

# DESTRAVE O CONHECIMENTO

 /mesttra

*Nasser Boan*





# *Fundamentos Python*

# Variáveis

- A variável é um espaço de memória reservado do computador para armazenar um determinado dado;
- Ela deve receber um identificador para que possa ser referenciada;
- Uma variável pode assumir vários valores diferentes, mas somente armazena um por vez.



# Variáveis

Operador	Ação
=	Atribuição

**altura = 171**

**b = a = 171**


# *Variáveis*

Para transmitir informações utilizamos formatos de dados padronizados.

Usamos letras organizadas para escrever um texto, números para determinar quantidade, frações para demonstrar partes.

Dentro do python também existem nativamente formas diferentes de representar informação.

# *Variáveis: Palavras reservadas*

and	except	lambda	with
as	finally	nonlocal	while
assert	false	None	yield
break	for	not	
class	from	or	
continue	global	pass	
def	if	raise	
del	import	return	
elif	in	True	
else	is	try	

# Variáveis

```
[56] 1 x = 'Nasser'  
      2  
      3 type(x)
```

str

## string

Dados textuais em python  
são representados por  
strings.

```
[57] 1 x = 10  
      2  
      3 type(x)
```

int

## integer

Valores numéricos  
INTEIROS são  
representados por  
integers (int).

```
[58] 1 x = 3.3  
      2  
      3 type(x)
```

float

## float

Valores flutuantes são  
aqueles que descrevem  
frações de um inteiro.

```
[14] 1 x = True  
      2  
      3 type(x)
```

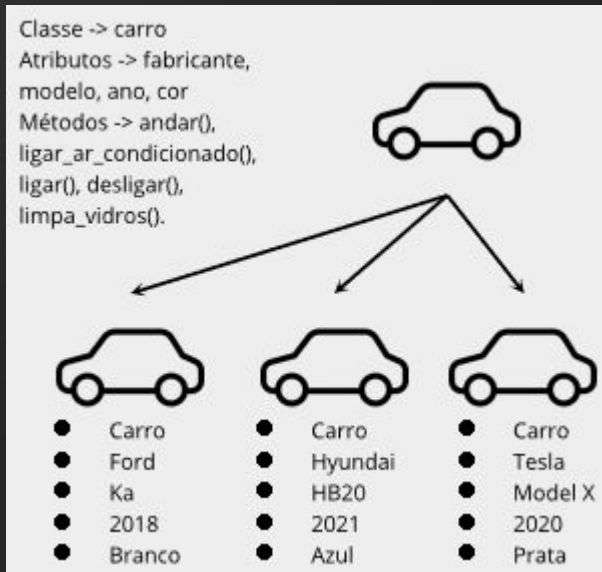
bool

## boolean

Valores booleanos são  
utilizados para  
representar uma  
expressão como  
verdadeira ou falsa

# Métodos

Python é uma linguagem orientada a objetos. Objetos são uma coleção de informações (dados) e funções (métodos) que atuam nesses dados. As instruções para criação de um objeto estão guardadas dentro de uma classe.

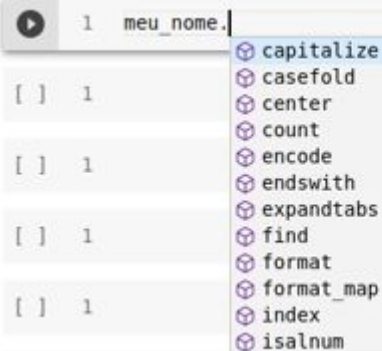




# Métodos

Os objetos python também possuem métodos. Para acessar os métodos primeiro precisamos instanciar o objeto dentro de uma variável. Depois basta colocar um ponto final depois da variável e escrever o método desejado.

```
[6] 1 meu_nome = 'nasser'
```



A screenshot of a Jupyter Notebook interface. The top cell shows the assignment `meu_nome = 'nasser'`. The bottom cell shows `meu_nome.` followed by a dropdown menu of string methods. The methods listed are: `capitalize`, `casefold`, `center`, `count`, `encode`, `endswith`, `expandtabs`, `find`, `format`, `format_map`, `index`, and `isalnum`. The `capitalize` method is currently selected and highlighted in blue.

```
[8] 1 meu_nome.upper()
```

```
'NASSER'
```

