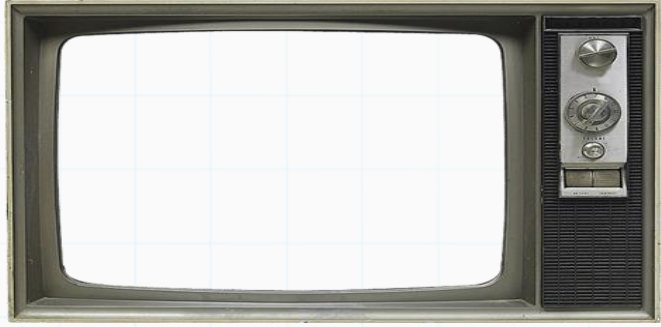
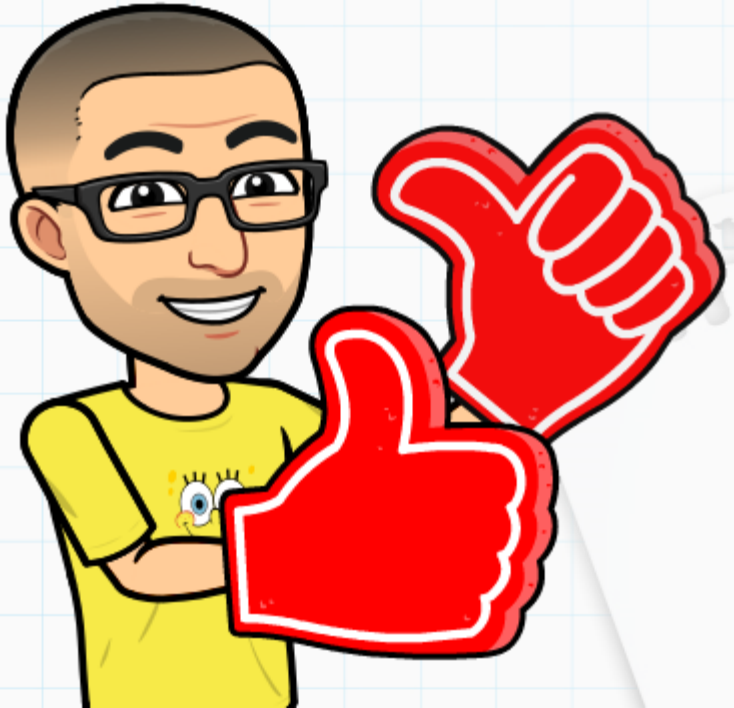
Programação De Computadores

