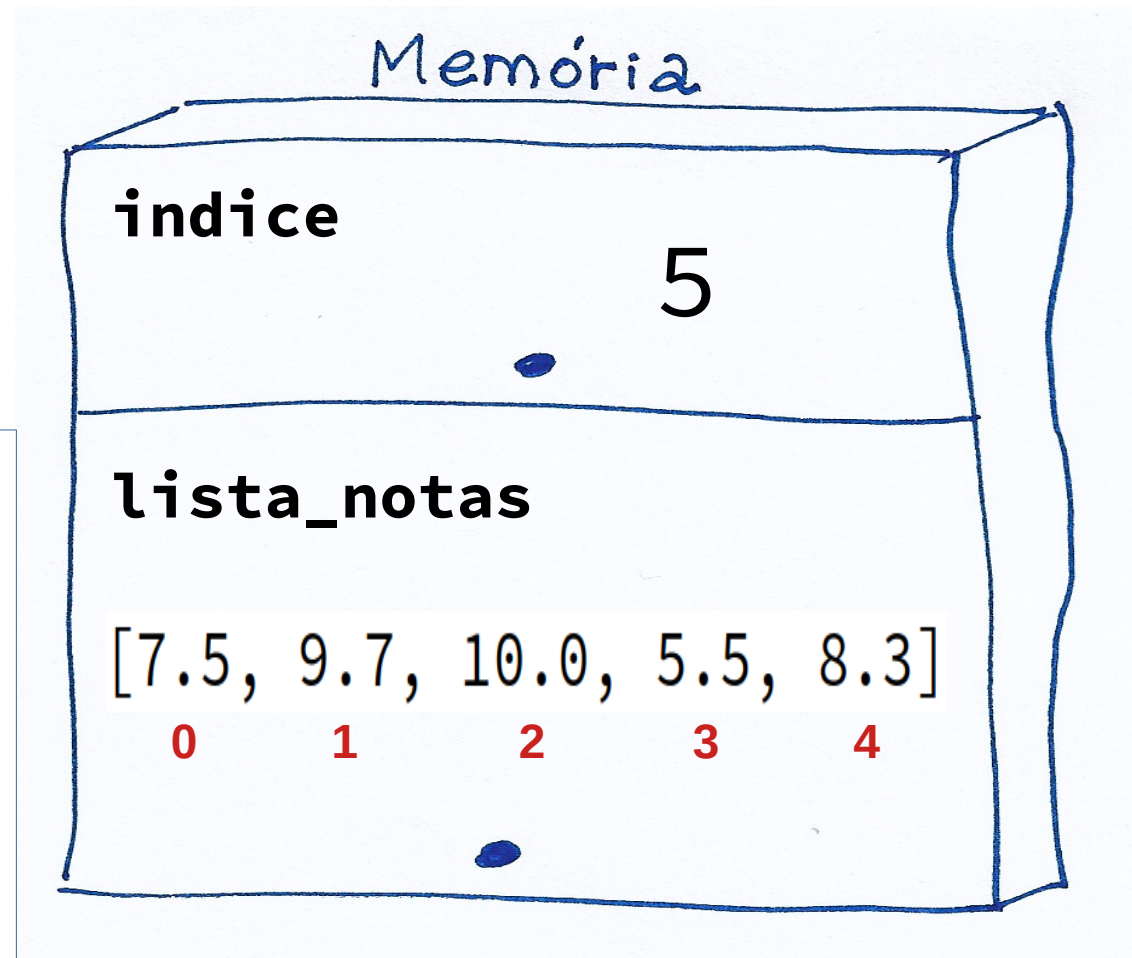
`len(lista_notas) = 5`

Tela:

7.5
9.7
10.0
5.5
8.3

===== RESTART:

7.5
9.7
10.0
5.5
8.3
>>> |



Com for:

```
lista_notas=[7.5, 9.7, 10.0, 5.5, 8.3]

for indice in range(0, len(lista_notas)):
    print(lista_notas[indice])
```

```
===== RESTART:
7.5
9.7
10.0
5.5
8.3
>>> |
```

Principais operações com listas

- Verificar se um elemento pertence a uma lista
- Inserir um novo elemento em uma lista
- Remover um elemento de uma lista
- Concatenar listas

