16 Reunião 16: 15/JUN/2021

16.1 Reuniões passadas

- tipos: `int`, `str`, `float`, `bool`, `NoneType`
- comandos de seleção: `if`, `if-else`, `if-elif-else`
- comandos de repetição `while`
- funções permitem **quebra uma tarefa** em tarefas menores
- funções permitem darmos **nome a um conjunto de comandos**
- funções eliminam trechos semelhantes de código
- funções permitem **testar mais facilmente** componentes de um programa
- funções bem projetadas **podem ser usadas** por vários programas (Aguardem!)
- módulos: `import math` `math.sqrt(x)`
- `float` × `int` × aproximações

16.2 Hoje

Figure 1: Fonte: Herbert José de Souza (Henfil)

Listas

Construtores

- Criamos listas

```
lst-vazia = []
lst = [1, 2, 3, -10]
```

índices

- acessamos elementos das listas

```
lst [2]
```

índices

[7, "oi", 3.14, True]

+ e *

- Concatenamos

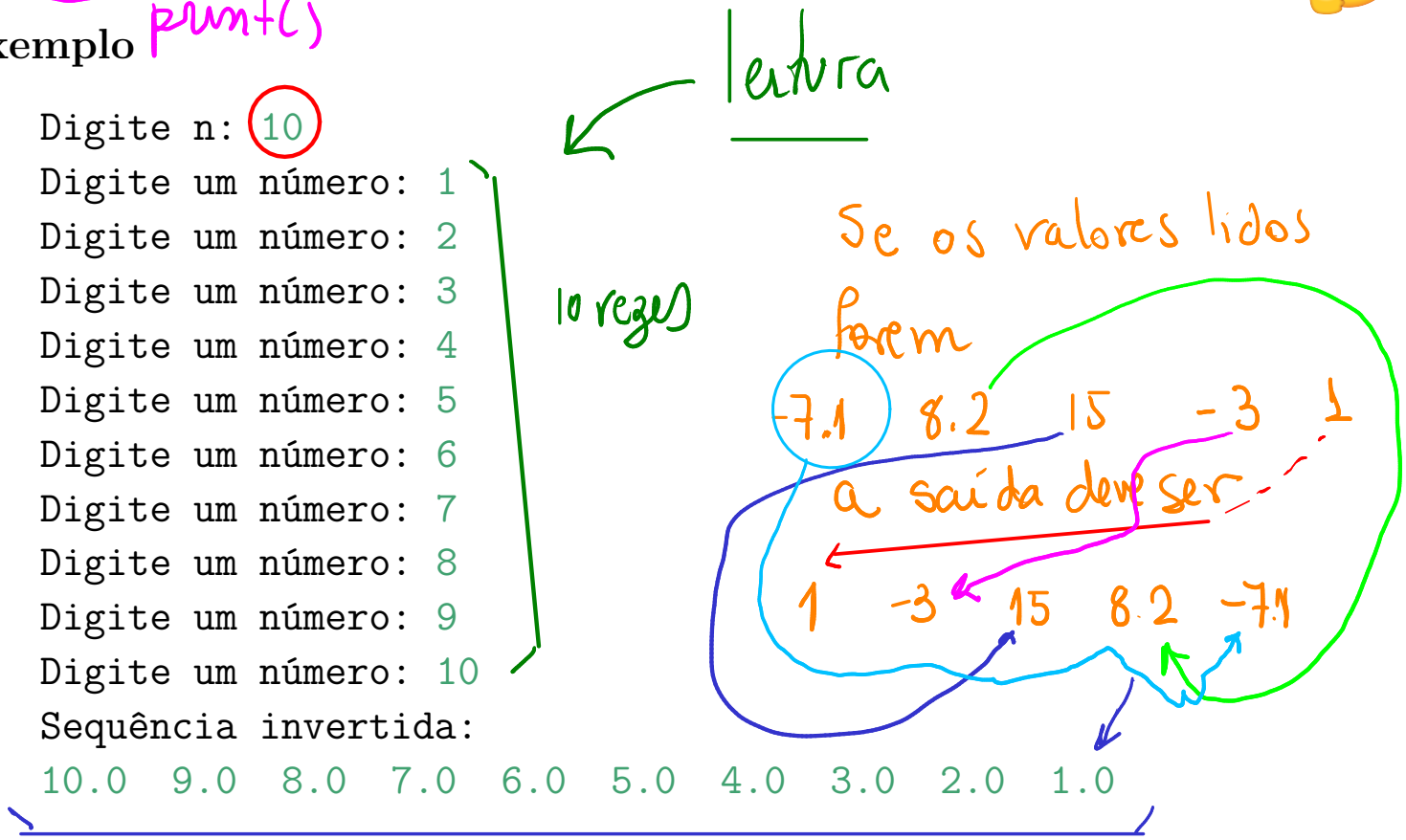
```
[3, 5, 6] + ["oi", False]
= [3, 5, 6, "oi", False]
[3] * 3 = [3] + [3] + [3] = [3, 3, 3]
```