# Métodos: String

Método	Resultado
.upper()	Converte todas as letras minúsculas da string em maiúsculas.
.lower()	Converte todas as maiúsculas em minúsculas.
.capitalize()	Converte a primeira letra da string para maiúscula.
.startswith('texto')	Retorna True se a string começa com 'texto'
.endswith('texto')	Retorna True se a string termina com 'texto'
.strip('?')	Remove '?' no início e fim da string
.find('texto')	Retorna o índice de início de 'texto'
.split()	Quebra a string e retorna uma lista

# Métodos: String

Crie uma string com seu nome completo totalmente minúsculo e guarde na variável “nome”;

Altere essa string modificando a primeira letra do primeiro nome para uma letra maiúscula (utilize um método);

Use um método para testar se a primeira letra do seu nome completo é “B”;

Use um método para testar se a última letra do seu nome completo é “A”;

Use um método para encontrar a primeira letra “A” do seu nome completo;

Utilize um método para retirar a última letra do seu último nome;

Utilize um método para separar por espaços em branco todo o seu nome.

# Operadores Aritméticos

Operador	Ação
-	subtração
+	adição
*	multiplicação
%	módulo da divisão (resto)
**	potenciação
+=	incremento (soma)
*=	incremento (multiplicação)
-=	decremento (subtração)

# Operadores Aritméticos

- Criar a variável “altura” (em M)
- Criar a variável “peso” (em kg)
- Criar a variável “IMC” com o resultado da seguinte fórmula:

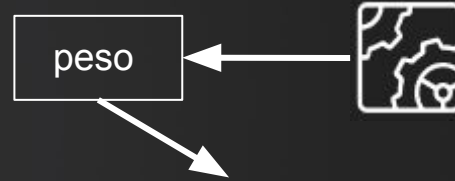
$$\text{IMC} = \frac{\text{Peso}}{\text{Altura} \times \text{Altura}}$$

# Operadores Aritméticos

- Crie uma variável “x” com o valor igual a 4;
- Criar um algoritmo para salvar o resultado da seguinte fórmula na variável “y”.

$$2x^2 - \frac{3x^{(x+1)}}{2} + \frac{\sqrt{x+1}}{x}$$

## Entrada e saída



```
print(f'Seu peso é: {peso} kilos.')
```

```
print(f'Sua altura é: {altura} metros.')
```

```
print(f'O IMC é: {IMC}.')
```

```
print(f'P: {peso}, A: {altura}, IMC:  
{IMC}.')
```

# *Entrada e saída*

- O input serve para coletar dados do usuário.
- O resultado SEMPRE será uma string (texto).

```
input("Qual é o seu nome?")
```



## *Entrada e saída*

Crie um algoritmo que pergunte a idade do usuário, transforme a idade em float, calcule a idade em meses, calcule a idade em dias e mostre o resultado.

Considere que todos os anos possuem 365 dias exatos.

Obs: Use a função `float()` para transformar o valor da variável.

# Operadores Lógicos

Operador	Ação
not	NÃO (negação)
and	E (conjunção)
or	OU (disjunção inclusiva)
>	maior que
>=	maior ou igual
<	menor que
<=	menor ou igual
==	igual
!=	diferente

!! “not” tem a maior precedência !!

# *Operadores Lógicos*

NOT	
A	X
0	1
1	0

OR		
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

AND		
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

# *Estrutura de Dados*

# *Estrutura de dados*

Organizar, gerenciar e armazenar dados são importantes, pois facilitam o acesso aos dados e criam a possibilidade de modificações eficientes.

As estruturas de dados permitem organizar seus dados de tal forma que você possa armazenar coleções de dados, relacioná-los e realizar operações sobre eles.

```
[1] 1 lista = [1,2,3,4,5]
```

```
[2] 1 tupla = (1,2,3,4,5)
```

```
[3] 1 dicionario = {'a':1,'b':2,'c':3,'d':4,'e':5}
```

# Estrutura de dados



## Estrutura de Dados: Lista

A “lista” é uma estrutura de dados para guardar vários dados numa só variável.