Elementos pertencentes a uma lista

- O operador lógico **in/not in** é usado para verificar se um elemento pertence a uma lista
 - O resultado é sempre verdadeiro (True) ou falso (False)

```
lista_notas=[7.5, 9.7, 10.0, 5.5, 8.3]
```

```
print(10 in lista_notas)
```

```
print(5.5 not in lista_notas)
```

```
===== RESTART:
True
False
>>> |
```


Inserir um novo elemento no final de uma lista

- A função **append()** adiciona um novo elemento no **final** de uma lista

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
```

```
print( lista_notas )
```

```
lista_notas.append(6.2)
```

```
valor = float( input( "Digite um valor:" ) )
```

```
lista_notas.append( valor )
```

```
print(lista_notas)
```

```
===== RESTART: /home/carlaolao
[7.5, 9.7, 10.0, 5.5, 8.3]
Digite um valor:7.5
[7.5, 9.7, 10.0, 5.5, 8.3, 6.2, 7.5]
>>> |
```

Inserir um novo elemento em uma lista

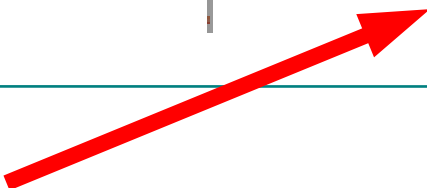
- A função **insert()** adiciona um novo elemento em uma posição qualquer de uma lista.
- Por exemplo, para inserir a nota 9.5 na posição de índice 1 da lista a seguir, faça:

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
```

```
print( lista_notas )
```

```
lista_notas.insert( 1, 9.5)
```

```
print(lista_notas)
```



0	1	2	3	4	
7.5	9.7	10.0	5.5	8.3	
7.5	9.5	9.7	10.0	5.5	8.3
0	1	2	3	4	5

Remover um elemento de uma lista

- O operador **del** remove um elemento de uma lista.

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
```

```
print(lista_notas)
```

```
del lista_notas[3]
```

```
print(lista_notas)
```

0	1	2	3	4
[7.5,	9.7,	10.0,	5.5,	8.3]
[7.5,	9.7,	10.0,	8.3]	
0	1	2	3	

Concatenação de listas

- O operador `+` concatena (soma) duas listas

```
L1 = [1,2,3]
```

```
L2 = [4,5,6]
```

```
L3 = L1 + L2
```

```
print(L3)
```

```
===== RESTART:
```

```
[1, 2, 3, 4, 5, 6]
```

```
>>> |
```

Encontrando a posição de um elemento qualquer de uma lista

- A função **index()** é usada para encontrar a posição de um elemento qualquer em uma lista.

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
```

```
print( lista_notas.index(10) )
```

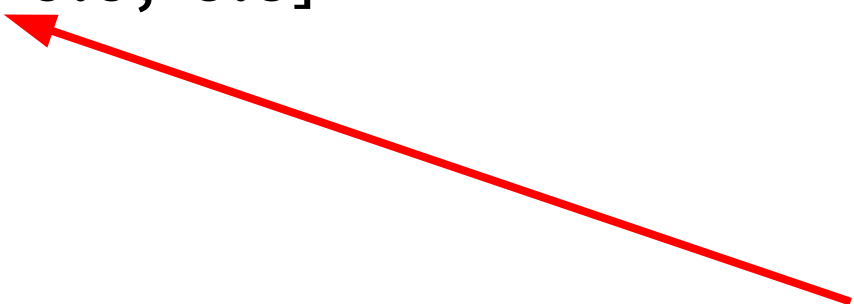
===== RESTART:

2

Encontrando a posição de um elemento qualquer de uma lista

- A função **index()** é usada para encontrar a posição de um elemento qualquer em uma lista.

