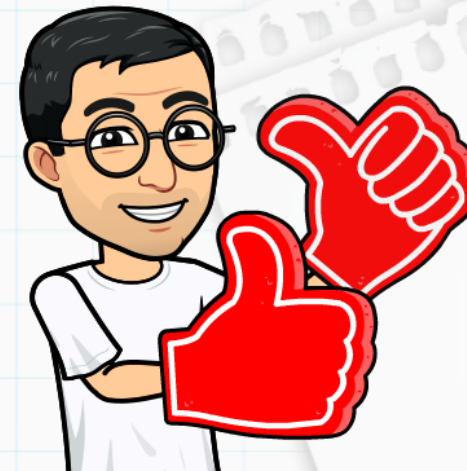
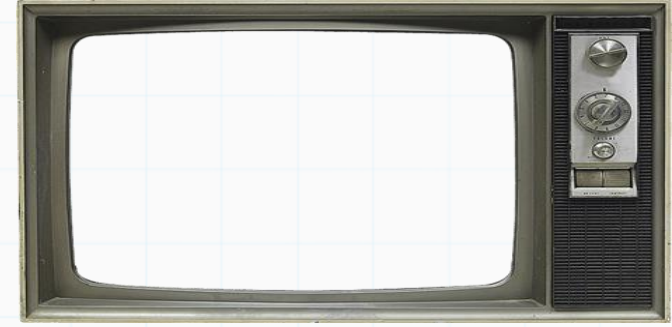


# Programação De Computadores

Professor : Yuri Frota

[www.ic.uff.br/~yuri/prog.html](http://www.ic.uff.br/~yuri/prog.html)

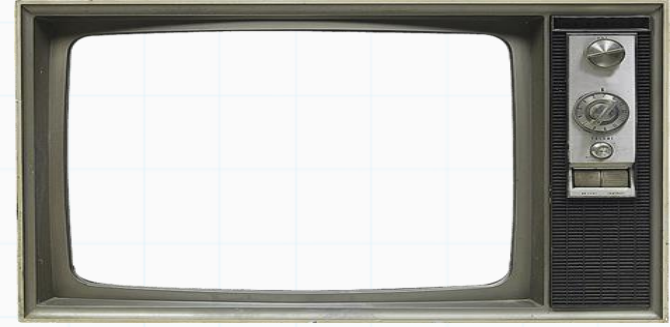
yuri@ic.uff.br



# Vetores

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

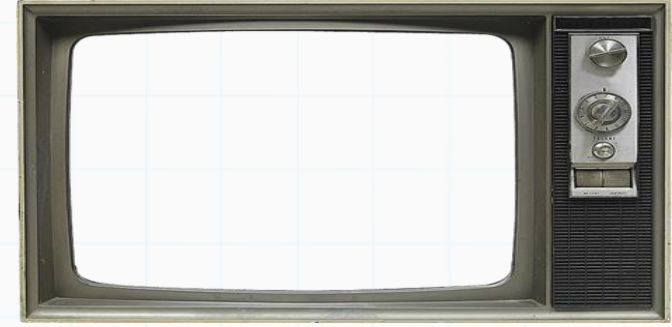
```
1 nota1 = float(input("Entre com a nota 1: "))
2 nota2 = float(input("Entre com a nota 2: "))
3 nota3 = float(input("Entre com a nota 3: "))
4 nota4 = float(input("Entre com a nota 4: "))
5 nota5 = float(input("Entre com a nota 5: "))
```



# Vetores

Mas e se fossem 15 alunos?

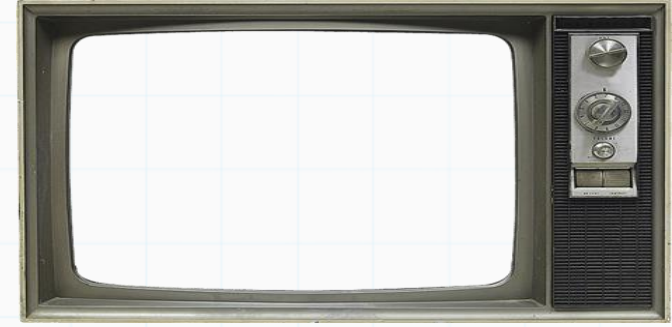
```
1 nota1 = float(input("Entre com a nota 1: "))
2 nota2 = float(input("Entre com a nota 2: "))
3 nota3 = float(input("Entre com a nota 3: "))
4 nota4 = float(input("Entre com a nota 4: "))
5 nota5 = float(input("Entre com a nota 5: "))
6 nota6 = float(input("Entre com a nota 6: "))
7 nota7 = float(input("Entre 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

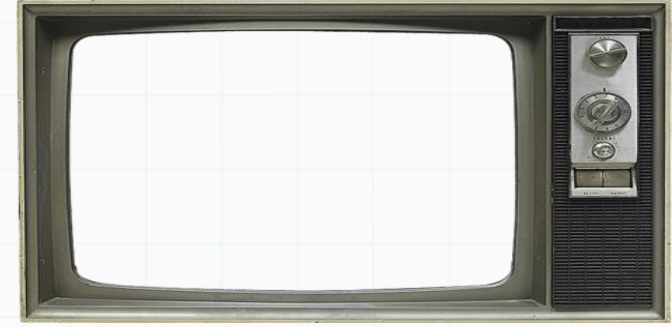
- É possível definir variáveis que guardam mais de um valor
- Essas variáveis são conhecidas como variáveis compostas, variáveis subscritas, variáveis indexáveis ou arranjos (array=vetores)



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)

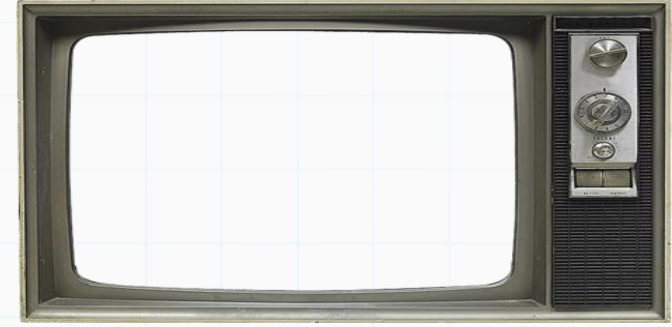
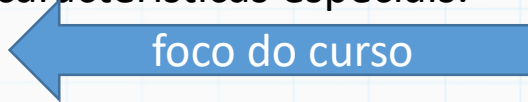


- Em Python existem três tipos principais de variáveis compostas (com a mesma lógica), cada uma com suas características especiais:

Listas

Tuplas

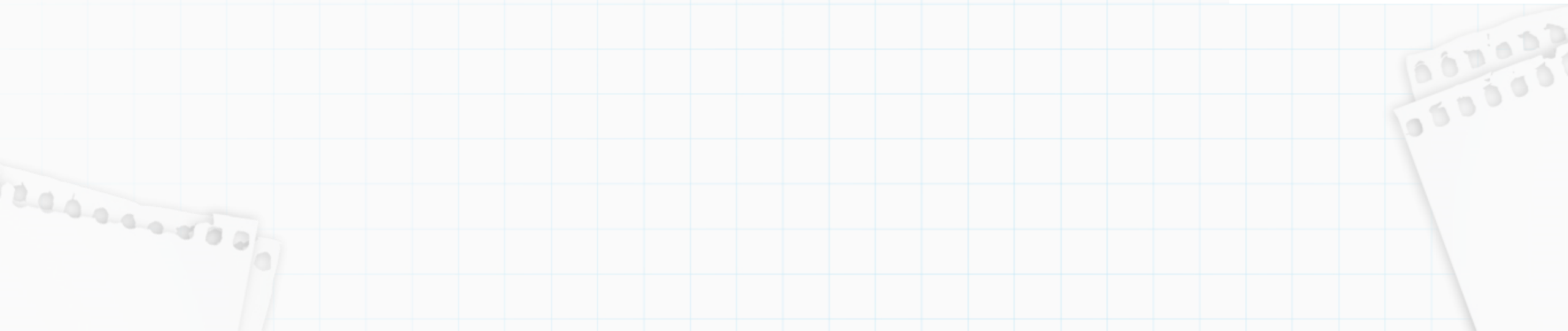
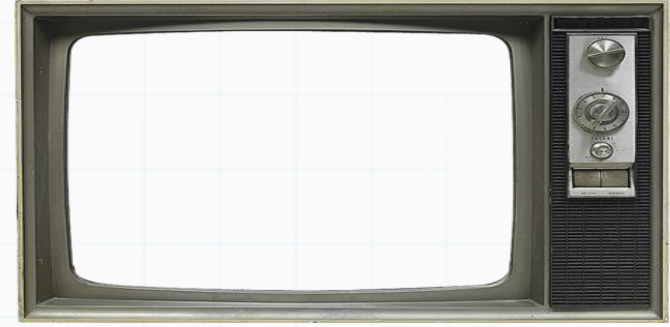
Dicionários



# 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]
```