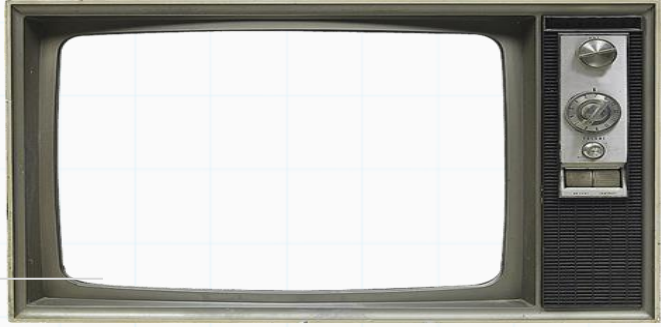


Professor : Yuri Frota

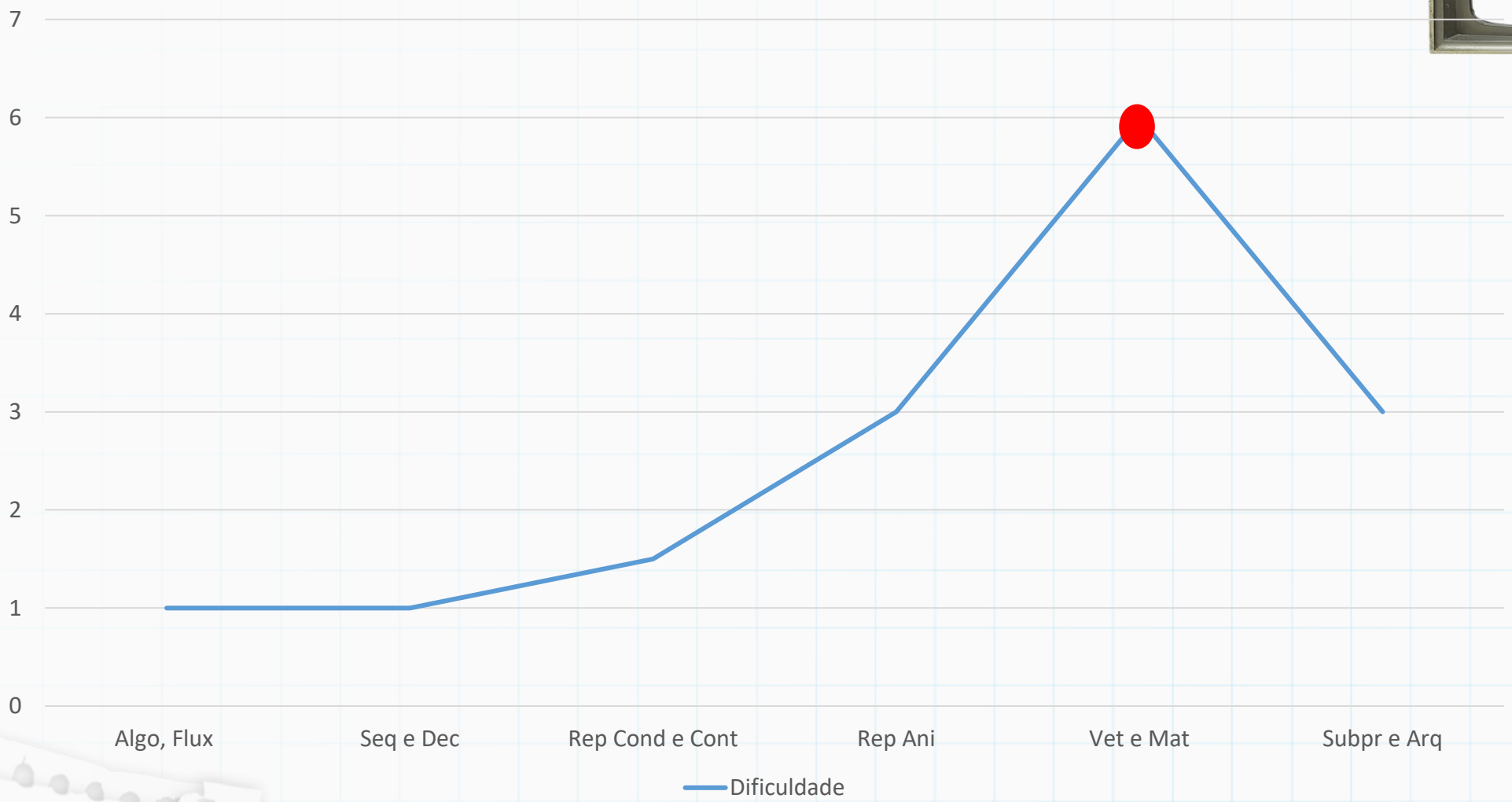
www.ic.uff.br/~yuri/prog.html

yuri@ic.uff.br

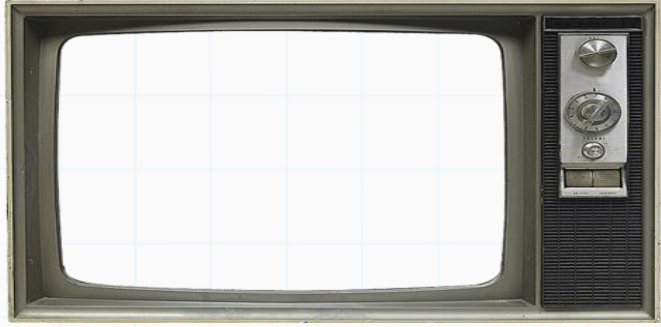




Percepção de Dificuldade dos Alunos



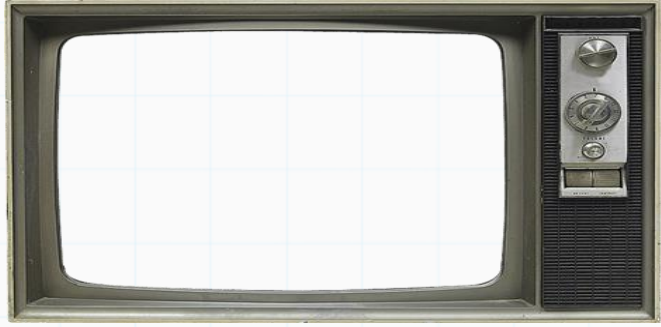
Vetores



Suponha você professor que tenha que armazenar num programa as notas de suas 5 alunos:

```
1  nota1 = float(input("Entre com a nota 1: "))
2  nota2 = float(input("Entre com a nota 2: "))
3  nota3 = float(input("Entre com a nota 3: "))
4  nota4 = float(input("Entre com a nota 4: "))
5  nota5 = float(input("Entre com a nota 5: "))
```

Vetores



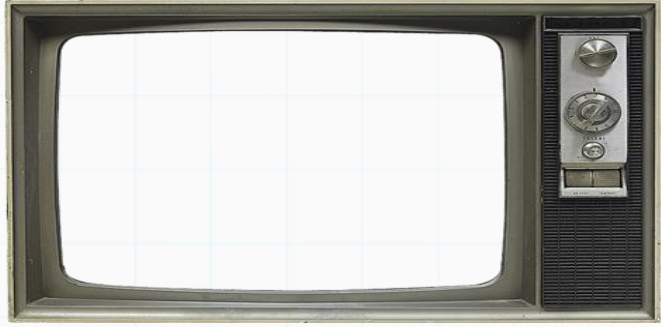
Mas e se fossem 15 alunos?

```
1  nota1 = float(input("Entre com a nota 1: "))
2  nota2 = float(input("Entre com a nota 2: "))
3  nota3 = float(input("Entre com a nota 3: "))
4  nota4 = float(input("Entre com a nota 4: "))
5  nota5 = float(input("Entre com a nota 5: "))
6  nota6 = float(input("Entre com a nota 6: "))
7  nota7 = float(input("Entre com a nota 7: "))
8  nota8 = float(input("Entre com a nota 8: "))
9  nota9 = float(input("Entre com a nota 9: "))
10 nota10 = float(input("Entre com a nota 10: "))
11 nota11 = float(input("Entre com a nota 11: "))
12 nota12 = float(input("Entre com a nota 12: "))
13 nota13 = float(input("Entre com a nota 13: "))
14 nota14 = float(input("Entre com a nota 14: "))
15 nota15 = float(input("Entre com a nota 15: "))
```

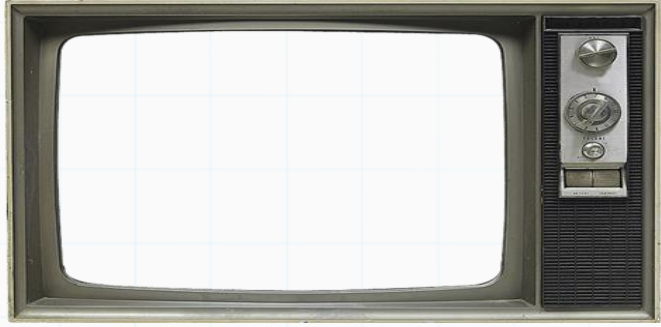
Vetores

Mas e se fossem 50 alunos?

```
1  nota1 = float(input("Entre com a nota 1: "))
2  nota2 = float(input("Entre com a nota 2: "))
3  nota3 = float(input("Entre com a nota 3: "))
4  nota4 = float(input("Entre com a nota 4: "))
5  nota5 = float(input("Entre com a nota 5: "))
6  nota6 = float(input("Entre com a nota 6: "))
7  nota7 = float(input("Entre com a nota 7: "))
8  nota8 = float(input("Entre com a nota 8: "))
9  nota9 = float(input("Entre com a nota 9: "))
10 nota10 = float(input("Entre com a nota 10: "))
11 nota11 = float(input("Entre com a nota 11: "))
12 nota12 = float(input("Entre com a nota 12: "))
13 nota13 = float(input("Entre com a nota 13: "))
14 nota14 = float(input("Entre com a nota 14: "))
15 nota15 = float(input("Entre com a nota 15: "))
```



Vetores



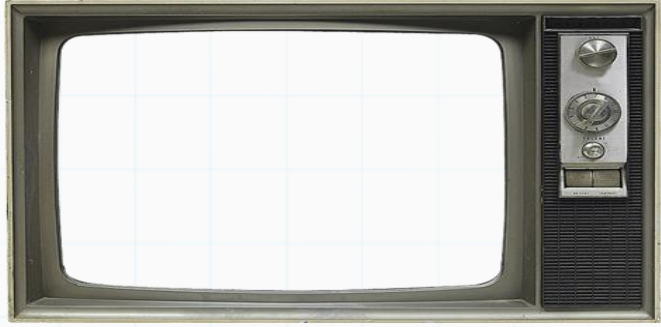
- É possível definir variáveis que guardam mais de um valor

- Essas variáveis são conhecidas como variáveis compostas, variáveis subscritas, variáveis indexáveis ou arranjos (*array=vetores*)

variável

variável vetor

Vetores



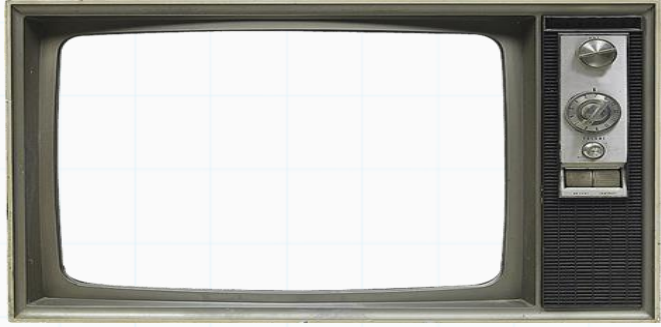
- É possível definir variáveis que guardam mais de um valor

- Essas variáveis são conhecidas como variáveis compostas, variáveis subscritas, variáveis indexáveis ou arranjos (*array=vetores*)

- Em Python existem três tipos principais de variáveis compostas (com a mesma lógica), cada uma com suas características especiais:

Listas ← foco do curso
Tuplas
Dicionários

Listas

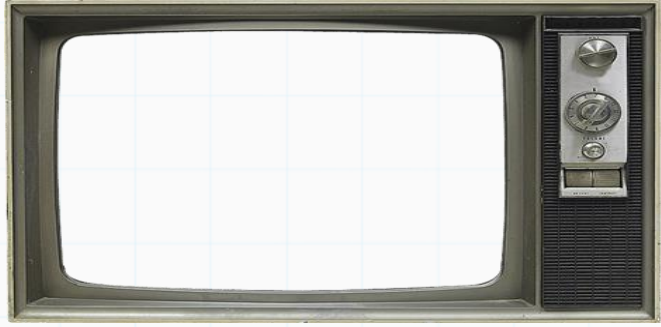


- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

- Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

```
1 lista1 = [10, 20, 30, 40]
```