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
```



```
print( lista_notas.index(10) )
```

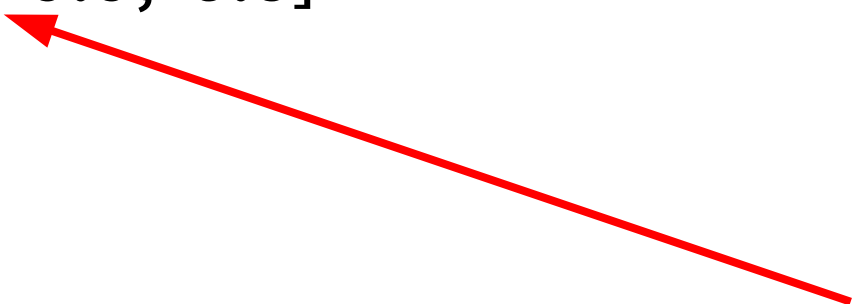
===== RESTART:

2

Encontrando a posição de um elemento qualquer de uma lista

- A função **index()** é usada para encontrar a posição de um elemento qualquer em uma lista.

```
lista_notas = [7.5, 9.7, 10.0, 5.5, 8.3]
print( lista_notas.index(10) )
```



Se não achar, dá erro!

===== RESTART:

2

Criando listas de inteiros

- Listas que contêm números inteiros consecutivos são bastante comuns
- O operador **range** fornece uma maneira simples de criar uma lista de inteiros. Exemplo:

```
lista_inteiros = list( range(3,13) )
```

```
print(lista_inteiros)
```

```
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

- O operador **range** pega dois parâmetros (dois inteiros) e devolve uma lista que contém todos os inteiros do primeiro até o segundo, incluindo o primeiro mas não incluindo o segundo!

Criando listas de inteiros

- Com um parâmetro apenas, **range** cria uma lista que inicia em 0. Exemplo:

```
lista_inteiros = list( range(15) )
```

```
print(lista_inteiros)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

- Se houver um terceiro argumento, ele especifica o tamanho do “passo” entre os elementos sucessivos. Exemplo:

```
lista_inteiros = list( range(1, 21, 2) )
```

```
print(lista_inteiros)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Comparações entre Strings e listas

■ Inicializações:

- `string vazia=""`
- `lista vazia=[]`

■ Atribuições:

- `minha_str="Bom dia com alegria"`
- `minha_lista_str=["Bom", "dia", "com", "alegria"]`
- `minha_lista_int=[3, 6, 1, 2, 43, 89]`
- `minha_listaflt=[4.5, 3.9, 2.1, 5.4, 10.7, 8.0]`

Comparações entre Strings e listas

- Concatenar um caracter numa string:

- `caracter=input("Digite uma letra:")`
- `minha_str = minha_str + caracter`

- Adicionar um elemento em uma lista:

- `num = int(input("Digite um número:"))`
- `minha_lista_int.append(num)`

Comparações entre Strings e listas

■ Tamanhos:

- `tam_str = len(minha_str)`
- `tam_lint = len(minha_lista_int)`
- `tam_lflt = len(minha_listaflt)`

■ Acesso por índice (de 0 a tamanho-1):

- Acesso a um caracter da string: `minha_str[4]`
- Acesso a um elemento da lista: `minha_lista[3]`

Comparações entre Strings e listas - Fatiamentos

- Pode-se também “fatiar” uma lista:

```
lista = [ 01.2, 13.5, 24.3, 30.2, 42.2, 51.7, 69.8 ]
```

```
print( lista[2:6] )
```

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

```
print( lista[:4] )
```

[4.3, 0.2, 2.2, 1.7]

[0.2, 2.2, 1.7, 9.8]

[1.2, 3.5, 4.3, 0.2]

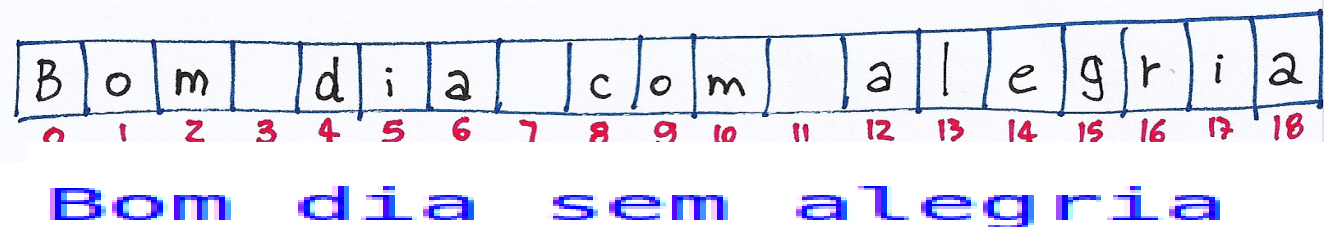
Comparações – Strings NÃO mutáveis X Listas mutáveis