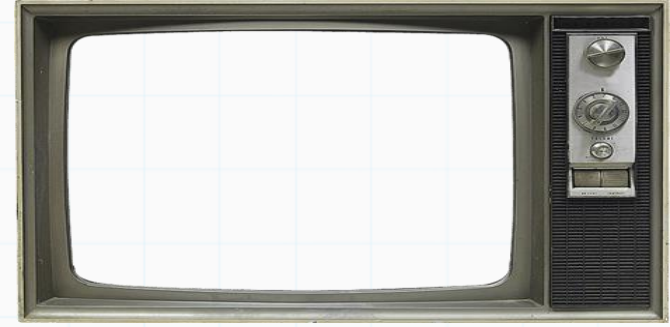
# Listas

- 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

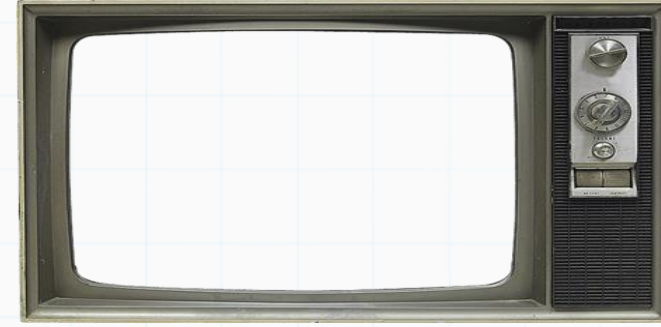
- Variável composta **unidimensional**, que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**
  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula
  - **É acessada via um índice inteiro**

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

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

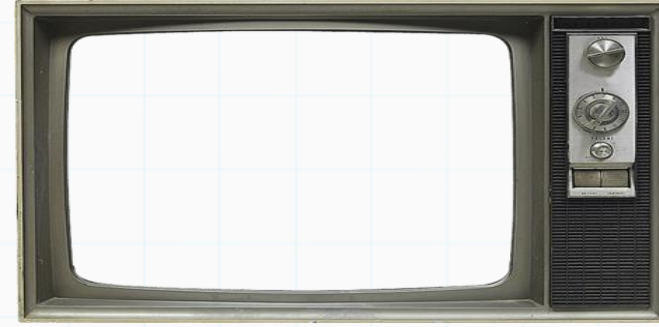
Shell ×

```
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

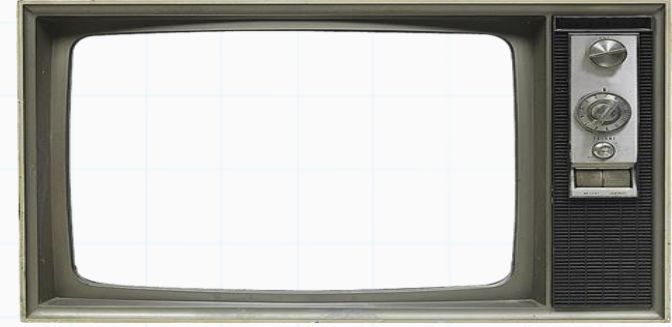
- Variável composta **unidimensional**, que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**
  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula
  - É acessada via um índice inteiro

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

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

Shell x

```
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

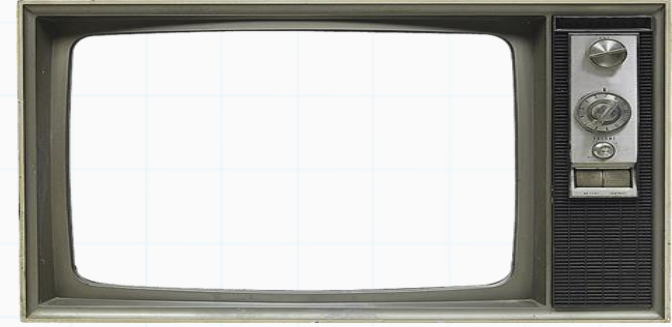
- Variável composta **unidimensional**, que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**
  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula
  - É acessada via um índice inteiro (índice inverso)

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

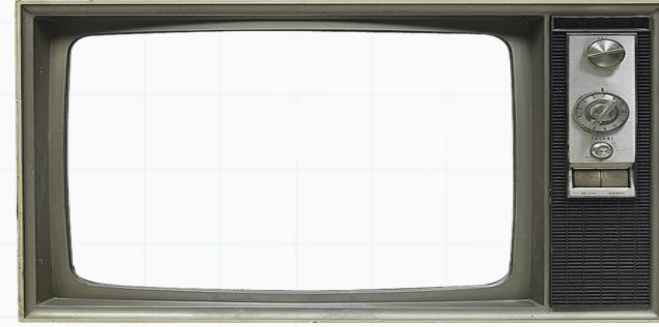
- Geralmente o acesso das listas é feita em laços para iterar todas as suas posições:

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

Shell x

python 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)

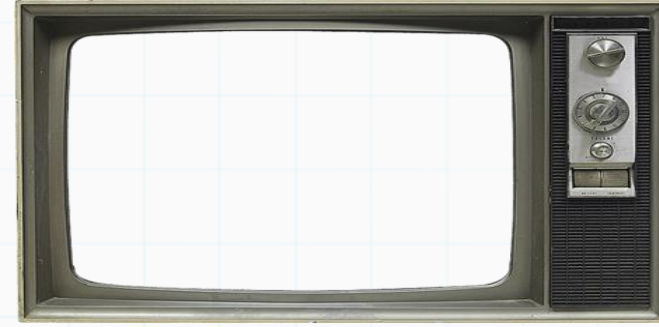
ind. inverso =	-5	-4	-3	-2	-1	
1	lista1 =	[10,	20,	30,	40,	50]
	ind =	0	1	2	3	4

- 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)

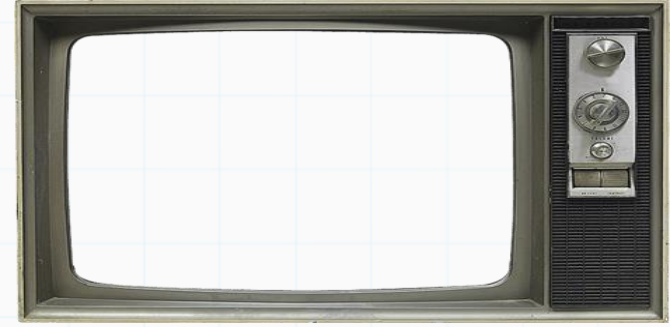
ind. inverso =	-5	-4	-3	-2	-1
1 lista1 =	[10,	20,	30,	40,	50]
ind =	0	1	2	3	4

- podemos alterar seus valores

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

Shell x

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





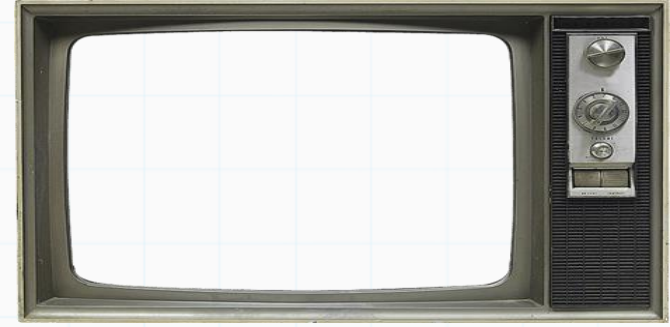
# Listas

- 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

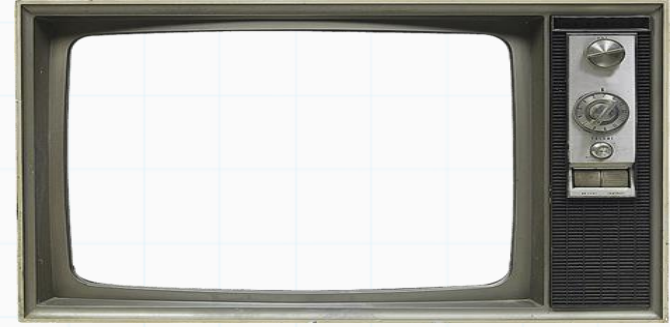
- Variável composta **unidimensional**, que armazena dados, em forma de sequência, e que **podem ser de um mesmo tipo ou não**
  - Contém espaço para armazenar diversos valores e é definida com colchetes [], separados os itens por vírgula
  - **É acessada via um índice inteiro** (índice inverso)

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)

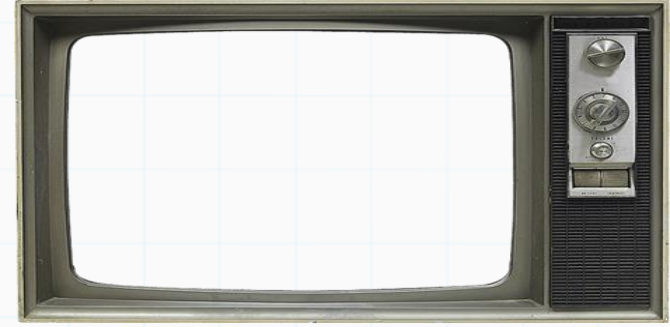
ind. inverso =	-5	-4	-3	-2	-1
1 lista1 =	[10,	20,	30,	40,	50]
ind =	0	1	2	3	4