Listas



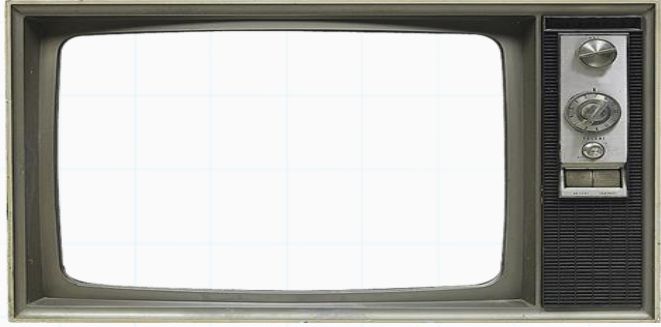
- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**
    - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

```
1 lista1 = [10, 20, 30, 40]
```

- Em outras linguagens de programação, listas são chamadas de **vetores** e possuem restrições que Python não impõe:
    - Em Python, os valores de uma lista podem ser de qualquer tipo
    - Em outras linguagens, os valores precisam ser do mesmo tipo

```
2 lista2 = ["programação", "Freeza", "python"]
3 lista3 = [4, True, "Adamantium"]
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro

```
1  lista1 = [10,   20,   30,   40,   50]
```

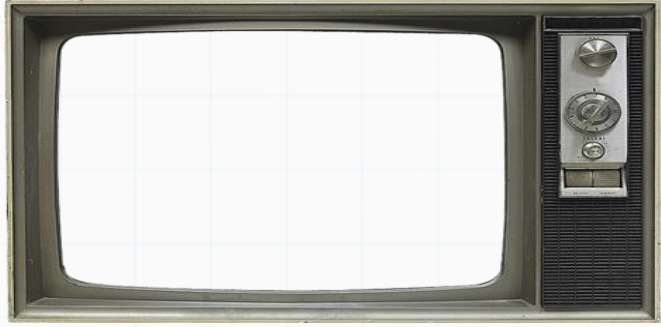
ind = 0 1 2 3 4

```
4  print(lista1[0])
5  print(lista1[4])
```

Shell

```
10
50
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro

```
1  lista1 = [10,   20,   30,   40,   50]
```

ind = 0 1 2 3 4

```
4  print(lista1[0])
5  print(lista1[4])
```

Shell

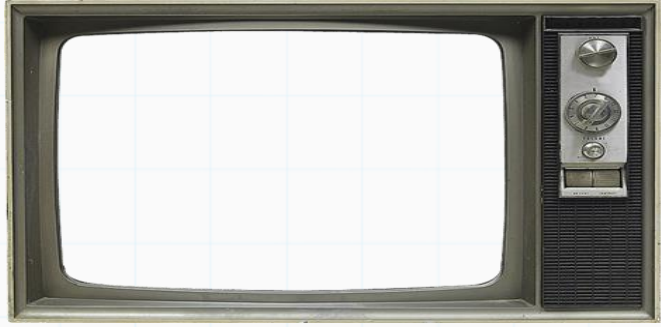
```
10
50
```

```
4  print(lista1)
5
```

Shell

```
[10, 20, 30, 40, 50]
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

- Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

- É acessada via um índice inteiro

```
1  lista1 = [10,   20,   30,   40,   50]
```

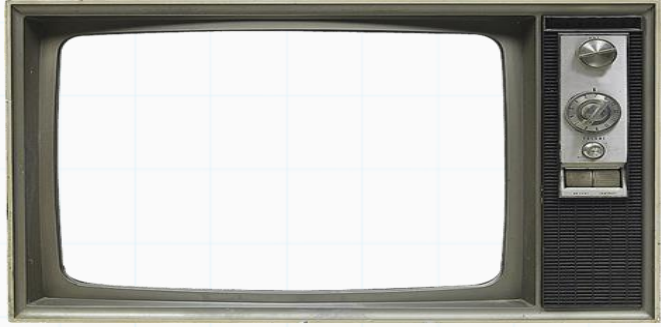
```
ind =  0     1     2     3     4
```

```
5  print(lista1[5])
```

Shell

```
Traceback (most recent call last):
  File "C:\Users\Yuri\Desktop\teste.py", line 4, in <module>
    print(lista1[5])
IndexError: list index out of range
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro (índice inverso)

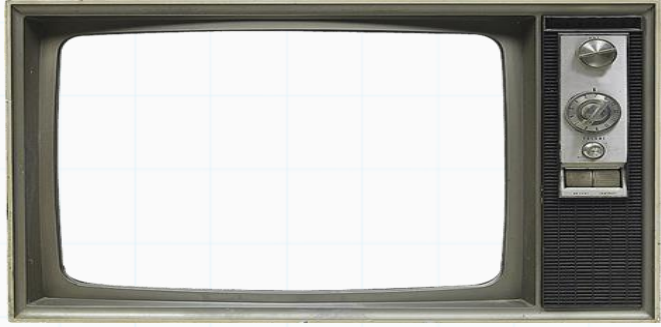
```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

```
5  print(lista1[-3])
```

Shell

```
30
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro (índice inverso)

```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

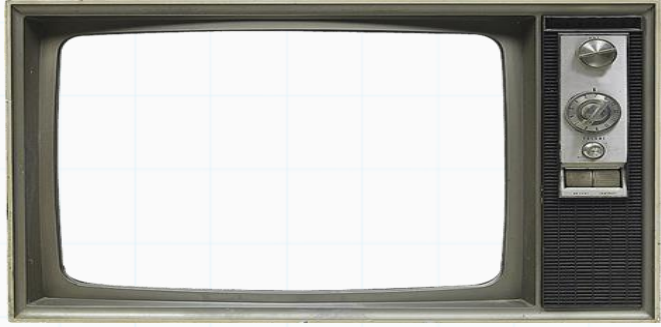
- Geralmente o acesso das listas é feita em laços para iterar todas as suas posições:

```
4  for i in range (5):
5      print(lista1[i])
```

Shell

```
>>> %Run teste.py
10
20
30
40
50
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro (índice inverso)

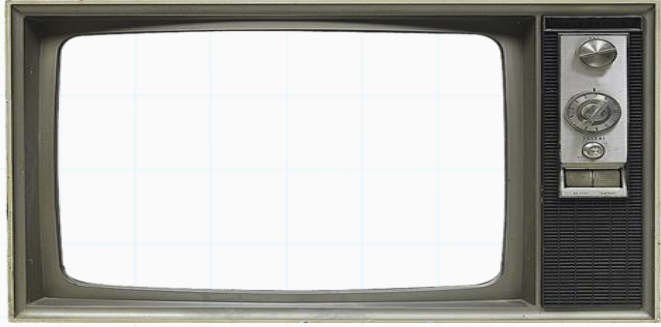
```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

- podemos alterar seus valores

```
4  lista1[1] = 1000
5  lista1[4] = lista1[1]+lista1[0]
6  print(lista1)
7
```

```
Shell
[10, 1000, 30, 40, 1010]
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro (índice inverso)

```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

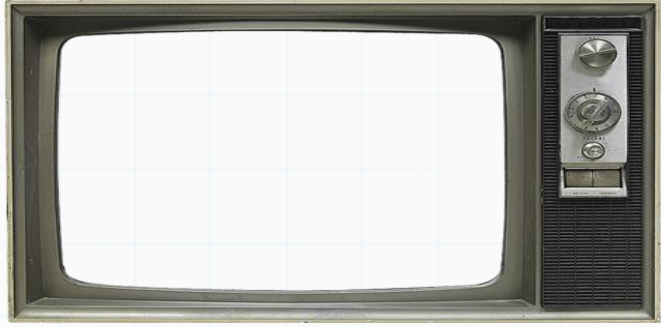
- podemos alterar seus valores

```
4  for i in range (5):
5      lista1[i]=lista1[i]+5
6  print(lista1)
```

Shell