- Para trocar o 'com' por 'sem' na string após encontrar a posição correspondente é:

```
minha_str = "Bom dia com alegria"
```

```
nova_str = minha_str[0:8] + "sem" + minha_str[11:19]
```

```
print(nova_str)
```



- Na lista `minha_lista_str=["Bom", "dia", "com", "alegria"]` para substituir o "com" por "sem" basta fazer

```
minha_lista_str = ["Bom", "dia", "com", "alegria"]
```

```
minha_lista_str[2] = "sem"
```

```
print(minha_lista_str)
```

```
['Bom', 'dia', 'sem', 'alegria']
```

Listas como elementos de Listas !

Recapitulando: O que é uma lista?

- Os valores que compõem uma lista são chamados de **elementos**
- Por ser heterogênea, os elementos de uma lista podem ser de **tipos diferentes**
 - inclusive pode ser outra lista

Recapitulando: O que é uma lista?

- Os valores que compõem uma lista são chamados de **elementos**
- Por ser heterogênea, os elementos de uma lista podem ser de **tipos diferentes**
 - inclusive pode ser **outra lista** 🤪



Exemplo passo a passo *(no terminal principal do idle):*

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)  
[GCC 9.3.0] on linux  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>
```

Exemplo passo a passo *(no terminal principal do idle):*

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
---
```



Exemplo passo a passo *(no terminal principal do idle):*

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
```



Exemplo passo a passo *(no terminal principal do idle):*

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
```

Exemplo passo a passo *(no terminal principal do idle)*:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
>>> # colocando um número real na lista:
>>> lista.append(7.56)
>>> print(lista)
[4, 7.56]
```



Exemplo passo a passo *(no terminal principal do idle)*:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
>>> # colocando um número real na lista:
>>> lista.append(7.56)
>>> print(lista)
[4, 7.56]
>>> # colocando uma string na lista:
>>> lista.append("IFSP")
>>> print(lista)
[4, 7.56, 'IFSP']
```

Exemplo passo a passo *(no terminal principal do idle)*:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
>>> # colocando um número real na lista:
>>> lista.append(7.56)
>>> print(lista)
[4, 7.56]
>>> # colocando uma string na lista:
>>> lista.append("IFSP")
>>> print(lista)
[4, 7.56, 'IFSP']
>>>
>>> # criando uma outra lista:
>>> outra_lista = [ 1, "TII"]
>>> print(outra_lista)
[1, 'TII']
```

Exemplo passo a passo *(no terminal principal do idle)*:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
>>> # colocando um número real na lista:
>>> lista.append(7.56)
>>> print(lista)
[4, 7.56]
>>> # colocando uma string na lista:
>>> lista.append("IFSP")
>>> print(lista)
[4, 7.56, 'IFSP']
>>>
>>> # criando uma outra lista:
>>> outra_lista = [ 1, "TII"]
>>> print(outra_lista)
[1, 'TII']
>>>
>>> # tchan tchan tchan!!!!
```

Exemplo passo a passo *(no terminal principal do idle)*:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
>>> # colocando um número real na lista:
>>> lista.append(7.56)
>>> print(lista)
[4, 7.56]
>>> # colocando uma string na lista:
>>> lista.append("IFSP")
>>> print(lista)
[4, 7.56, 'IFSP']
>>>
>>> # criando uma outra lista:
>>> outra_lista = [ 1, "TII"]
>>> print(outra_lista)
[1, 'TII']
>>>
>>> # tchan tchan tchan!!!!
>>> # colocando a outra_lista DENTRO da lista!!!!
>>> lista.append( outra_lista )
```

Exemplo passo a passo *(no terminal principal do idle)*:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> # criando uma lista vazia:
>>> lista = []
>>>
>>> # exibindo a lista:
>>> print(lista)
[]
>>> # colocando um número inteiro na lista:
>>> lista.append(4)
>>> print(lista)
[4]
>>> # colocando um número real na lista:
>>> lista.append(7.56)
>>> print(lista)
[4, 7.56]
>>> # colocando uma string na lista:
>>> lista.append("IFSP")
>>> print(lista)
[4, 7.56, 'IFSP']
>>>
>>> # criando uma outra lista:
>>> outra_lista = [ 1, "TII"]
>>> print(outra_lista)
[1, 'TII']
>>>
>>> # tchan tchan tchan!!!!
>>> # colocando a outra_lista DENTRO da lista!!!!
>>> lista.append( outra_lista )
>>> print(lista)
[4, 7.56, 'IFSP', [1, 'TII']]
>>>
>>>
```