- podemos alterar seus valores

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

Shell ×

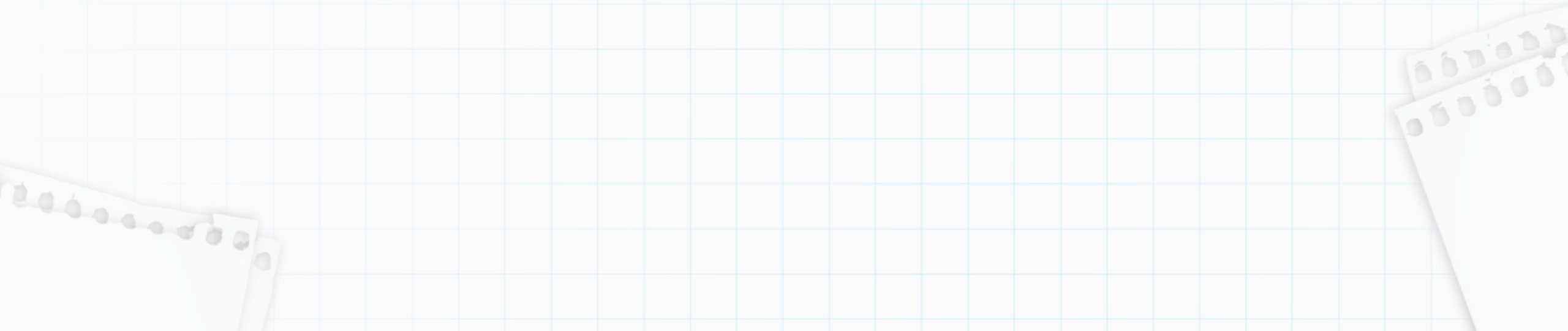
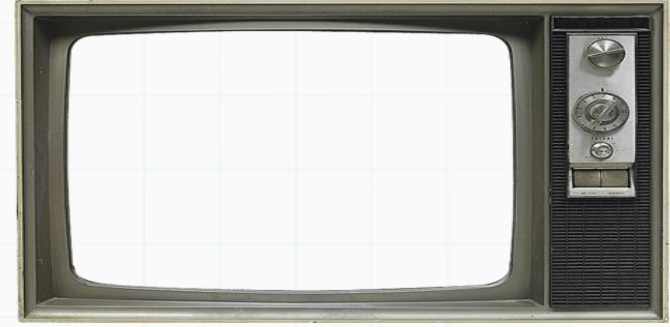
```
[30, 50, 70, 90, 50]
```



# 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]
```

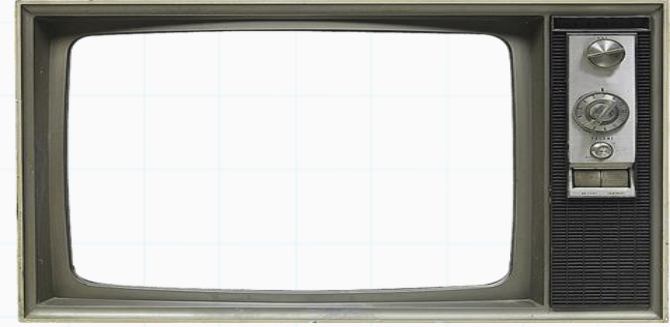
- direta com os mesmos valores (duplicados)

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

shell x

%%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 ×

~/teste.py

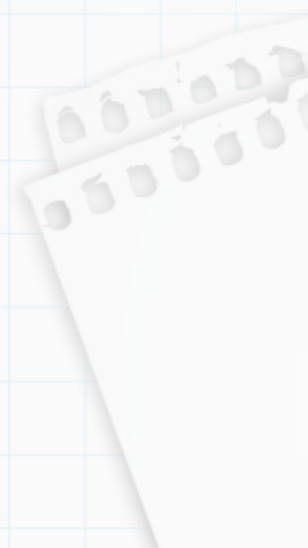
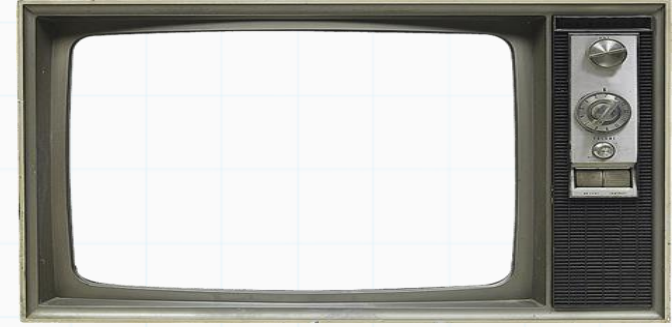
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- 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])
```

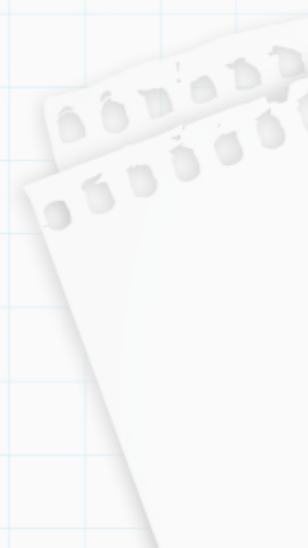
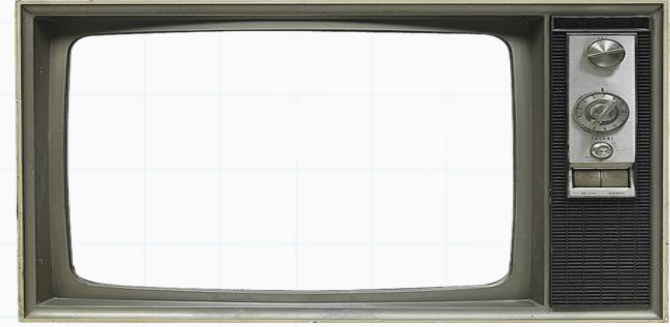
Shell ×

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

```
8.0
```

```
5.5
```

```
1.5
```





# Listas

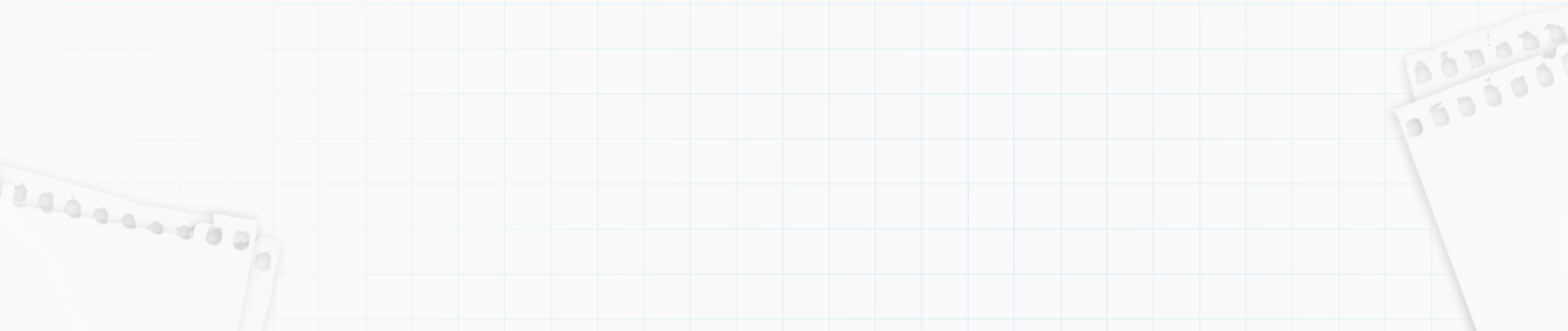
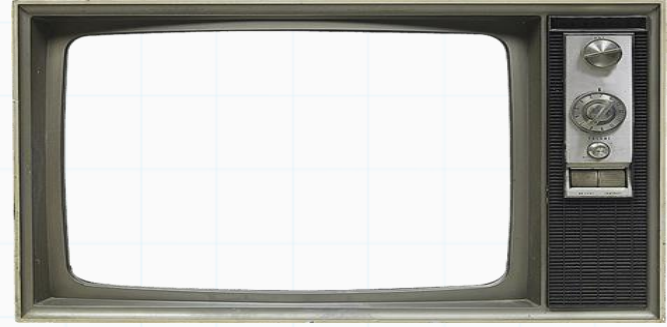
-Aumentando o tamanho da lista ( comando append) :

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

Shell ×

```
[9]
```

insere no fim da lista



# Listas

-Aumentando o tamanho da lista ( comando append ) :

insere no fim da lista

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

Shell ×

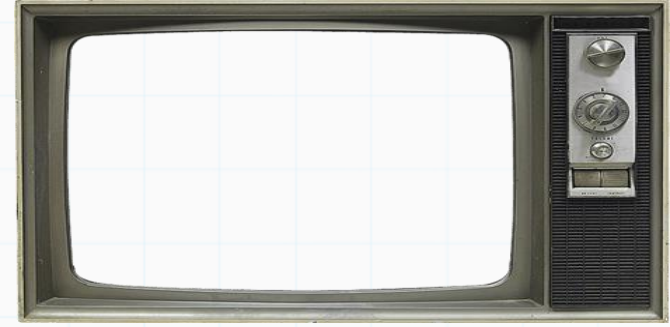
```
[9]
```

```
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 x

```
[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 x

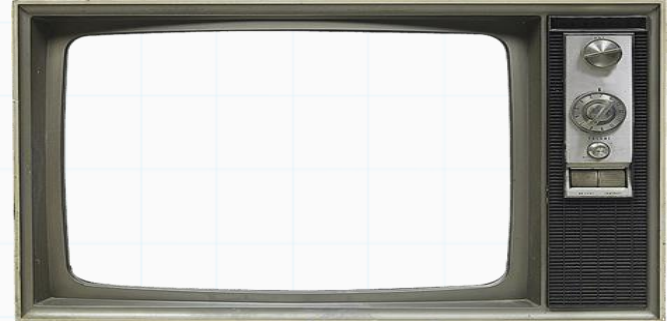
```
>>> %Run teste.py
```

```
9  
5.4  
7.8
```