```
[15, 25, 35, 45, 55]
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

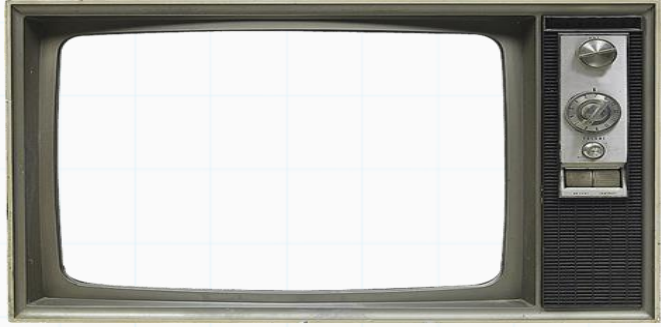
  - É acessada via um índice inteiro (índice inverso)

```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

- podemos alterar seus valores

```
4  for i in range (5):
5      lista1[i]=lista1[i]+lista1[i+1]
6  print(lista1)
```

Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

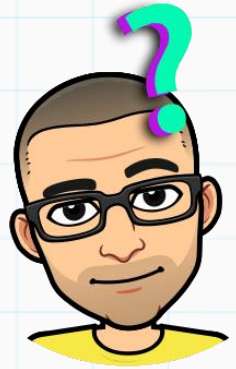
  - É acessada via um índice inteiro (índice inverso)

```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

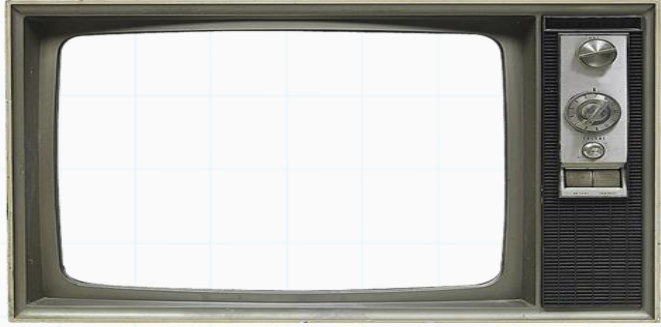
- podemos alterar seus valores

```
4  for i in range (5):
5      lista1[i]=lista1[i]+lista1[i+1]
6  print(lista1)
```

```
Traceback (most recent call last):
  File "C:\Users\Yuri\Desktop\teste.py", line 5, in <module>
    lista1[i]=lista1[i]+lista1[i+1]
IndexError: list index out of range
```



Listas



- Variável composta **unidimensional,** que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**

  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula

  - É acessada via um índice inteiro (índice inverso)

```
ind. inverso =   -5    -4    -3    -2    -1
1  lista1 = [10,   20,   30,   40,   50]
       ind =   0     1     2     3     4
```

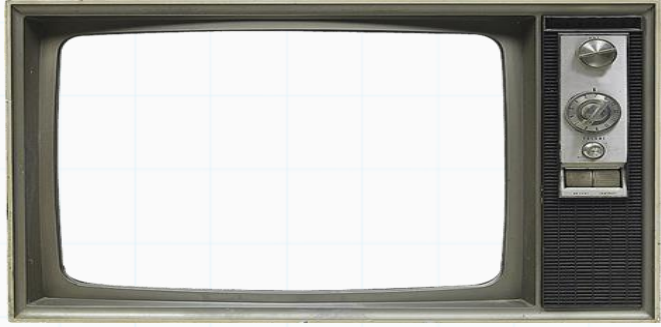
- podemos alterar seus valores

```
4  for i in range (4):
5      lista1[i]=lista1[i]+lista1[i+1]
6  print(lista1)
```

Shell

```
[30, 50, 70, 90, 50]
```

Listas

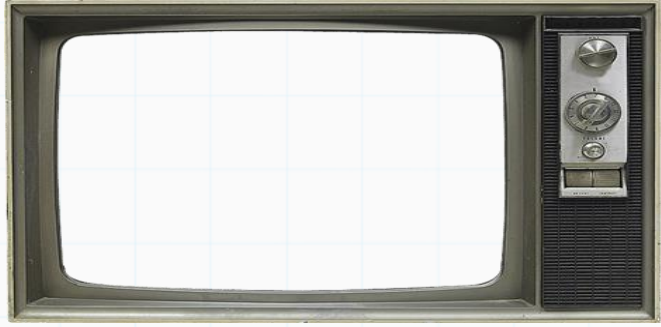


-Existem 3 tipos de inicialização de listas:

- direta com valores

```
2 notas = [8.0,5.5,1.5]
```

Listas



-Existem 3 tipos de inicialização de listas:

- direta com valores

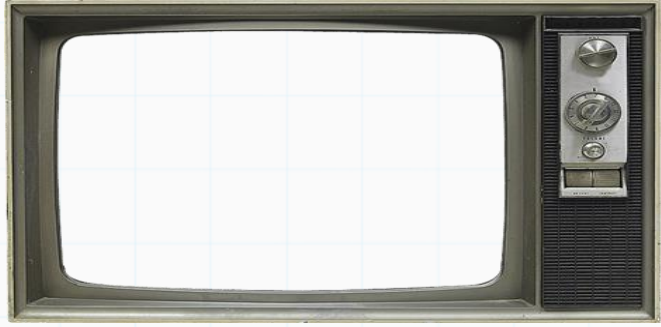
```
2 notas = [8.0,5.5,1.5]
```

- direta com os mesmos valores (duplicados)

```
2 notas = [0] * 10
3 print(notas)
```

```
Shell ×
>>> %Run teste.py
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Listas



-Existem 3 tipos de inicialização de listas:

- direta com valores

```
2 notas = [8.0,5.5,1.5]
```

- direta com os mesmos valores (duplicados)

```
2 notas = [0] * 10
3 print(notas)
```

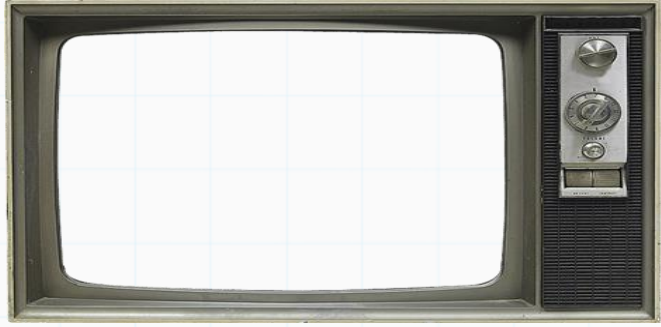
```
Shell
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- vazia (nenhum elemento)

```
2 notas = []
3 print(notas)
```

```
Shell
[]
```

Listas



-A função **len** retorna o número de itens na lista :

```
2  notas = [8.0,5.5,1.5]
3  print(len(notas))
```

Shell

```
>>> %Run teste.py
3
```

```
2  notas = []
3  print(len(notas))
```

Shell

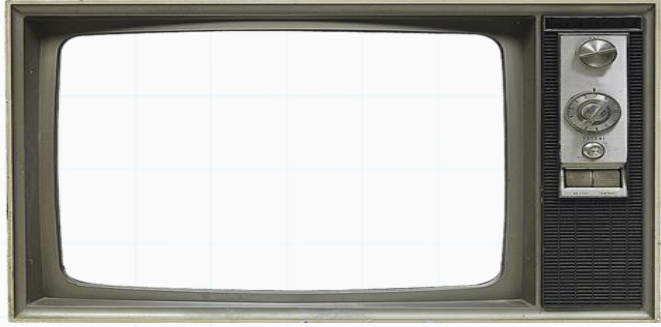
```
>>> %Run teste.py
0
```

```
2  notas = [8.0,5.5,1.5]
3  for i in range (len(notas)):
4      print(notas[i])
```

Shell

```
>>> %Run teste.py
8.0
5.5
1.5
```

Listas



-Aumentando o tamanho da lista (comando append) :