[4, 7.56, 'IFSP', [1, 'TII']]

[4, 7.56, 'IFSP', [1, 'TII']]



[4, 7.56, 'IFSP', [1, 'TII']]

[4, 7.56, 'IFSP', [1, 'TII']]

0

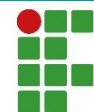
1

2

0

3

1



Acessando elementos

```
>>> lista1 = [1, 2, 3]
>>> print(lista1)
[1, 2, 3]
>>>
>>> lista2 = [4, 5, 6]
>>> print(lista2)
[4, 5, 6]
>>>
>>> lista3 = []
>>> lista3.append(lista1)
>>> lista3.append(lista2)
>>> print(lista3)
[[1, 2, 3], [4, 5, 6]]

>>> print( lista3[0] )
[1, 2, 3]
>>> print( lista3[0][0] )
1
>>> print( lista3[0][1] )
2
>>> print( lista3[0][2] )
3
>>> print( lista3[1][0] )
4
>>> print( lista3[1][1] )
5
>>> print( lista3[1][2] )
6
>>>
```

1. Faça um programa que receba o nome, o email, e duas notas de um aluno, e armazene numa lista. Exiba essa lista no final.

1. Faça um programa que receba o nome, o email, e duas notas de um aluno, e armazene numa lista. Exiba essa lista no final.

```
# Criando uma lista para armazenar os dados de um aluno:
dados_aluno = []

# Recebendo os dados do aluno:
nome = input( "Digite o nome:" )
email = input( "Digite o email:" )
nota1 = float( input("Digite a nota 1:") )
nota2 = float( input("Digite a nota 2:") )

# Armazenando os dados na lista:
dados_aluno.append(nome)
dados_aluno.append(email)
dados_aluno.append(nota1)
dados_aluno.append(nota2)

# Exibindo a lista:
print(dados_aluno)
```

2. Faça um programa que receba o nome, o email, e duas notas de um aluno, e armazene numa lista.

Armazene essa lista dentro de uma lista “maior”, chamada de “banco_de_dados”.

Repita a entrada de dados de alunos enquanto o nome do aluno for diferente de “fim”.

Ao final, exiba a lista “banco_de_dados”.

3. Faça um programa que receba o nome, o email, e duas notas de um aluno, e armazene numa lista.

Armazene essa lista dentro de uma lista “maior”, chamada de “banco_de_dados”.

Repita a entrada de dados de alunos enquanto o nome do aluno for diferente de “fim”.

**Ao final, exiba a lista “banco_de_dados”,
mas de um jeito “bonitinho” (exibir os dados, não a lista!!).**

4. Faça um programa que receba o nome, o email, e duas notas de um aluno, e armazene numa lista.

Armazene essa lista dentro de uma lista “maior”, chamada de “banco_de_dados”.

Repita a entrada de dados de alunos enquanto o nome do aluno for diferente de “fim”.

Ao final, exiba a lista “banco_de_dados”, mas de um jeito “bonitinho” (os dados, não a lista!!).

Peça para o usuário digitar um nome, procure no banco de dados.
Se achar exiba seus dados,
se não achar, exiba “não encontrei”.

Exercícios



ATENÇÃO!!