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

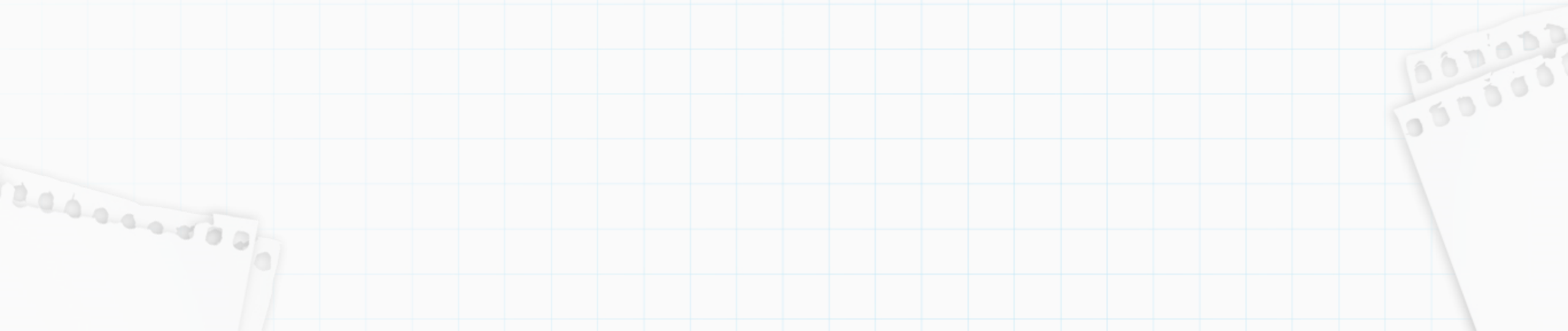
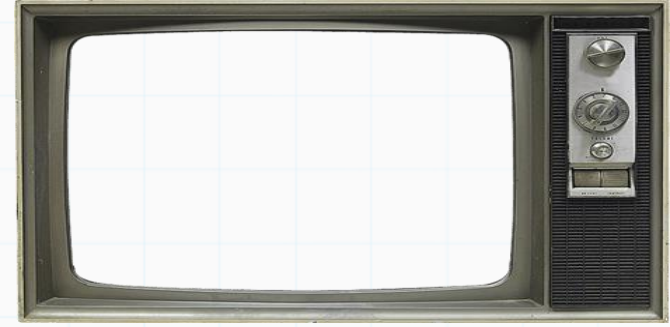
Shell x

```
>>> %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)
```



# Listas

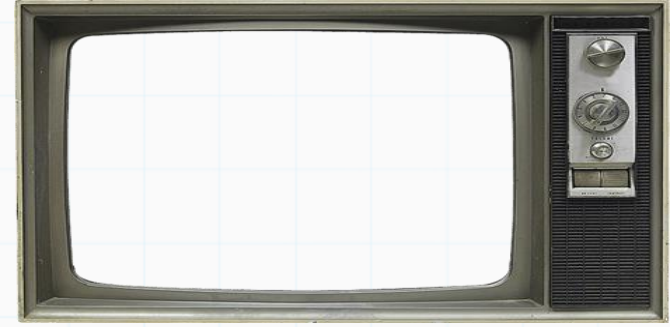
-Vamos fazer um programa para guardar os nomes e as notas de 40 alunos e dar os parabéns aqueles que tiraram nota acima da média:



# 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:

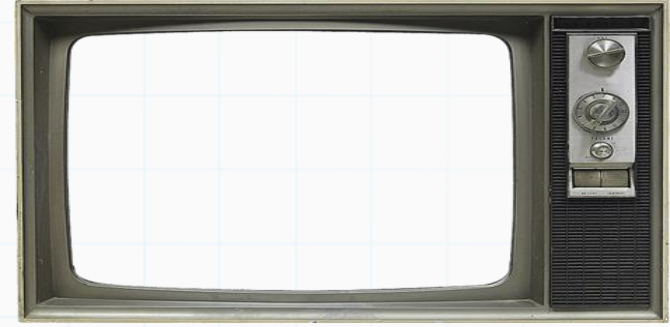
```
1 num = 40
2 nomes = []
3 notas = []
4 media = 0
5
6 for i in range(num):
7     nomes.append(input('nome: '))
8     notas.append(float(input('nota: ')))
9     media = media + notas[i]
10
11 media = media / num
12 print('A media da turma eh ', media)
13
14 for i in range(num):
15     if notas[i] > media:
16         print('Parabens', nomes[i])
17
```



# 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)
```

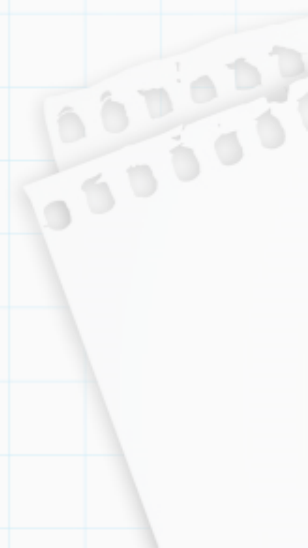
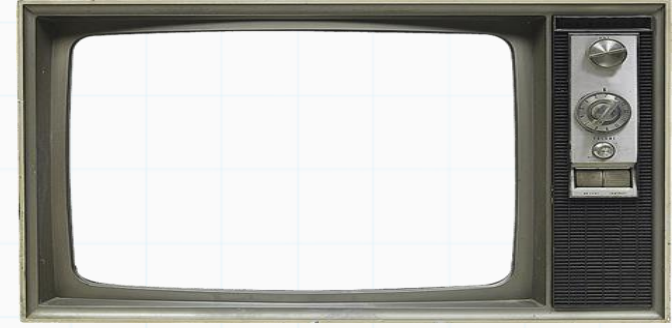
Shell ×

>>> %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 x

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 x

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

```
[1, 2, 3]
```

```
[1, 2, 3, 'Shun', False]
```

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

Shell x

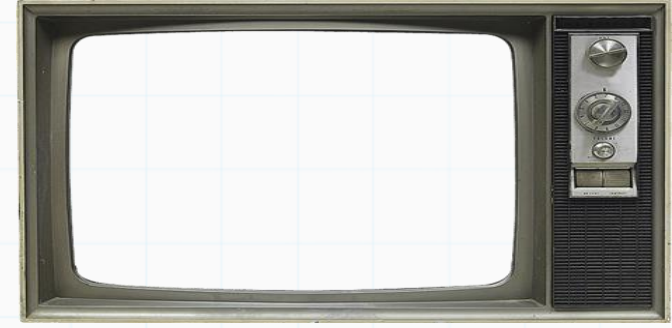
```
>>> %Run teste.py
```

```
['a', 'b', 'c']
```

```
['x', 'y', 'z', 'a', 'b', 'c']
```

coloca no fim da lista

ou no começo



# Listas

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

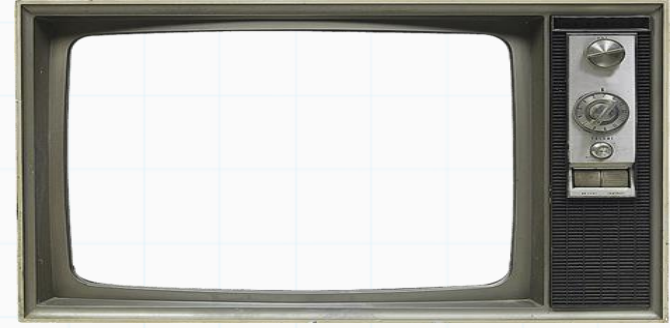
```
1 n = int(input('Digite um numero: '))
2 lista = []
3 for i in range(2,n+1,2):
4     lista = lista + [i]
5 print(lista)
```

Shell ×

/// %run teste.py

Digite um numero: 15

[2, 4, 6, 8, 10, 12, 14]



# 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
```

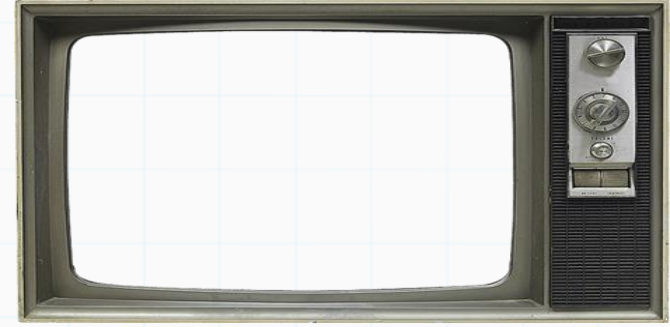
Shell x

Python 3.7.7 (bundled)

>>> %Run teste.py

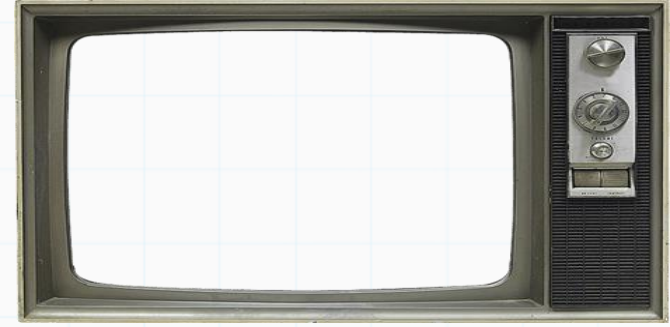
Digite um numero: 15

[14, 12, 10, 8, 6, 4, 2]