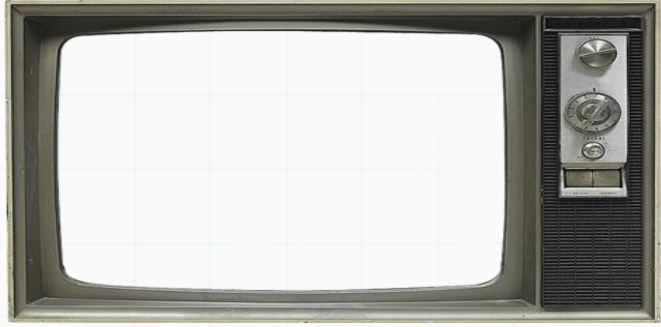
insere no fim da lista

```
1 notas = []
2 notas.append(9)
3 print(notas)
```

Shell

```
[9]
```

Listas



-Aumentando o tamanho da lista (comando append) :

insere no fim da lista

```
1 notas = []
2 notas.append(9)
3 print(notas)
```

Shell

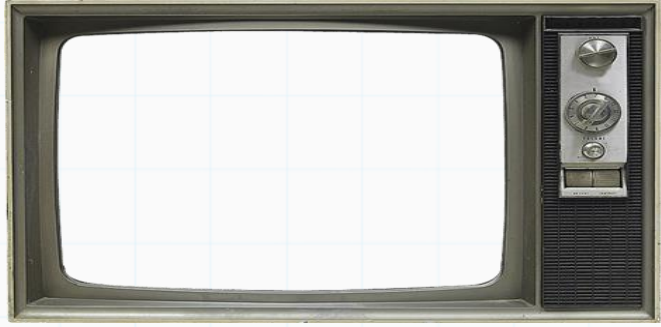
```
[9]
```

```
1 notas = []
2 notas.append(9)
3 notas.append(5.4)
4 notas.append(7.8)
5 for i in range(len(notas)):
6     print(notas[i])
```

Shell

```
>>> %Run teste.py
9
5.4
7.8
```

Listas



-Aumentando o tamanho da lista (comando append) :

insere no fim da lista

```
1 notas = []
2 notas.append(9)
3 print(notas)
```

Shell

```
[9]
```

```
1 notas = []
2 notas.append(9, 5.4)
```

Shell

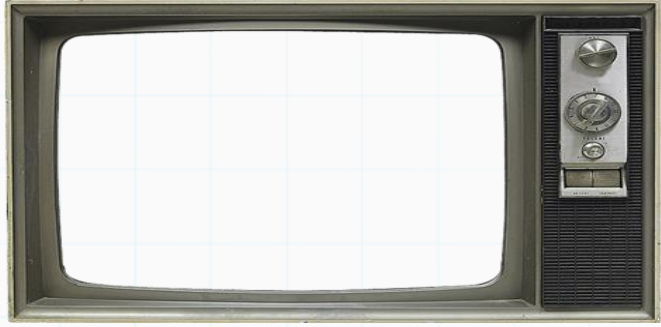
```
>>> %Run teste.py
Traceback (most recent call last):
  File "C:\Users\Yuri\Desktop\teste.py", line 2, in <module>
    notas.append(9, 5.4)
TypeError: append() takes exactly one argument (2 given)
```

```
1 notas = []
2 notas.append(9)
3 notas.append(5.4)
4 notas.append(7.8)
5 for i in range(len(notas)):
6     print(notas[i])
```

Shell

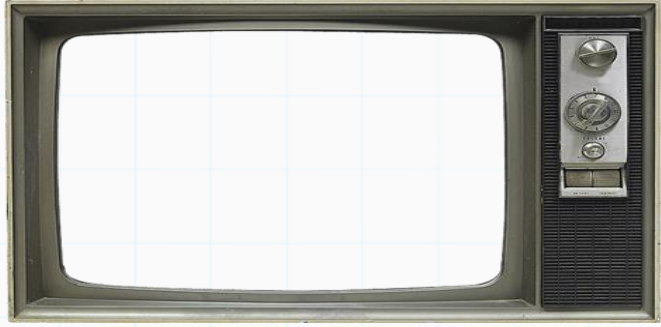
```
>>> %Run teste.py
9
5.4
7.8
```

Listas



-Vamos fazer um programa para guardar os nomes e as notas de 40 alunos e dar os parabéns aqueles que tiraram nota acima da média:

Listas

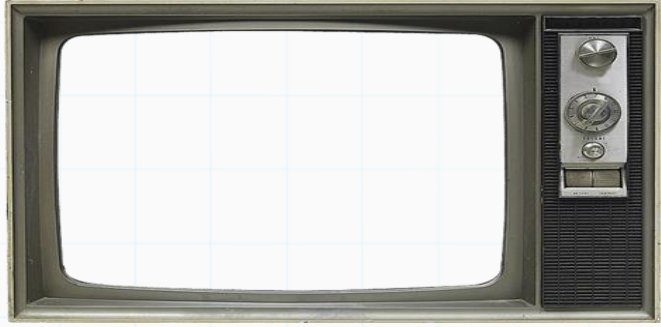


-Vamos fazer um programa para guardar os nomes e as notas de 40 alunos e dar os parabéns aqueles que tiraram nota acima da média:

```
num   = 40
nomes = []
notas = []
media = 0

for i in range(num):
    nomes.append(input('nome: '))
    notas.append(float(input('nota:')))
    media = media + notas[i]

media = media / num
print('A media da turma eh ', media)

for i in range(num):
    if notas[i] > media:
        print('Parabens', nomes[i])
```

código

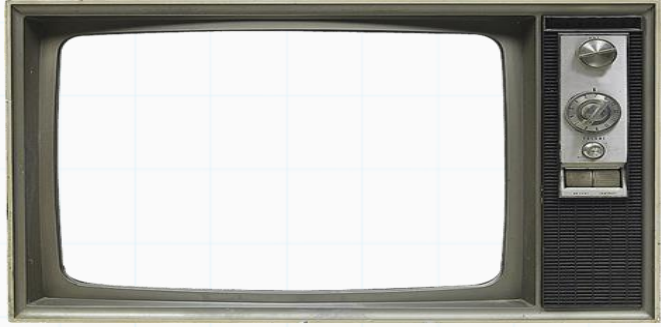
Listas



- Concatenando listas (operador +):

    É possível anexar os valores de uma lista em outra usando o operador “+”

Listas



- Concatenando listas (operador +):

É possível anexar os valores de uma lista em outra usando o operador “+”

```
1 lista = [1,2,3]
2 print(lista)
3 lista = lista + [4]
4 print(lista)
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run teste.py
  [1, 2, 3]
  [1, 2, 3, 4]
```

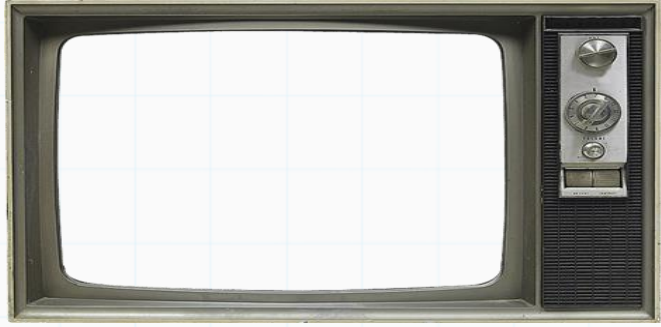
```
1 lista = [1,2,3]
2 print(lista)
3 lista = lista + ["Shun", False]
4 print(lista)
```

Shell

```
>>> %Run teste.py
  [1, 2, 3]
  [1, 2, 3, 'Shun', False]
```

coloca no fim da lista

Listas



- Concatenando listas (operador +):

É possível anexar os valores de uma lista em outra usando o operador “+”

```
1 lista = [1,2,3]
2 print(lista)
3 lista = lista + [4]
4 print(lista)
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run teste.py
 [1, 2, 3]
 [1, 2, 3, 4]
```

```
1 lista = [1,2,3]
2 print(lista)
3 lista = lista + ["Shun", False]
4 print(lista)
```