- Em todos os exercícios onde o usuário deve fornecer a quantidade de elementos da lista (N) e os elementos, qualquer operação que deva ser feita (p.ex. achar o menor elemento, achar o maior elemento, obter a soma dos elementos etc) deve ser implementada com a **varredura da lista montada e não durante o laço de leitura** dos elementos
- Então os passos obrigatórios são:
 - monta a lista
 - dar o `print(lista)` para ver se a lista foi construída direitinho
 - implementar as outras coisas

listas simples

Exercícios

1. Crie uma lista de números inteiros de 1 a 15 utilizando a função range.
2. Crie uma lista de 10 números reais fornecidos pelo usuário.
3. Solicite ao usuário para fornecer uma frase com palavras separadas por espaços. Utilize o método `split()` para gerar uma lista a partir da frase informada e observe o resultado.
4. A partir de uma lista de N números inteiros, onde N e os valores inteiros devem ser fornecidos pelo usuário, mostre qual é o maior valor e em qual índice da lista ele se encontra.
5. A partir de uma lista de N números reais, onde N e os valores reais devem ser fornecidos pelo usuário, mostre qual é o menor valor e em qual índice da lista ele se encontra.
6. Faça um programa que mostre na tela uma lista contendo a tabuada de um número N, fornecido pelo usuário. A tabuada deverá ser de 1 a 10.

Exercícios

7. Dada uma lista L, contendo N elementos inteiros, fornecidos pelo usuário, faça um programa que mostre na tela
- a) O maior elemento da lista
 - b) O menor elemento da lista
 - c) A soma de todos os elementos da lista
 - d) Os elementos ímpares
 - e) Os elementos maiores do que 18
 - f) A quantidade de elementos pares
 - g) A média dos valores da lista.
8. Gere uma lista contendo os múltiplos de 3 entre 1 e 150. Mostre a lista gerada e sorteie automaticamente um número da lista (o sorteio deverá ser do índice).

Para gerar um número aleatório:

No início do programa: `import random`

`x = random.randint(a, b)` → gera um número tal que $a \leq x \leq b$

listas em listas

Exercícios

1. Complete o programa visto no slide 78 (exemplo 4) para exibir também a média das notas do aluno, tanto na exibição geral de dados, quanto na exibição do aluno procurado.
2. Complete o programa acima para ao final da execução exibir as seguintes informações:
 - a) Quantidade de alunos na turma
 - b) Nome do aluno com a maior média
 - c) Nome do aluno com a menor média
 - d) A média da nota 1 para a turma toda
 - e) A média da nota 2 para a turma toda
 - f) Quantos alunos estão aprovados (média ≥ 6)
3. Faça um programa que receba os seguintes dados, de 2 alunos:
 - nome
 - nota 1
 - nota 2Os dados deverão estar organizados da seguinte forma:
 - as notas deverão estar em uma lista;
 - os dados do aluno (nome e lista com notas) deverão estar em outra lista;
 - os dados dos 2 alunos (as duas listas acima) deverão estar em uma 3a lista

Exercícios

4. Utilizando como base o programa apresentado nesta aula (cadastro de alunos), faça um programa que deverá apresentar ao usuário um “menu de opções”:

- 1 - inserir dados de aluno
- 2 - procurar aluno
- 3 - exibir todos os alunos
- 4 - finalizar o programa

Um exemplo de execução será apresentado nos slides seguintes.