# 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 x

~/run teste.py

```
[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)
```

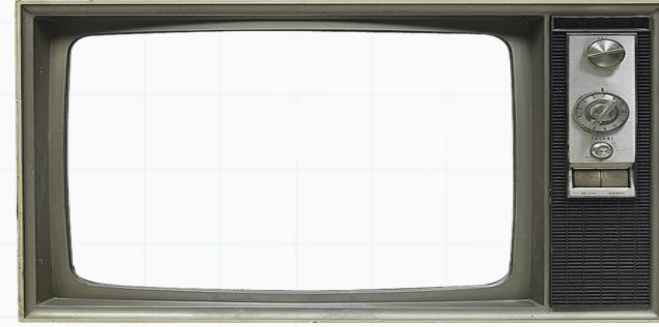
Shell x

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 x

```
%Run teste.py
```

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

# Listas

- Cópia de listas

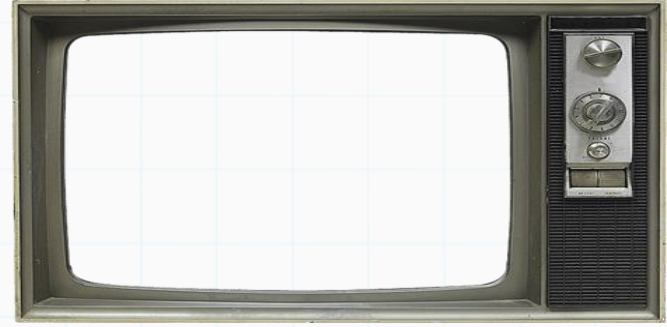
```
1 lista1 = [1,2,3]
2 lista2 = lista1
3 lista2[1] = 100
4 print(lista1)
```

Shell ×

Python 3.7.7 (bundled)

>>> %Run teste.py

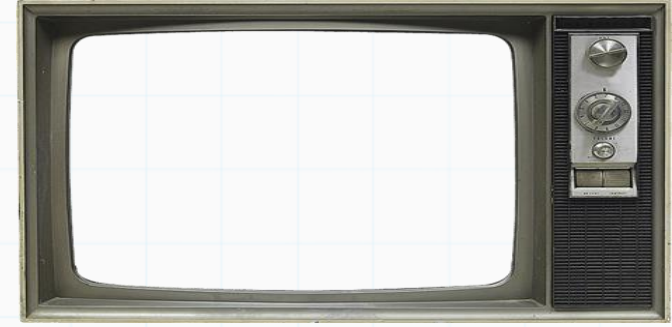
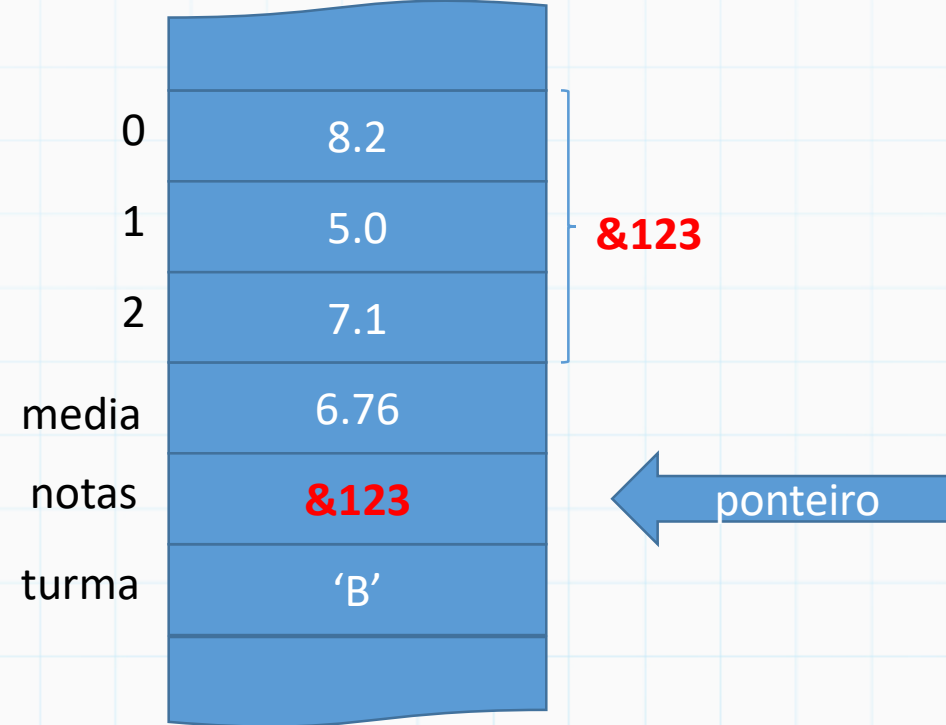
[1, 100, 3]



# 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

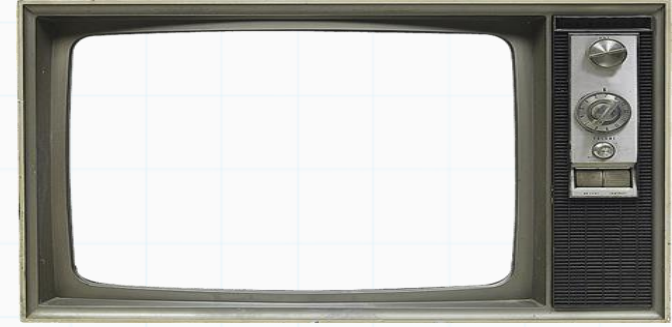
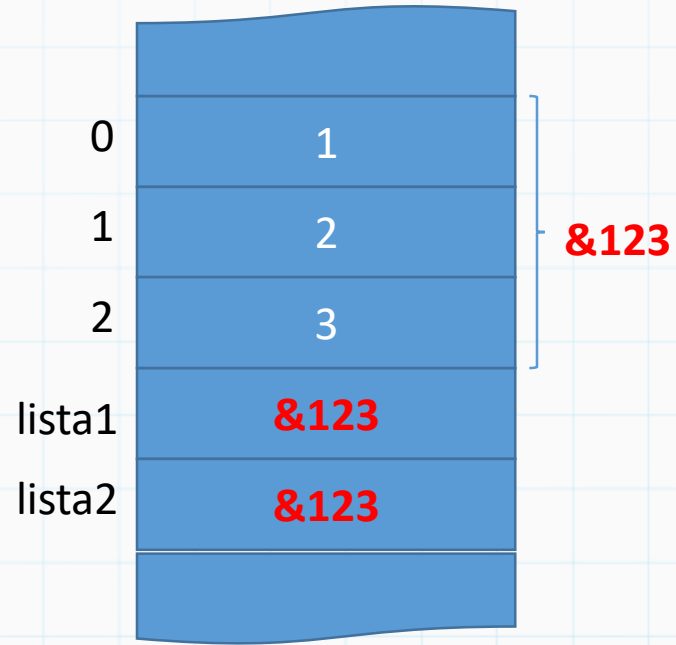
```
1 lista1 = [1,2,3]
2 lista2 = lista1
3 lista2[1] = 100
4 print(lista1)
```

Shell x

Python 3.7.7 (bundled)

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

```
[1, 100, 3]
```



# Listas

- Cópia de listas

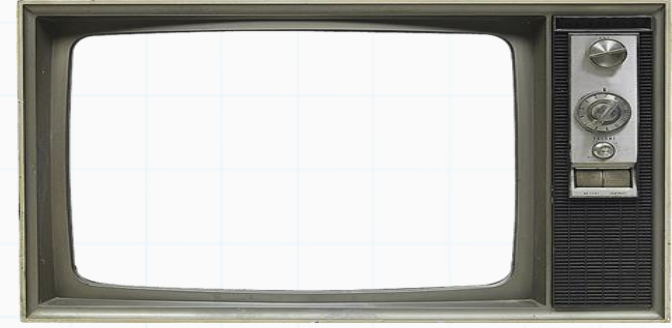
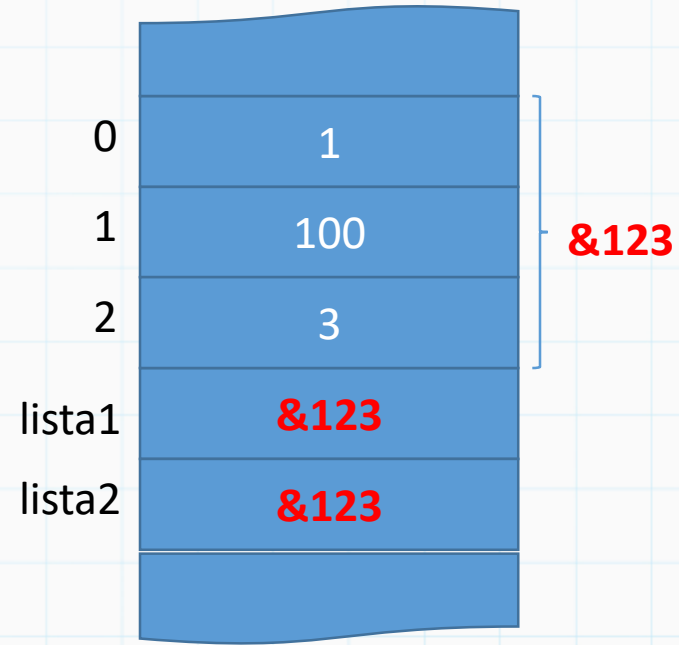
```
1 lista1 = [1,2,3]
2 lista2 = lista1
3 lista2[1] = 100
4 print(lista1)
```