Shell

```
>>> %Run teste.py
 [1, 2, 3]
 [1, 2, 3, 'Shun', False]
```

coloca no fim da lista

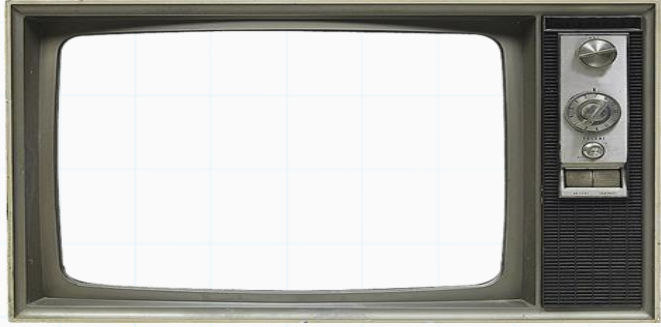
```
1 lista = ['a','b','c']
2 print(lista)
3 lista = ['x','y','z'] + lista
4 print(lista)
```

Shell

```
>>> %Run teste.py
 ['a', 'b', 'c']
 ['x', 'y', 'z', 'a', 'b', 'c']
```

ou no começo

Listas



- Exemplo: programa que retorna uma lista com todos os números pares entre 2 e n

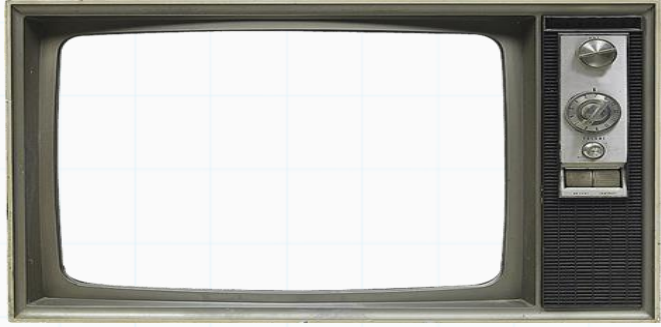
```
1  n = int(input('Digite um numero: '))
2  lista = []
3  for i in range(2,n+1,2):
4      lista = lista + [i]
5  print(lista)
```

Shell

```
>>> %Run teste.py
Digite um numero: 15
[2, 4, 6, 8, 10, 12, 14]
```

código

Listas



- Exemplo: programa que retorna uma lista com todos os números pares entre 2 e n, **em ordem reversa**.

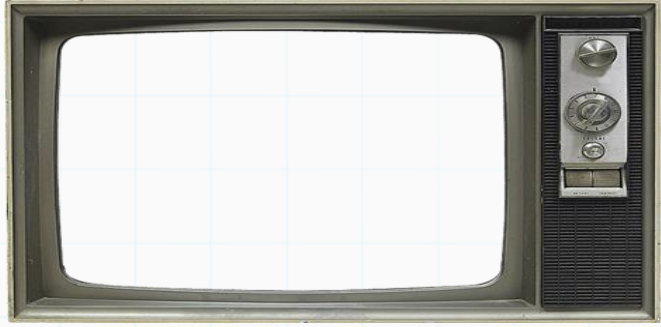
```
1  n = int(input('Digite um numero: '))
2  lista = []
3  for i in range(2,n+1,2):
4      lista = [i] + lista
5  print(lista)
6
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run teste.py

 Digite um numero: 15
 [14, 12, 10, 8, 6, 4, 2]
```

código

Listas



- Multiplicação de listas (operador *):
    - O operador “*” repete **n** vezes os elementos que já estão na lista
    - **lista * n** equivale a **lista + lista + ... + lista** (n vezes)

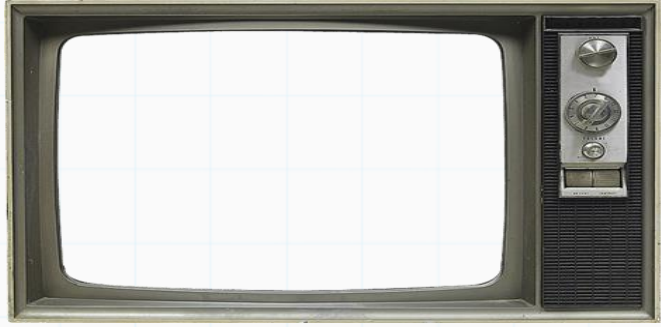
já tínhamos usado esse operador na inicialização das listas

```
2  notas = [0] * 10
3  print(notas)
```

Shell

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Listas



- Multiplicação de listas (operador *):
    - O operador “*” repete **n** vezes os elementos que já estão na lista
    - **lista * n** equivale a **lista + lista + ... + lista** (n vezes)

```
1  lista=[5,10,20]
2  print(lista)
3  lista=lista*3
4  print(lista)
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run teste.py

  [5, 10, 20]
  [5, 10, 20, 5, 10, 20, 5, 10, 20]
```

já tínhamos usado esse operador na inicialização das listas

```
2  notas = [0] * 10
3  print(notas)
```

Shell

```
>>> %Run teste.py

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

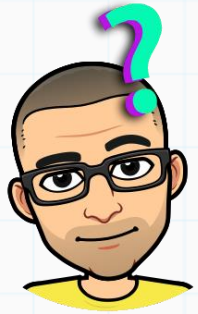
Listas

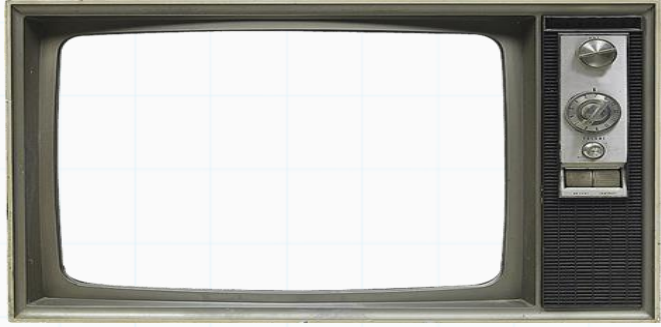
- Cópia de listas

```
1  lista1    = [1,2,3]
2  lista2    = lista1
3  lista2[1] = 100
4  print(lista1)
```

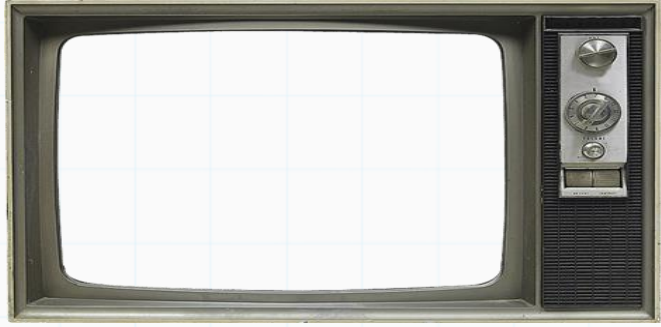
Shell