A lista por si só são ordenáveis e mutáveis, elas também podem conter valores repetidos e outras estruturas de dados dentro de si mesmas.

```
[67] 1 cidades_que_visitei = []
```

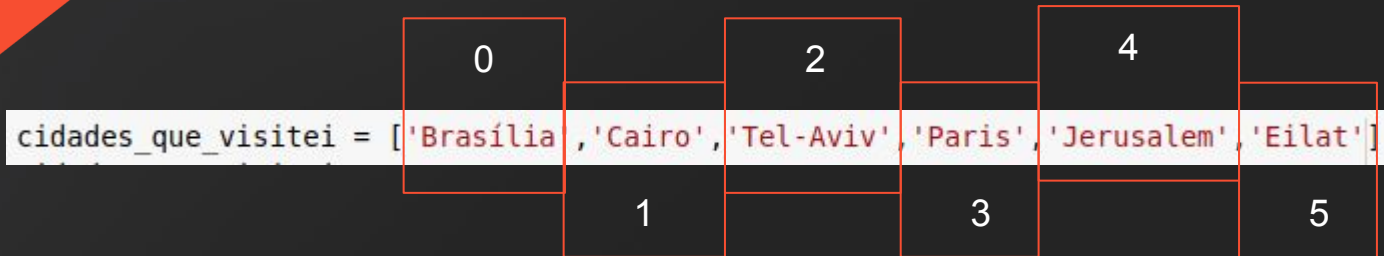
```
[68] 1 cidades_que_visitei  
      []
```

```
[70] 1 cidades_que_visitei = ['Brasília', 'Cairo', 'Tel-Aviv', 'Paris', 'Jerusalem', 'Eilat']  
     2 cidades_que_visitei  
      ['Brasília', 'Cairo', 'Tel-Aviv', 'Paris', 'Jerusalem', 'Eilat']
```

# Estrutura de Dados: Lista

Em programação, começamos a contar do valor 0.

Ou seja, o primeiro valor da lista tem o índice 0, o segundo tem o índice 1 e assim por diante.





## *Estrutura de Dados: Lista*

Podemos indexar as listas utilizando também os colchetes e o índice do item que gostaríamos de acessar.

```
[74] 1 cidades_que_visitei[0]
      'Brasília'

[75] 1 cidades_que_visitei[1]
      'Cairo'

[76] 1 cidades_que_visitei[3]
      'Paris'
```


## Estrutura de Dados: Lista

Também podemos selecionar janelas utilizando o sinal “:”

```
[78] 1 cidades_que_visitei[0:2]  
      ['Brasília', 'Cairo']
```

**[ 0 : 2 ]**  
inclusivo exclusivo

```
cidades_que_visitei = ['Brasília', 'Cairo', 'Tel-Aviv', 'Paris', 'Jerusalem', 'Eilat']
```



```
idades_que_visitei = ['Brasília', 'Cairo', 'Tel-Aviv', 'Paris', 'Jerusalem', 'Eilat']
```

**[ 0 : 2 ]**  
inclusivo exclusivo

## *Estrutura de Dados: Lista*

```
[79] 1 cidades_que_visitei[:]  
      ['Brasília', 'Cairo', 'Tel-Aviv', 'Paris', 'Jerusalem', 'Eilat']  
  
[80] 1 cidades_que_visitei[2:]  
      ['Tel-Aviv', 'Paris', 'Jerusalem', 'Eilat']  
  
[88] 1 cidades_que_visitei[3:5]  
      ['Paris', 'Jerusalem']
```

# *Estrutura de Dados: Lista*

Qual o tamanho da lista?

Podemos usar a função “len()” para descobrir.