Shell x

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

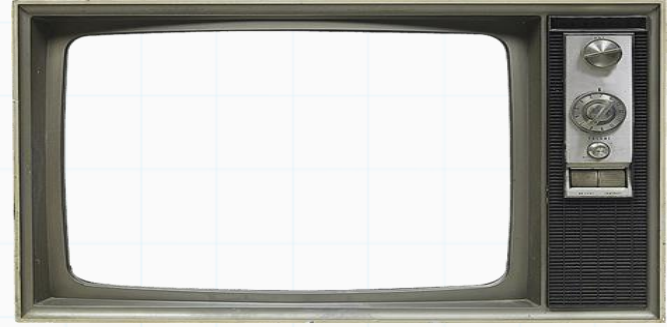
[1, 2, 3]

```
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 ×

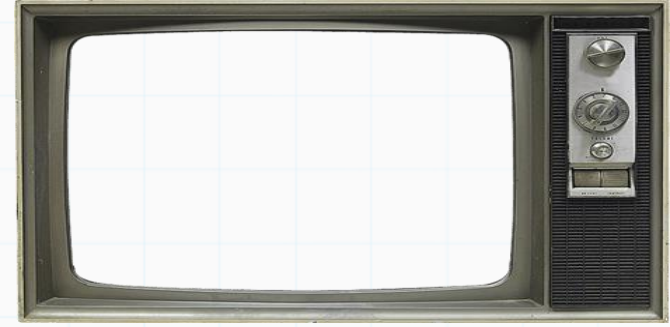
>>> %Run teste.py

[1, 2, 3]

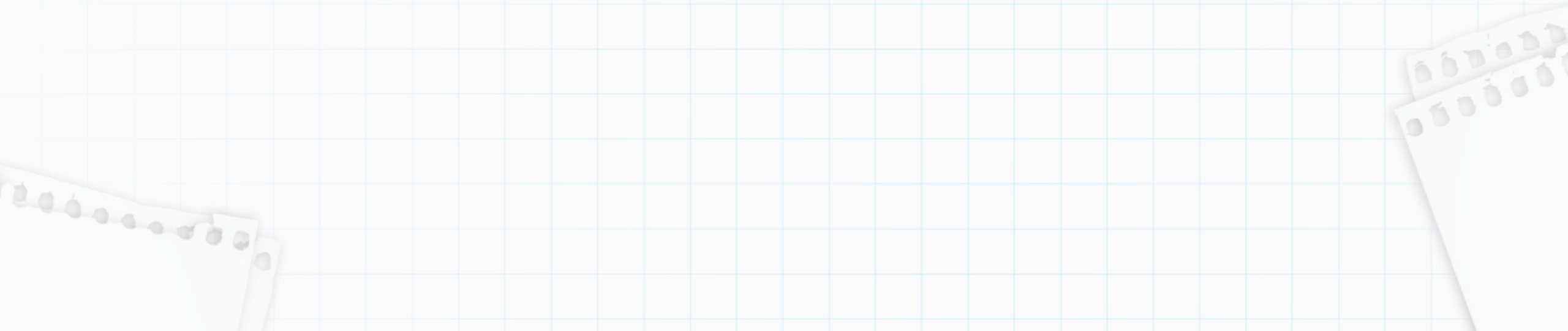


# Listas

- Considerações:

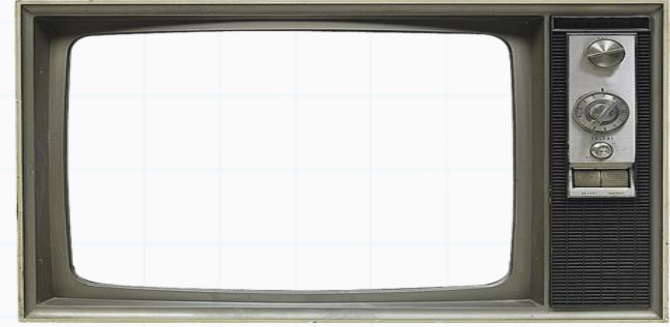


Listas		
indexadas por posição		
mutáveis		
definidas com []		



# 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 ×

///

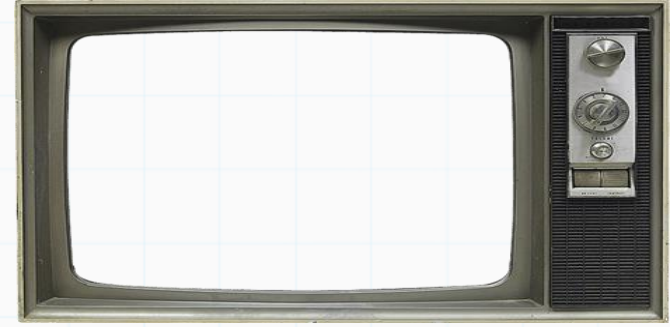
```
>>> %Run teste.py
```

```
10
Cebolinha
False
```



# Listas

- Considerações:

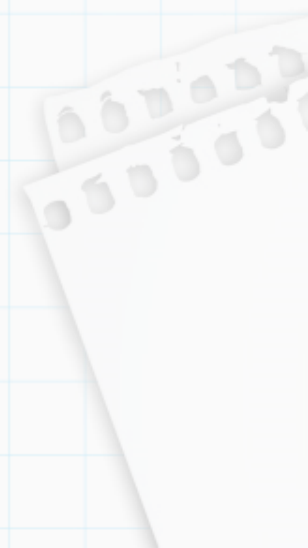


Listas	Tuplas	
indexadas por posição	indexadas por posição	
mutáveis	imutáveis	
definidas com []	definidas com ()	

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

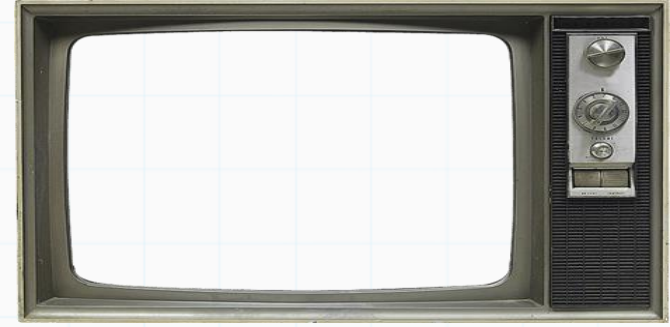
Shell ×

```
>>> %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}
```

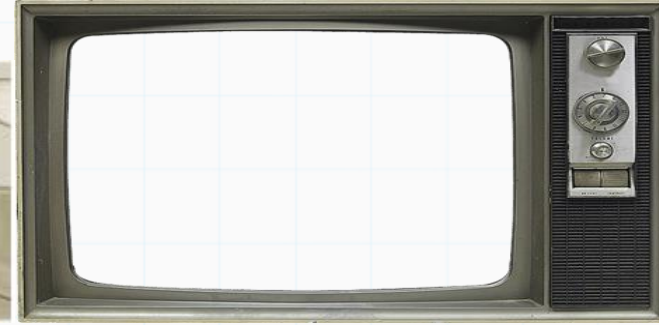


# Listas

Fura Olho: O que será escrito ?

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

```
1 # ENTRADA: 1, 7, 5, 5, 7, 1
2 x=[0]*3
3 y=[0]*3
4 z=[True]*3
5
6 for j in range(0,3):
7     x[j] = int(input())
8     y[j] = int(input())
9
10 for k in range(0,3):
11     z[k]=(x[k]==y[k])
12
13 if (z[0] and z[1] and z[2]):
14     print("datte")
15 else:
16     print("bayo")
```



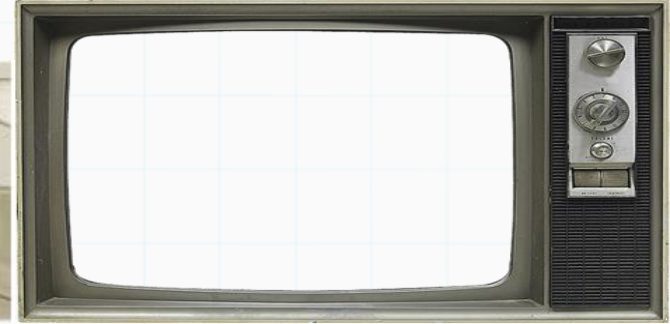
```
1 # ENTRADA: 10, 20, 30, 40
2 a=[]
3 b=[0]*4
4
5 for j in range(0,4):
6     a.append(int(input()))
7
8 for i in range(0,4):
9     b[i] = (a[4-(i+1)])**2
10
11 print(b)
```

```
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)
```





# Listas



Fura Olho: O que será escrito ?

[1,3,9]

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

```
1 # ENTRADA: 1, 7, 5, 5, 7, 1
2 x=[0]*3
3 y=[0]*3
4 z=[True]*3
5
6 for j in range(0,3):
7     x[j] = int(input())
8     y[j] = int(input())
9
10 for k in range(0,3):
11     z[k]=(x[k]==y[k])
12
13 if (z[0] and z[1] and z[2]):
14     print("datte")
15 else:
16     print("bayo")
```