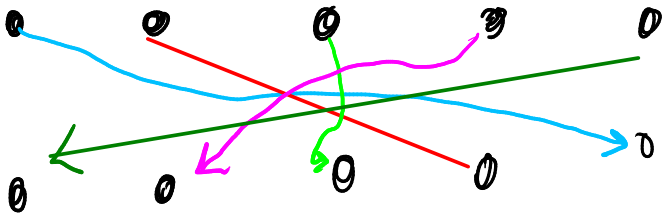
16.3 Exercício: sequência invertida

Dados `n > 0` e uma sequência com `n` números reais, imprimi-los na ordem inversa a da leitura.

Exemplo

```
Digite n: 10
Digite um número: 1
Digite um número: 2
Digite um número: 3
Digite um número: 4
Digite um número: 5
Digite um número: 6
Digite um número: 7
Digite um número: 8
Digite um número: 9
Digite um número: 10
Sequência invertida:
10.0 9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0 1.0
```





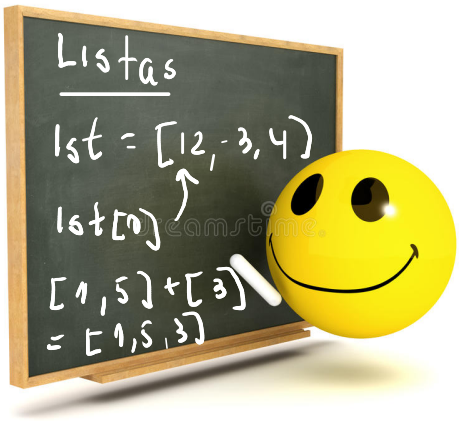


Figure 2: Fonte: Dreamstime.com

Solução

```
#----------------------------------------
# Programa principal
def main():
    '''
    Programa que LÊ um inteiro n > 0 e uma sequência
    com n números reais e IMPRIME os número na ordem
    inversa a da leitura.
    '''
    n = int(input("Digite n: "))
    lst = [] # lista vazia
    i = 0
    while i < n:
        x = int(input("Digite um número: "))
        lst = lst + [x]
        i = i + 1

    print("Sequência invertida: ")
    i = n-1
    while i >= 0:
        print(f"{lst[i]} ", end='')
        i = i - 1
    print() # muda de linha

#----------------------------------------
if __name__ == "__main__":
    main()
```