Ela é uma função nativa do python, ou seja, ela não depende de nenhuma classe (ou objeto) e serve para descobrir a extensão de um objeto que é passado para ela.

```
[71] 1 len(cidades_que_visitei)
```

```
6
```

```
[72] 1 len(tweet)
```

```
33
```

## *Estrutura de Dados: Lista*

Listas **são poderosas**  
porque elas podem conter  
**outros tipos de dados**  
dentro delas.

```
[73] 1  paises_que_visite = ['Brasil', {'Jerusalem':5, 'Tel-Aviv':2, 'Eilat':1}, ('França', 'Paris')]
```

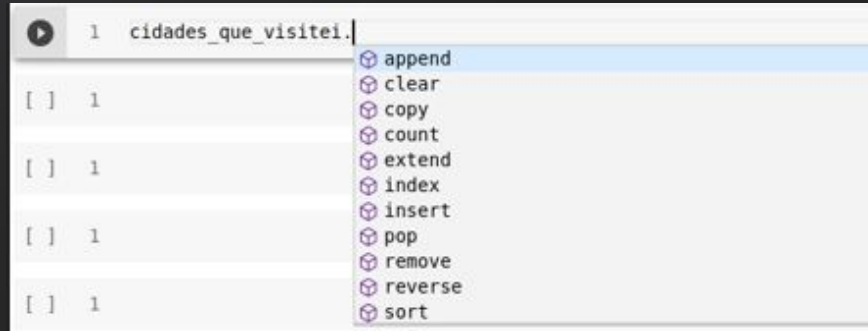
string

dicionário

tupla

## *Estrutura de Dados: Lista*

Listas também possuem  
**métodos**



# *Estrutura de Dados: Lista*

Método	Resultado
<code>.append(valor)</code>	Adiciona um valor no final da lista.
<code>.count(valor)</code>	Conta a quantidade de ocorrências de um valor específico dentro da lista
<code>.index(valor)</code>	Retorna o índice do valor.
<code>.reverse()</code>	Altera a ordem dos valores de uma lista
<code>.sort()</code>	Organiza os valores de uma lista.
<code>.pop(índice)</code>	Retorna o valor no índice especificado e deleta aquele valor da lista.
<code>.remove(valor)</code>	Remove a primeira ocorrência de valor

# *Estrutura de Dados: Tupla*

A “tupla” é uma lista imutável.

A principal diferença entre a tupla e a lista é a impossibilidade de adição de mais elementos após a criação.

Para criar uma tupla basta utilizar parênteses ou a função tuple().

```
1 fundo_imob = ('ALZR11', 'BBF011', 'BLMG11', 'BLCP11', 'BRC011')
2 fundo_imob

('ALZR11', 'BBF011', 'BLMG11', 'BLCP11', 'BRC011')
```



## *Estrutura de Dados: Tupla*

As tuplas também são indexadas iguais as listas

```
1 print(fundo_imob[0])  
2 print(fundo_imob[1])  
3 print(fundo_imob[2])
```

```
ALZR11  
BBF011  
BLMG11
```

## Estrutura de Dados: Tupla

Porém não podemos alterar os valores de uma tupla.

```
1 print(fundo_imob[0])
2 print(fundo_imob[1])
3 print(fundo_imob[2])
```

```
ALZR11
BBF011
BLMG11
```

```
[6] 1 fundo_imob[0] = 'BACON11'
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-6-12e9c749c4c8> in <module>()
----> 1 fundo_imob[0] = 'BACON11'
```

```
TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

## *Estrutura de Dados: Tupla*

A “tupla” é uma lista imutável. A principal diferença entre a tupla e a lista é a impossibilidade de adição de mais elementos após a criação.

Para criar uma tupla basta utilizar parênteses ou a função tuple().

```
1 fundo_imob = ('ALZR11', 'BBF011', 'BLMG11', 'BLCP11', 'BRC011')
2 fundo_imob

('ALZR11', 'BBF011', 'BLMG11', 'BLCP11', 'BRC011')
```

## *Estrutura de Dados: Tupla*

As tuplas também são indexadas iguais as listas.

```
1 print(fundo_imob[0])  
2 print(fundo_imob[1])  
3 print(fundo_imob[2])
```

```
ALZR11  
BBF011  
BLMG11
```

## Estrutura de Dados: Tupla

Porém não podemos alterar os valores de uma tupla.

```
1 print(fundo_imob[0])
2 print(fundo_imob[1])
3 print(fundo_imob[2])
```

```
ALZR11
BBF011
BLMG11
```

```
[6] 1 fundo_imob[0] = 'BACON11'
```

-----  
TypeError

Traceback (most recent call last)

<ipython-input-6-12e9c749c4c8> in <module>()

----> 1 fundo\_imob[0] = 'BACON11'

TypeError: 'tuple' object does not support item assignment

SEARCH STACK OVERFLOW

## *Estrutura de Dados: Tupla*

Tuplas podem ser heterogêneas também!