```
Python 3.7.7 (bundled)
>>> %Run teste.py

  [1, 100, 3]
```



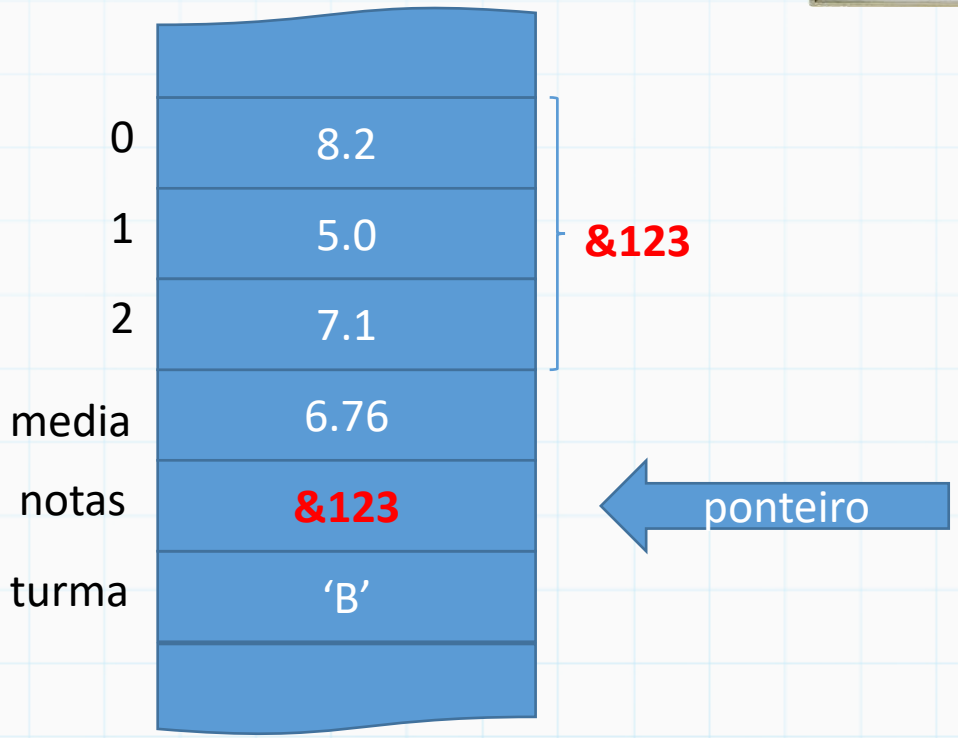


Listas



- Cópia de listas
    - alocação em memória

```
1  notas = [8.2, 5.0, 7.1]
2  turma = 'B'
3  media = 0
4  for i in range(len(notas)):
5      media = media + notas[i]
6  media = media/len(notas)
```



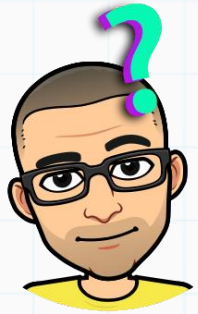
Listas

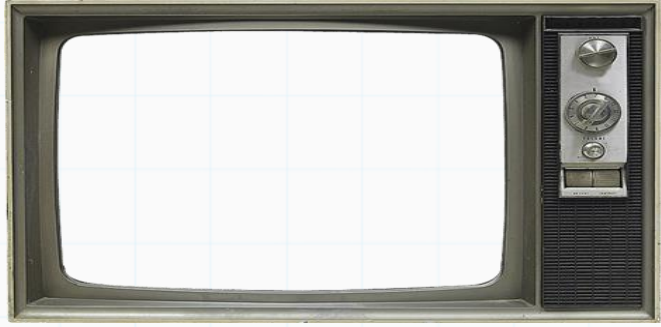
- Cópia de listas

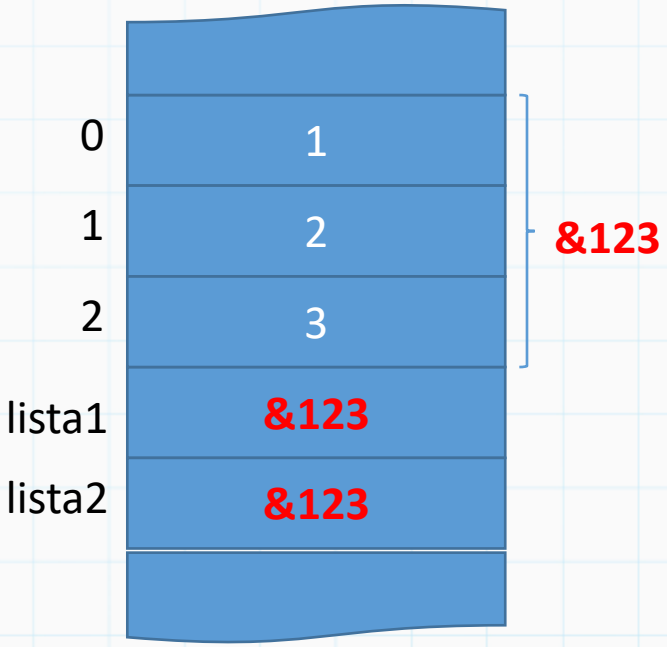
```
1  lista1    = [1,2,3]
2  lista2    = lista1
3  lista2[1] = 100
4  print(lista1)
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run teste.py

  [1, 100, 3]
```







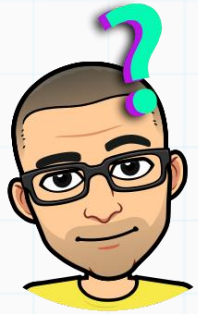
Listas

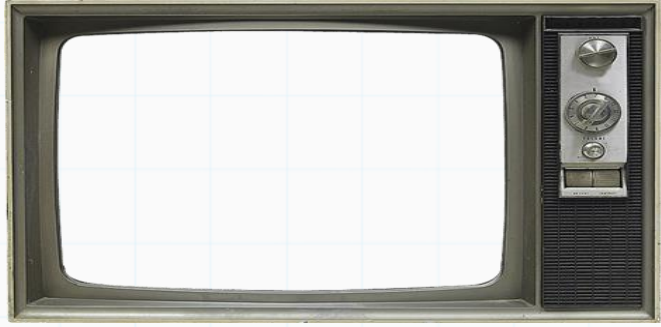
- Cópia de listas

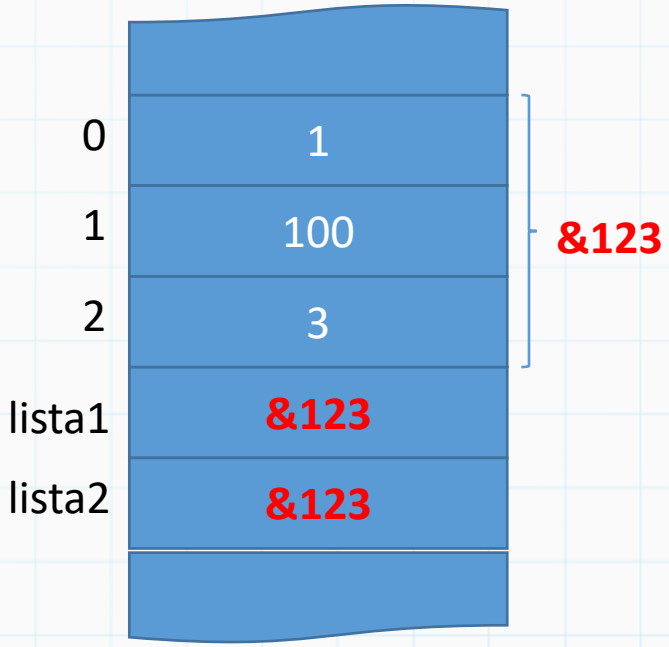
```
1  lista1    = [1,2,3]
2  lista2    = lista1
3  lista2[1] = 100
4  print(lista1)
```

Shell

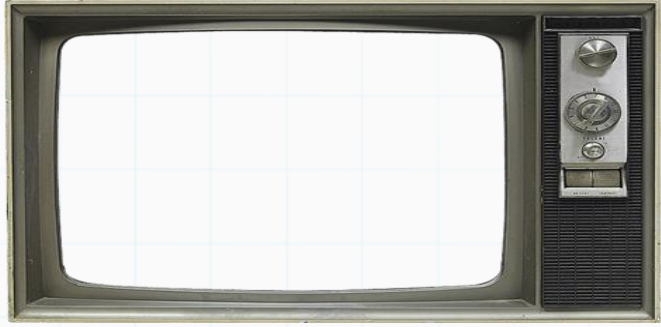
```
Python 3.7.7 (bundled)
>>> %Run teste.py

  [1, 100, 3]
```







Listas



- Como copiar listas então ?