```
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)
```

[1600,900,400,100]

```
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)
```

[False, False, True, False, True]

bayo

# Listas

Exercício 1): Receba do usuário um vetor de notas de 10 alunos e imprima a maior e a menor nota

Ex. de execução: `nota?10`

`nota?9`

`nota?7`

`nota?6`

`nota?5.5`

`nota?8`

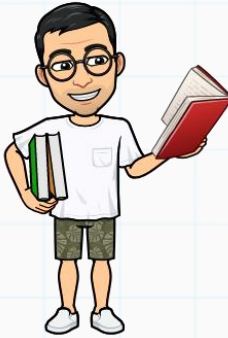
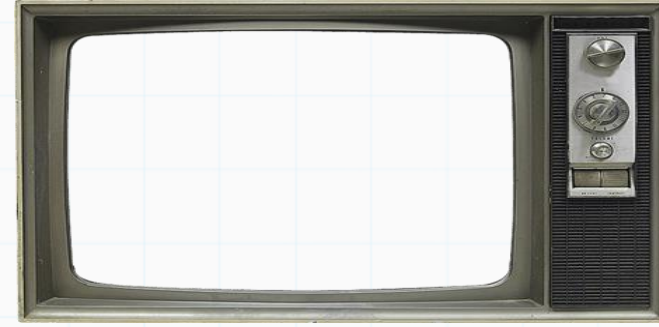
`nota?5`

`nota?4`

`nota?5`

`nota?6`

`maior= 10.0   menor= 4.0`



MAIOR  
nota = -1

MAIOR  
nota = 6

MAIOR  
nota = 7

MAIOR  
nota = 9.5

...

notas

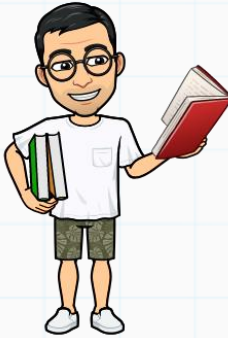
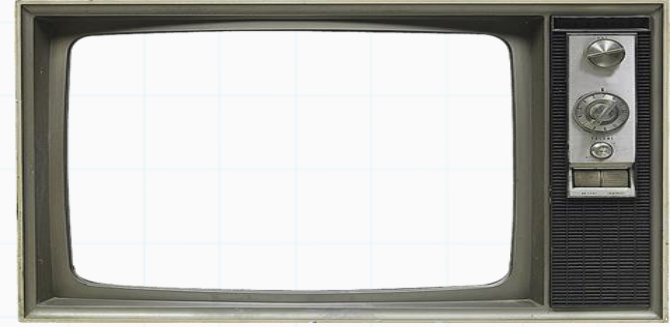


# Listas

Exercício 1): Receba do usuário um vetor de notas de 10 alunos e imprima a maior e a menor nota

```
1 n =[0.0]*10
2 ma=-1.0
3 me=11.0
4 for i in range(0,10):
5     n[i]=float(input("nota?"))
6     if (n[i]<me):
7         me=n[i]
8     if (n[i]>ma):
9         ma=n[i]
10 print("maior=",ma," menor=",me)
```

[código](#)



# Listas

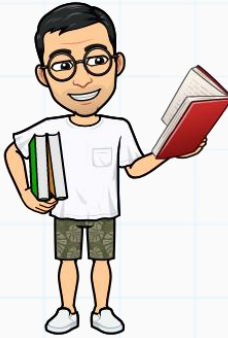
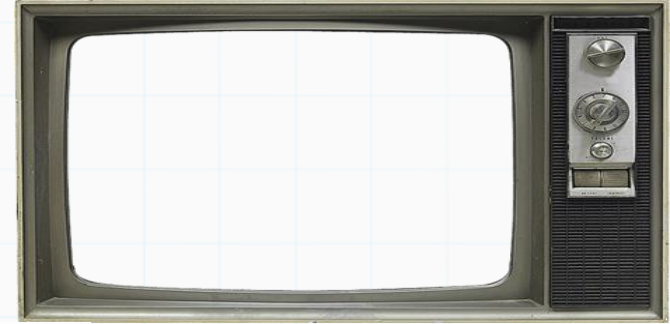
Exercício 1): Receba do usuário um vetor de **número (positivos e negativos)** de tamanho 10 e imprima o maior e o menor número

```
1 n =[0.0]*10
2 ma=-1.0
3 me=11.0
4 for i in range(0,10):
5     n[i]=float(input("nota?"))
6     if (n[i]<me):
7         me=n[i]
8     if (n[i]>ma):
9         ma=n[i]
10 print("maior=",ma," menor=",me)
```

[código](#)

```
1 n =[0.0]*10
2 ma=0
3 me=0
4 for i in range(0,10):
5     n[i]=float(input("nota?"))
6     if (i==0):
7         ma=n[i]
8         me=n[i]
9     if (n[i]<me):
10        me=n[i]
11    if (n[i]>ma):
12        ma=n[i]
13 print("maior=",ma," menor=",me)
```

[código](#)



# Listas

Exercício 2): A coordenação do curso de computação deseja saber quantos alunos estão cursando ( de forma irregular ) as disciplinas de PROG1 e PROG2 ao mesmo tempo. Faça um programa que:

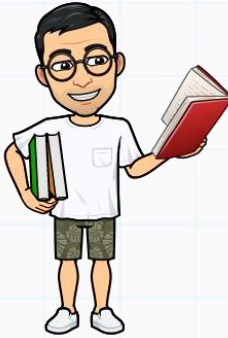
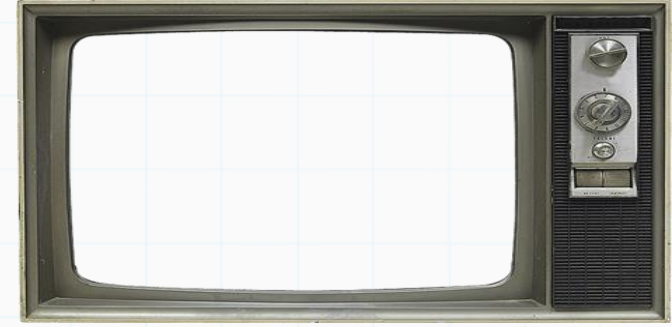
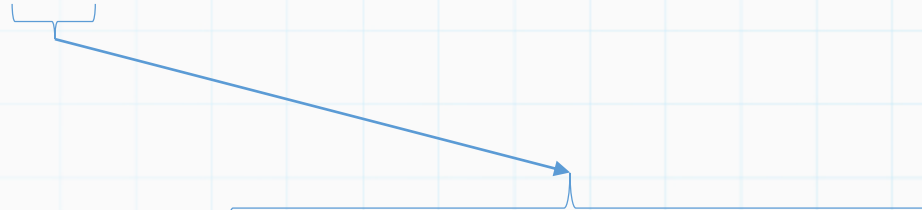
- (i) leia as matriculas (inteiro) dos 30 alunos de PROG1 e dos 40 alunos de PROG2,
- (ii) imprima as matrículas dos alunos irregulares e
- (iii) imprima no fim a quantidade de alunos irregulares

Prog1

23	47	12	8	7
----	----	----	---	---

Prog2

34	101	23	76	82
----	-----	----	----	----



# Listas

Exercício 2): A coordenação do curso de computação deseja saber quantos alunos estão cursando ( de forma irregular ) as disciplinas de PROG1 e PROG2 ao mesmo tempo. Faça um programa que:

- (i) leia as matriculas (inteiro) dos 30 alunos de PROG1 e dos 40 alunos de PROG2,
- (ii) imprima as matrículas dos alunos irregulares e
- (iii) imprima no fim a quantidade de alunos irregulares

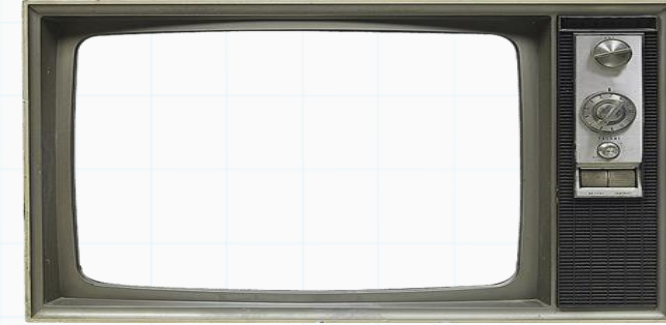
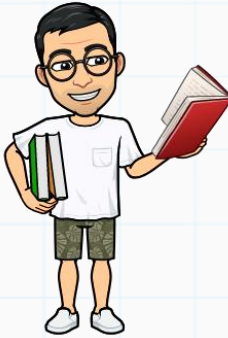
Prog1

23	47	12	8	7
----	----	----	---	---

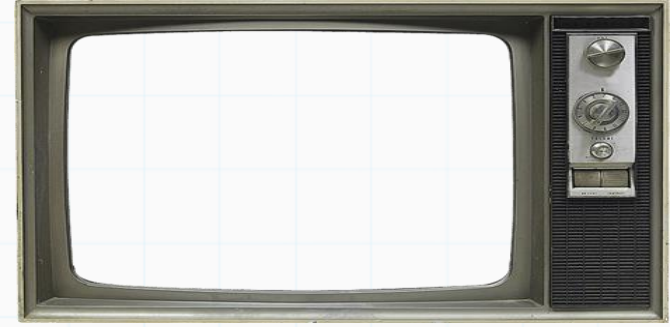
Prog2

34	101	23	76	82
----	-----	----	----	----

laço duplo



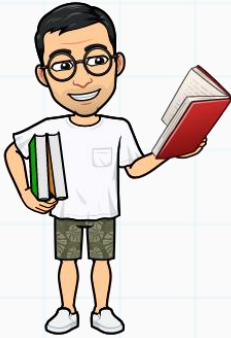
# Listas



Exercício 2): A coordenação do curso de computação deseja saber quantos alunos estão cursando ( de forma irregular ) as disciplinas de PROG1 e PROG2 ao mesmo tempo. Faça um programa que:

- (i) leia as matriculas (inteiro) dos 30 alunos de PROG1 e dos 40 alunos de PROG2,
- (ii) imprima as matrículas dos alunos irregulares e
- (iii) imprima no fim a quantidade de alunos irregulares