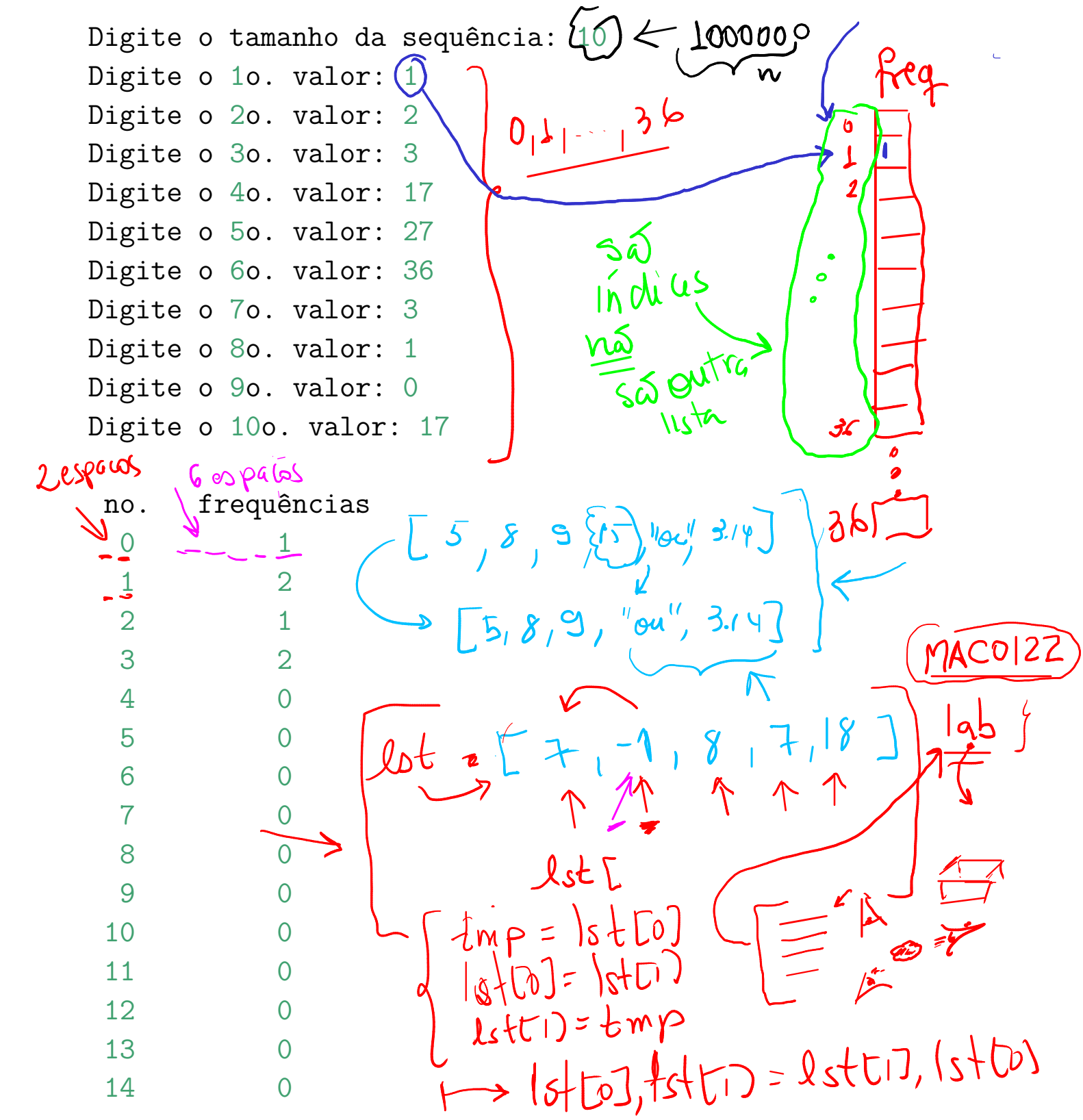
16.4 Exercício: frequências

Dados $n > 0$ e uma sequência com `n` números inteiros entre 0 e 36, calcular o número de ocorrências da cada valor.

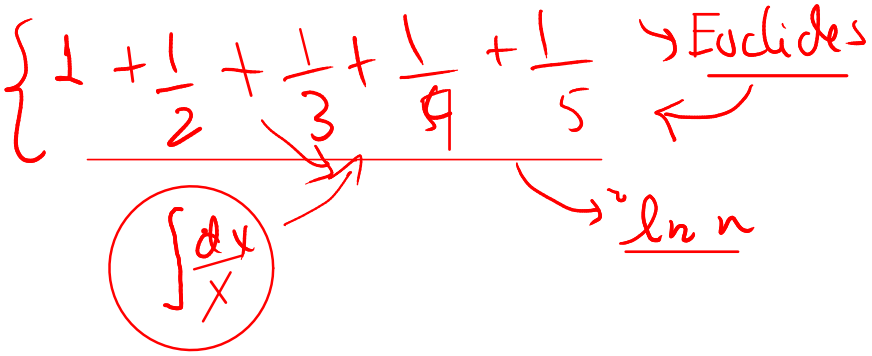
Exemplo

```
Digite o tamanho da sequência: 10
Digite o 1o. valor: 1
Digite o 2o. valor: 2
Digite o 3o. valor: 3
Digite o 4o. valor: 17
Digite o 5o. valor: 27
Digite o 6o. valor: 36
Digite o 7o. valor: 3
Digite o 8o. valor: 1
Digite o 9o. valor: 0
Digite o 10o. valor: 17
```

```
 no.  frequências
  0         1
  1         2
  2         1
  3         2
  4         0
  5         0
  6         0
  7         0
  8         0
  9         0
 10         0
 11         0
 12         0
 13         0
 14         0
```