```
1  lista1 = [1, 2, 3]
2  lista2 = []
3  for i in range(len(lista1)):
4      lista2.append(lista1[i])
5
6  lista2[1] = 100
7  print(lista1)
```

Shell

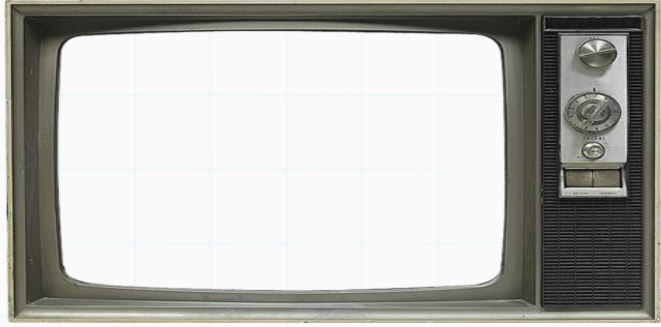
```
>>> %Run teste.py
 [1, 2, 3]
```

```
1  lista1 = [1, 2, 3]
2  lista2 = [0]*3
3  for i in range(len(lista1)):
4      lista2[i] = lista1[i]
5
6  lista2[1] = 100
7  print(lista1)
```

Shell

```
>>> %Run teste.py
 [1, 2, 3]
```

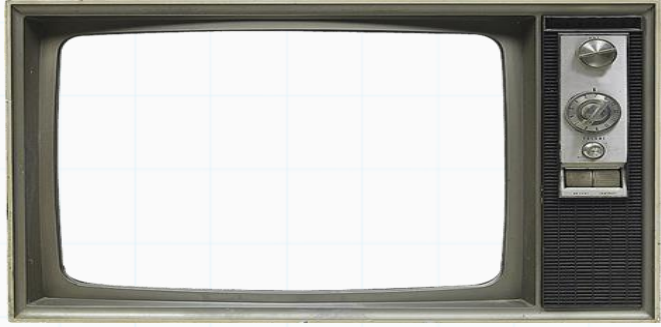
Listas



- Considerações:

Listas		
indexadas por posição		
mutáveis		
definidas com []		

Listas



- Considerações:

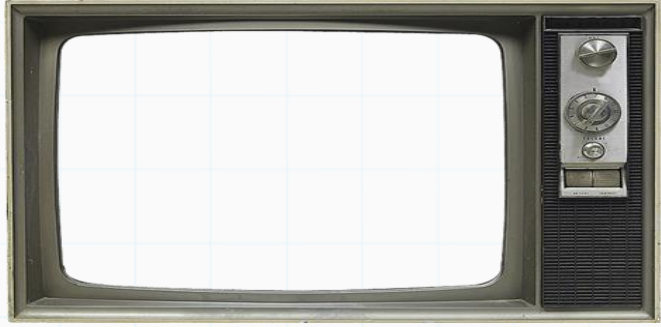
Listas	Tuplas	
indexadas por posição	indexadas por posição	
mutáveis	imutáveis	
definidas com []	definidas com ()	

```
1 tupla1 = (10, "Cebolinha", False)
2 for i in range(len(tupla1)):
3     print(tupla1[i])
```

Shell

```
>>>
>>> %Run teste.py
10
Cebolinha
False
```

Listas



- Considerações:

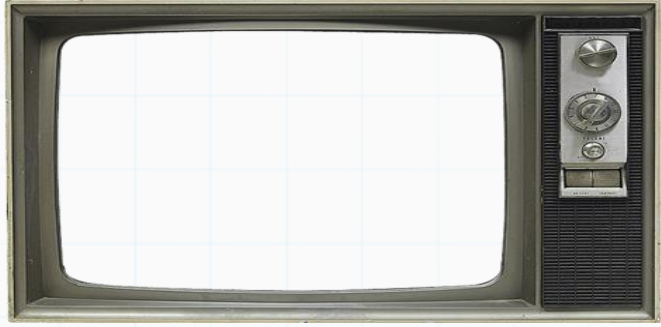
Listas	Tuplas	
indexadas por posição	indexadas por posição	
mutáveis	imutáveis	
definidas com []	definidas com ()	

```
1 tupla1 = (10, "Cebolinha", False)
2 tupla1[0] = "Monica"
3 print(tupla1)
```

Shell

```
>>> %Run teste.py
 Traceback (most recent call last):
   File "C:\Users\Yuri\Desktop\teste.py", line 2, in <module>
     tupla1[0] = "Monica"
 TypeError: 'tuple' object does not support item assignment
```

Listas



- Considerações:

Listas	Tuplas	Dicionários
indexadas por posição	indexadas por posição	indexadas por chaves
mutáveis	imutáveis	mutáveis
definidas com []	definidas com ()	Definidas com {}

```
1 treinamento = 10000
2 nivel = {"Goku": 229874, "Gohan": 215793, "Goten": 199745}
3 print(nivel["Goku"])
4 nivel["Gohan"] = nivel["Gohan"] + treinamento
5 print(nivel)
```

Shell

```
>>> %Run teste.py
 229874
 {'Goku': 229874, 'Gohan': 225793, 'Goten': 199745}
```

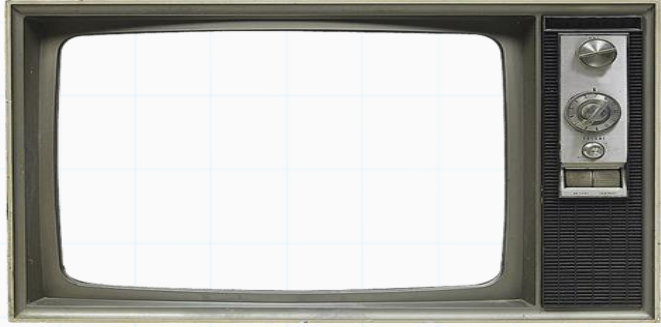
Listas

Fura Olho: O que será escrito ?

```
a=[0]*3
a[0]=1
for j in range(1,3):
    a[j]=3*a[j-1]
print(a)
```

```
# ENTRADA: 1, 7, 5, 5, 7, 1
x=[0]*3
y=[0]*3
z=[True]*3

for j in range(0,3):
    x[j] = int(input())
    y[j] = int(input())

for k in range(0,3):
    z[k]=(x[k]==y[k])

if (z[0] and z[1] and z[2]):
    print("datte")
else:
    print("bayo")
```

Listas

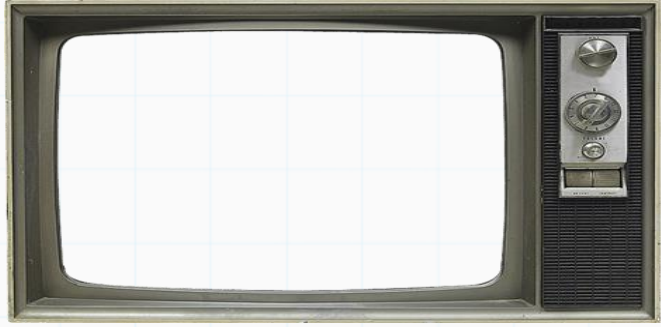


Fura Olho: O que será escrito ?

```
1  # ENTRADA: 10, 20, 30, 40
2  a=[]
3  b=[0]*4
4
5  for j in range(0,4):
6      a.append(int(input()))
7
8  for i in range(0,4):
9      b[i] = (a[4-(i+1)])**2
10
11 print(b)
```

```
1  x=[True]*5
2  for j in range(0,5):
3      k=(j+1)%5
4      x[k]=not x[j]
5  print(x)
```

Até a próxima



Slides baseados no curso de Vanessa Braganholo