```
1 np1=30
2 np2=40
3 cont=0
4 p1=[0]*np1
5 p2=[0]*np2
6 print('Prog1')
7 for i in range(np1):
8     p1[i]=(int(input(str(i)+" ")))
9 print('Prog2')
10 for i in range(np2):
11     p2[i]=(int(input(str(i)+" ")))
12
13 for i in range(np1):
14     for j in range(np2):
15         if (p1[i]==p2[j]):
16             print("Aluno ",p1[i]," irregular")
17             cont=cont+1
18 print("total =",cont)
```



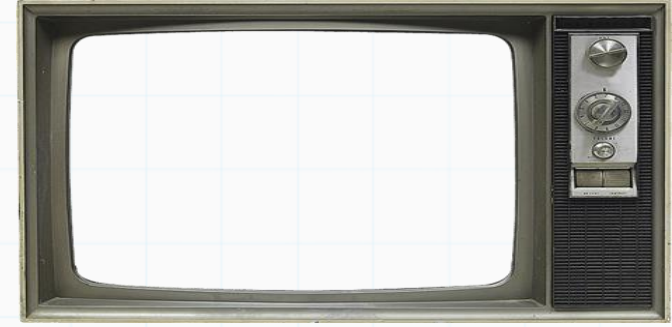
```
Prog1
0) 23
1) 47
2) 12
3) 8
4) 7
```

str() -> converte para texto

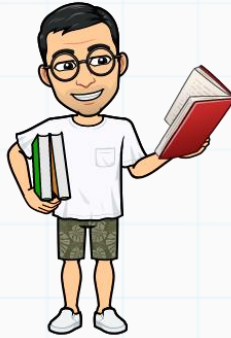
posso concatenar (+)  
textos também



# Listas



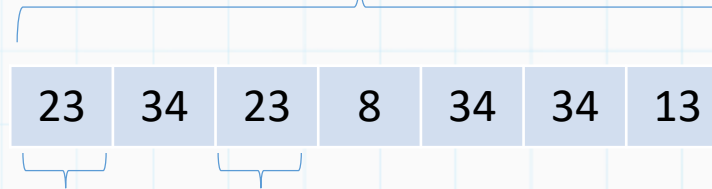
Exercício 3): Escreva um programa que lê um vetor de 10 números inteiros e imprima o número que aparece mais vezes (se houver empate, pode ser qualquer um). Além disso, imprima quantas vezes ele aparece.



vet



vet



MELHOR

Elem = 23

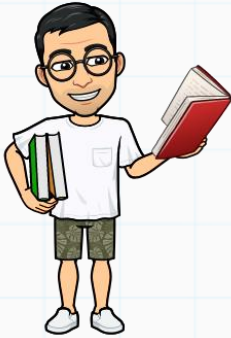
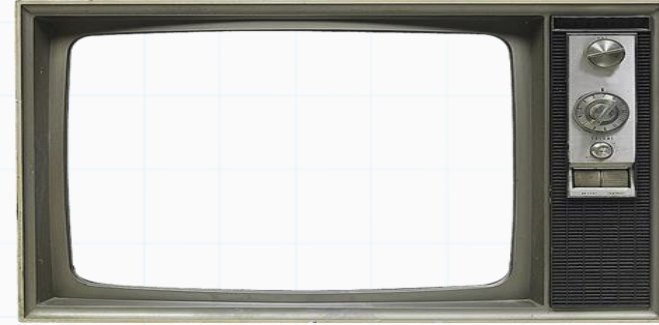
cont = 2

Dica: Essa questão pode ser feita com laço duplo, mas os dois laços percorrerão o mesmo vetor. O primeiro laço irá percorrer cada posição do vetor. O segundo laço percorrerá o vetor novamente contando quantas vezes o elemento do primeiro laço aparece no vetor.



# Listas

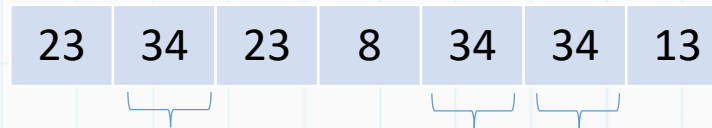
Exercício 3): Escreva um programa que lê um vetor de 10 números inteiros e imprima o número que aparece mais vezes (se houver empate, pode ser qualquer um). Além disso, imprima quantas vezes ele aparece.



vet



vet



MELHOR

Elem = 23

cont = 2

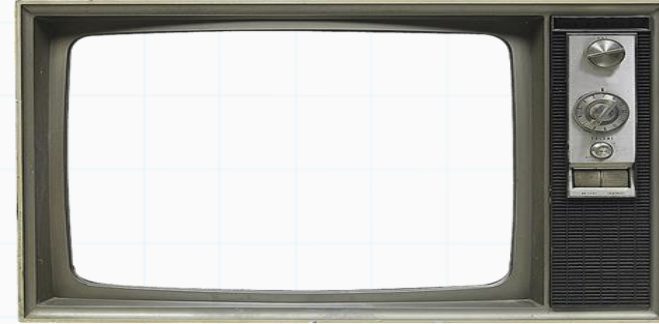
MELHOR

Elem = 34

cont = 3

Dica: Essa questão pode ser feita com laço duplo, mas os dois laços percorrerão o mesmo vetor. O primeiro laço irá percorrer cada posição do vetor. O segundo laço percorrerá o vetor novamente contando quantas vezes o elemento do primeiro laço aparece no vetor.

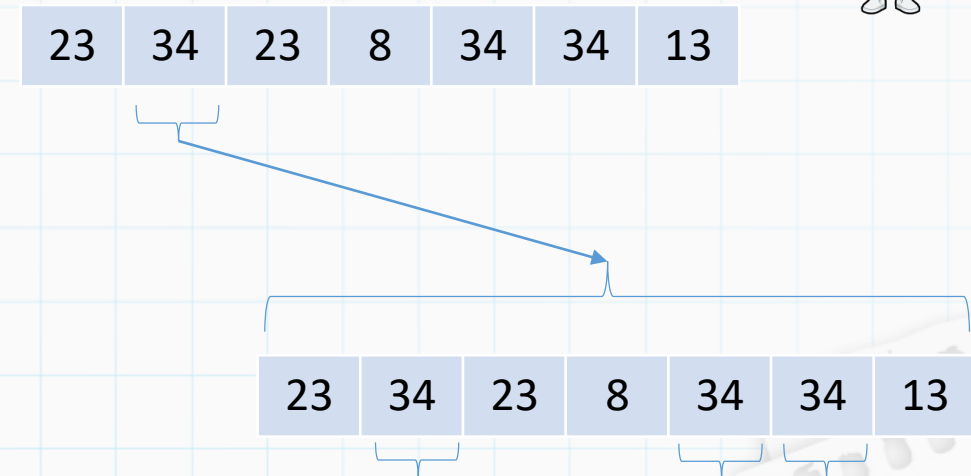
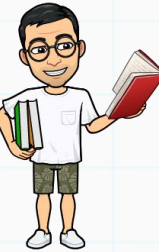
# Listas



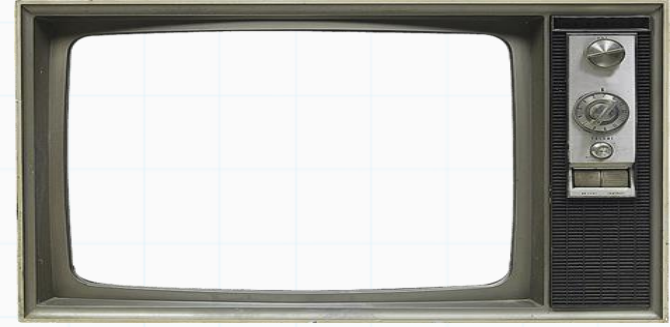
Exercício 3): Escreva um programa que lê um vetor de 10 números inteiros e imprima o número que aparece mais vezes (se houver empate, pode ser qualquer um). Além disso, imprima quantas vezes ele aparece.

```
1  n      =10
2  v      =[0]*n
3  max_cont =0
4  elem    =0
5
6  for i in range(n):
7      v[i]=(int(input(str(i)+" ")))
8
9  for i in range(n):
10     cont =0
11     for j in range(n):
12         if (v[i]==v[j]):
13             cont=cont+1
14
15     if (max_cont<cont):
16         elem    =v[i]
17         max_cont =cont
18
19  print("número ",elem," e ",max_cont," vezes")
```

[código](#)



Até a próxima



Slides baseados no curso de Vanessa Braganholo