15	0
16	0
17	2
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	1
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	0
36	1




Frequências
5 36 0 1 4 36
n seq

Solução

```
def main():
    '''
    Programa que LÊ um inteiro n e uma sequência
    de inteiros entre 0 e 36 e IMPRIME o número
    de ocorrências (frequência) de cada valor.
    '''
    # leia o tamanho da sequência
    n = int(input("Digite o tamanho da sequência: "))

    # crie uma lista para as ocorrências [0..36]
    freq = [0]*37 # crie uma lista
    i = 0
    while i < n:
        valor = int(input(f"Digite o {i+1}o. valor: "))
        freq[valor] += 1
        i += 1

    print('\n no.   frequências ')
    i = 0
    while i < 37:
        print(f" {i:2}     {freq[i]:6}")
        i += 1

#---------------------------------------------------
if __name__ == "__main__":
    main()
```

outra maneira

```
freq = [0]
i = 0
while i < 37:
    freq += [0]
    i += 1
```

mais outra maneira

```
freq = [0,0,0,...,0]
```

37 zeros

Contadores

NOVO

↑ o mesmo que [0]+[0]+[0]+...+[0]

37 vezes

número a ser lido

usa valor como índice !!

muda de linha

6 espaços para escrever freq[i]

1 2 3

2 espaços para escrever i

2
3 6

Exemplo: versão *de luxe*

```
Digite o tamanho da sequência: 10
Digite o 1o. valor: 1
Digite o 2o. valor: 2
Digite o 3o. valor: 3
Digite o 4o. valor: 17
Digite o 5o. valor: 27
Digite o 6o. valor: 36
Digite o 7o. valor: 3
Digite o 8o. valor: 1
Digite o 9o. valor: 0
Digite o 10o. valor: 17

 no.   frequências
  0          1
  1          2
  2          1
  3          2
 17          2
 27          1
 36          1
Frequência zero: 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18,
19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35
```


Frequências
versão deLuxe
if freq[i]!=0

Solução

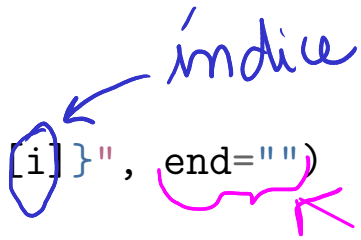
```
def main():
    '''
    Programa que lê um inteiro n e uma sequência
    de inteiros entre 0 e 36 e imprime o número
    de ocorrências (frequência) de cada valor.
    '''

    # leia o tamnho da sequência
    n = int(input("Digite o tamanho da sequência: "))

    # crie uma lista para as ocorrências [0..36]
    freq = [0]*37 # crie uma lista

    i = 0
    while i < n:
        valor = int(input(f"Digite o {i+1}o. valor: "))
        freq[valor] += 1
        i += 1

    print("\n no.   frequências ")
    freq_zero = []
    i = 0
    while i < 37:
        if freq[i] > 0:
            print(f" {i:2}    {freq[i]:6}")
        else:
            freq_zero += [i]
        i += 1

    if len(freq_zero) > 0:
        print("Frequência zero: ", end="")
        k = len(freq_zero)
        print(f"{freq_zero[0]}", end="")
```