1 - inserir dados de aluno
2 - procurar aluno
3 - exibir todos os alunos
4 - finalizar o programa
Digite a opcao desejada:|

```
-----  
1 - inserir dados de aluno  
2 - procurar aluno  
3 - exibir todos os alunos  
4 - finalizar o programa  
Digite a opcao desejada:1  
Digite o nome:Carlos  
Digite o email:carlos@gmail  
Digite a nota 1:7  
Digite a nota 2:8
```

```
-----  
1 - inserir dados de aluno  
2 - procurar aluno  
3 - exibir todos os alunos  
4 - finalizar o programa  
Digite a opcao desejada:1  
Digite o nome:Silvana  
Digite o email:sil@ifsp  
Digite a nota 1:4  
Digite a nota 2:3
```

```
-----  
1 - inserir dados de aluno  
2 - procurar aluno  
3 - exibir todos os alunos  
4 - finalizar o programa  
Digite a opcao desejada:|
```

```
-----
1 - inserir dados de aluno
2 - procurar aluno
3 - exibir todos os alunos
4 - finalizar o programa
Digite a opcao desejada:2
Digite o nome a procurar:Carlos
-----
Aluno encontrado!
['Carlos', 'carlos@gmail', 7.0, 8.0]
-----
1 - inserir dados de aluno
2 - procurar aluno
3 - exibir todos os alunos
4 - finalizar o programa
Digite a opcao desejada:2
Digite o nome a procurar:Silvana
-----
Aluno encontrado!
['Silvana', 'sil@ifsp', 4.0, 3.0]
-----
1 - inserir dados de aluno
2 - procurar aluno
3 - exibir todos os alunos
4 - finalizar o programa
Digite a opcao desejada:2
Digite o nome a procurar:Asdrubal
-----
Não encontrei!
-----
1 - inserir dados de aluno
2 - procurar aluno
3 - exibir todos os alunos
4 - finalizar o programa
Digite a opcao desejada:|
```

```
-----  
1 - inserir dados de aluno  
2 - procurar aluno  
3 - exibir todos os alunos  
4 - finalizar o programa  
Digite a opcao desejada:3
```

```
-----  
Nome: Carlos  
Email: carlos@gmail  
Nota1: 7.0  
Nota2: 8.0
```

```
-----  
Nome: Silvana  
Email: sil@ifsp  
Nota1: 4.0  
Nota2: 3.0
```

```
-----  
1 - inserir dados de aluno  
2 - procurar aluno  
3 - exibir todos os alunos  
4 - finalizar o programa  
Digite a opcao desejada:|
```

```
-----  
1 - inserir dados de aluno  
2 - procurar aluno  
3 - exibir todos os alunos  
4 - finalizar o programa  
Digite a opcao desejada:4  
  
**Fim do Programa**  
/// |
```

Exercícios que valem a presença nas aulas desta semana

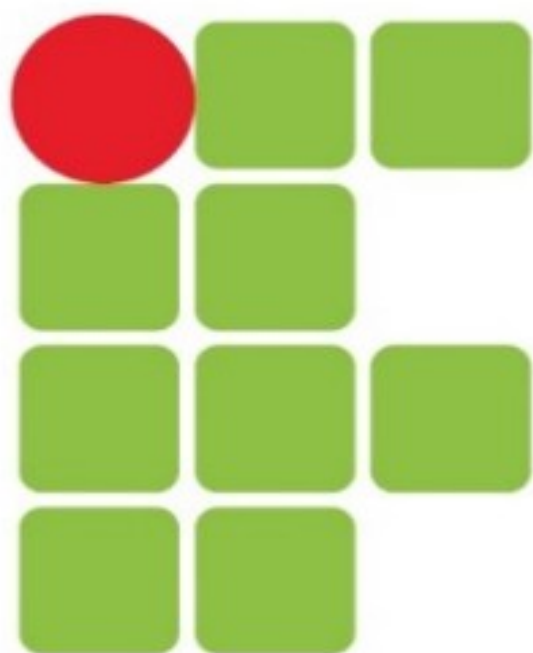
- Obrigatório apenas para quem não esteve “presente” em qualquer das aulas on-line (ou nas duas...)
- Exercícios 7 e 8, slide 82

Exercícios

7. Dada uma lista L, contendo N elementos inteiros, fornecidos pelo usuário, faça um programa que mostre na tela
 - a) O maior elemento da lista
 - b) O menor elemento da lista
 - c) A soma de todos os elementos da lista
 - d) Os elementos ímpares
 - e) Os elementos maiores do que 18
 - f) A quantidade de elementos pares
 - g) A média dos valores da lista.
8. Gere uma lista contendo os múltiplos de 3 entre 1 e 150. Mostre a lista gerada e sorteie automaticamente um número da lista (o sorteio deverá ser do índice).

Para gerar um número aleatório:
No início do programa: `import random`
`x = random.randint(a, b)` → gera um número tal que $a \leq x \leq b$

- Junte tudo em um arquivo compactado, e submeta no Moodle
- Prazo: 23:55 de 9.maio (Domingo)



**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
SÃO PAULO
Câmpus São Carlos