```
[8] 1 fundo_imob = tuple(['ALZR11',11,{'BLMG11':97.30},15.2,['BRC011',16]))  
    2 fundo_imob  
  
('ALZR11', 11, {'BLMG11': 97.3}, 15.2, ['BRC011', 16])
```

# Estrutura de Dados: Tupla

Perceba que por termos colocado uma lista dentro da tupla e lista serem mutáveis, existe a possibilidade de mudança daquela lista.

```
1 fundo_imob = tuple(['ALZR11',11,{'BLMG11':97.30},15.2,['BRC011',16]))
2 fundo_imob

('ALZR11', 11, {'BLMG11': 97.3}, 15.2, ['BRC011', 16])

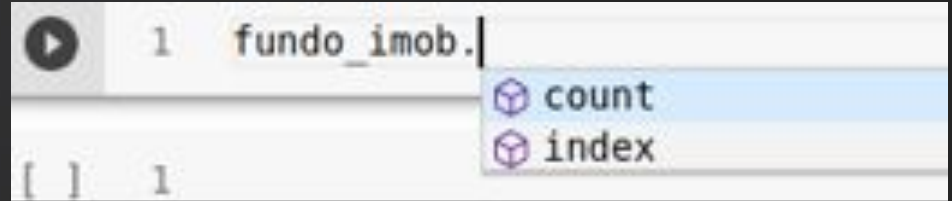
[10] 1 fundo_imob[4][1] = 15

[11] 1 fundo_imob

('ALZR11', 11, {'BLMG11': 97.3}, 15.2, ['BRC011', 15])
```

# *Estrutura de Dados: Tupla*

Por serem imutáveis as tuplas possuem poucos métodos.



A screenshot of a Python IDE with a dark background. The first line of code is `1 fundo_imob. |`. A dropdown menu is open, showing two methods: `count` and `index`, each preceded by a small purple cube icon. The second line of code is partially visible as `[ ] 1`.



## *Estrutura de Dados: Tupla*

O método `.count(11)` conta quantos valores '11' existem dentro da tupla.

```
[50] 1 fundo_imob  
      ('ALZR11', 11, {'BLMG11': 97.3}, 15.2, ['BRC011', 15])
```

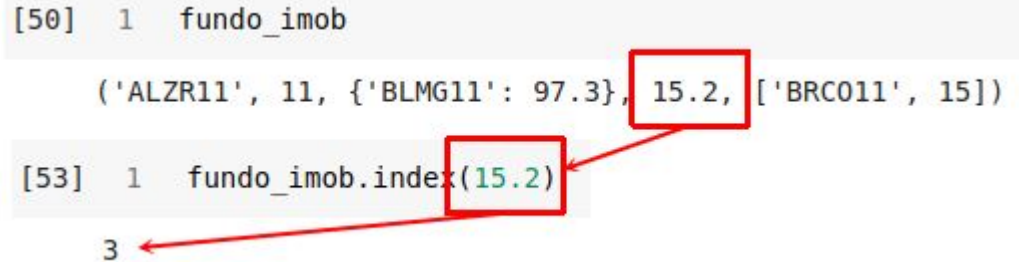
```
[35] 1 fundo_imob.count(11)  
      1
```

## *Estrutura de Dados: Tupla*

O método `.index(15.2)` retorna o índice onde o valor 15.2 se encontra.

```
[50] 1 fundo_imob
      ('ALZR11', 11, {'BLMG11': 97.3}, 15.2, ['BRC011', 15])

[53] 1 fundo_imob.index(15.2)
      3
```

A diagram with red boxes and arrows illustrating the `.index()` method. A red box highlights the value `15.2` in the tuple `('ALZR11', 11, {'BLMG11': 97.3}, 15.2, ['BRC011', 15])` at line [50]. Another red box highlights the argument `15.2` in the method call `fundo_imob.index(15.2)` at line [53]. A red arrow points from the box around `15.2` in the tuple to the box around `15.2` in the method call. A second red arrow points from the box around the method call to the output `3`.

# Obrigado

 /mesttra mesttra.com