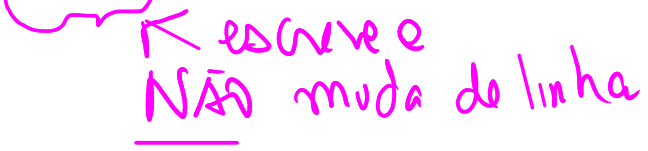
lista de valores com frequência 0

NOVO

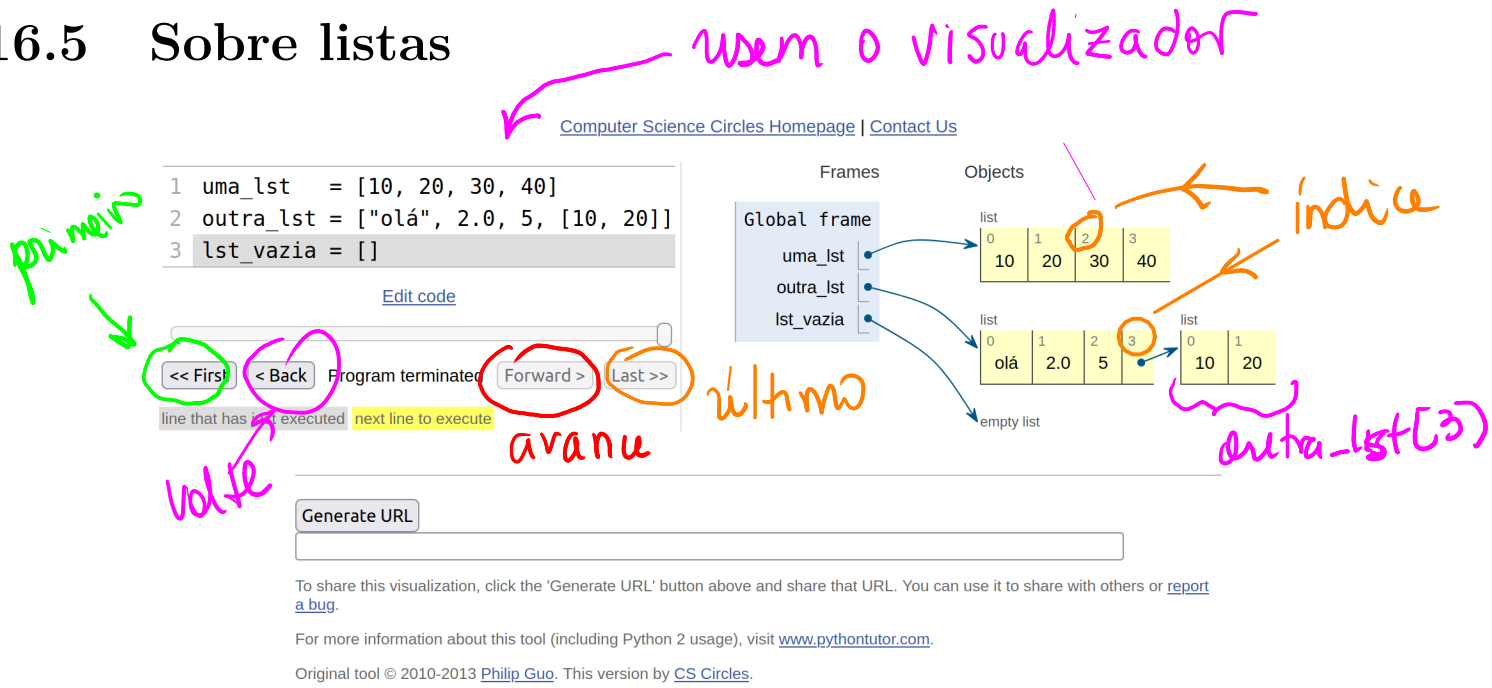
NOVO

```
        i = 1
        while i < k:
            print(f", {freq_zero[i]}", end="")
            i += 1
        print() # pula linha

#-------------------------------------------------------
if __name__ == "__main__":
    main()
```





16.5 Sobre listas



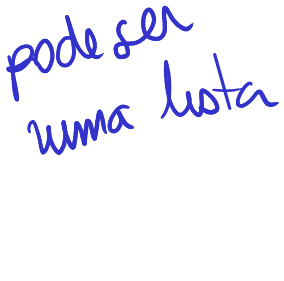
Uma **lista** em Python é uma sequência ou coleção ordenada de valores de qualquer tipo.

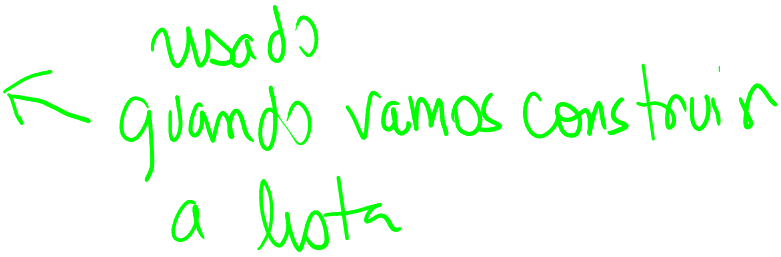
```
[10, 20, 30, 40]
 0   1   2   3
```

Nasce uma estrela

Existem várias maneiras de criarmos uma lista.
A maneira simple é envolver os elementos da
lista por colchetes ([e]).

```
In [1]: uma_lst   = [10, 20, 30, 40]

In [2]: uma_lst
Out[2]: [10, 20, 30, 40]

In [3]: outra_lst = ["olá", 2.0, 5, [10, 20]]

In [4]: outra_lst
Out[4]: ['olá', 2.0, 5, [10, 20]]

In [5]: lst_vazia = []

In [6]: lst_vazia
Out[6]: []
```





Comprimento

A função `len()` retorna o **comprimento** de uma lista, o número de elementos na lista.

```
In [5]: print(len(uma_lst))
4

In [6]: len(uma_lst)
Out[6]: 4

In [7]: len(outra_lst)
Out[7]: 4

In [8]: len(lst_vazia)
Out[8]: 0
```

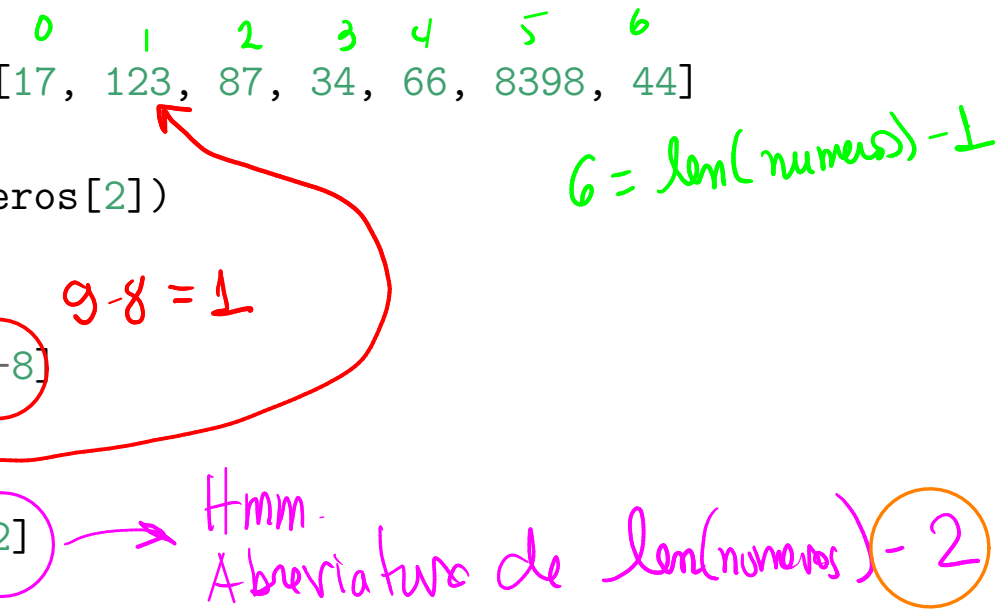
Acessar elementos

Cada valor na lista é identificado por um **índice**.

Para acessar um elemento de uma lista usamos o operador de indexação `[]`.

A expressão dentro dos conchetes especifica o índice.
O índice do **primeiro elemento** é 0. O índice do **último elemento** é `len(lst)-1`.

Índices negativos indicarão elementos da direita para a esquerda ao invés de da esquerda para a direita.




```
In [9]: numeros = [17, 123, 87, 34, 66, 8398, 44]

In [10]: print(numeros[2])
87

In [11]: numeros[9-8]
Out[11]: 123

In [12]: numeros[-2]
Out[12]: 8398

In [13]: numeros[len(numeros)-1]
Out[13]: 44

In [14]: uma_lst = [3, 67, "gato", [56, 57, "cachorro"], [ ], 3.

In [15]: print(uma_lst[2][0])
g

In [16]: uma_lst[2][0]
Out[16]: 'g'

In [17]: uma_lst = [ [4, [True, False], 6, 8], [888, 999] ]

In [18]: uma_lst[0]
```

```
Out[18]: [4, [True, False], 6, 8]
```

16.5.1 Concatenações + e *

O operador + concatena duas listas

```
n [19]: uma_lst   = [10, 20, 30, 40]

In [20]: outra_lst = ['oi', True, None]

In [21]: lst_nova = uma_lst + outra_lst

In [22]: lst_nova
Out[22]: [10, 20, 30, 40, 'oi', True, None]
```

O operador * repete a concatenação de uma lista um dado número inteiro de vezes

```
In [26]: lst_zeros = [0] * 5 # [0] + [0] + [0] + [0] + [0]

In [27]: lst_zeros
Out[27]: [0, 0, 0, 0, 0]
```