

Curso de Python_{v4}

Uma introdução ao python:

- Tipos de dados
- Loops
- Condicionais

Disclaimer



<https://penseallen.github.io/PensePython2e/>

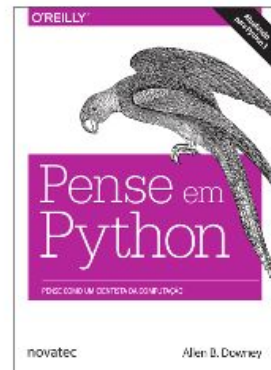
Pense em Python


Pense como um cientista da computação

Este livro ensina programação para quem nunca programou, usando **Python 3** nos exemplos. É aplicado no Olin College, IBMEC e outras faculdades de engenharia de primeira linha.

DICA: Você pode comprar um exemplar impresso de **Pense em Python** no site da **Editora Novatec** ou em livrarias. ISBN: 978-85-7522-508-0.


Pense em Python é uma tradução do livro **Think Python** (2ª edição), de **Allen B. Downey**, traduzido e publicado no Brasil pela **Editora Novatec** sob licença da O'Reilly Media.





Olar bbs

```
{twitter,  
  git(lab|hub),  
  telegram,  
} /dunossauro
```

 Live de
Python

<https://www.youtube.com/c/eduardomendes>



Changelog

V2

- + print
- + input

V3

- - UTF-8
- * Simplificação da parte de objetos
- * Métodos de lista
- * Float - exercício 4
- * Strings if-else
- + Formatação de strings
- + Capa
- + conectivos (and, not, or) - TODO
- + condições vazias (0, False, None) - TODO
- + Revisar exercícios - TODO
- + Tipos de argumentos de funções - TODO

V4

- + Comentários sobre else (while e for)
- -+ Retornos de listas
- + Condições falsas nos ifs
- + Conectivos lógicos
- Melhorias nos slide de print e input
- + Correções gramaticais



- Variáveis não são caixas
 - Tudo é objeto
 - Tipos numéricos
 - Estruturas de decisão
 - Strings
 - Laço While
 - Laço for
 - Listas
 - Funções
- Tuplas
 - Conjuntos
 - Dicionários

Estrutura



Antes de tudo, duas funções importantes

```
print("Minha mensagem")
```

**Mostra um texto
na tela.**

```
input("insira um texto")
```

**Pede para o
usuário inserir um
texto**



Problema #1

Faça um programa que escreva 'Olá mundo' na tela



Problema #2

Faça um programa que pergunte o nome e a idade do usuário e o exiba na tela.

“Bem vindo <nome>, fico feliz em saber que tem <idade> anos”

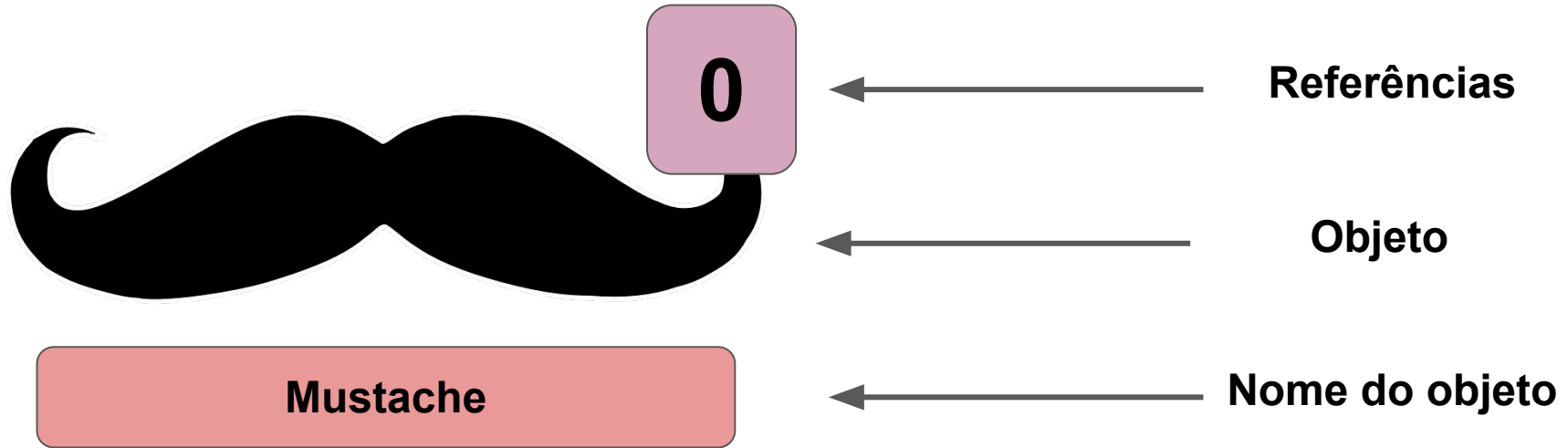


Variáveis não são caixas

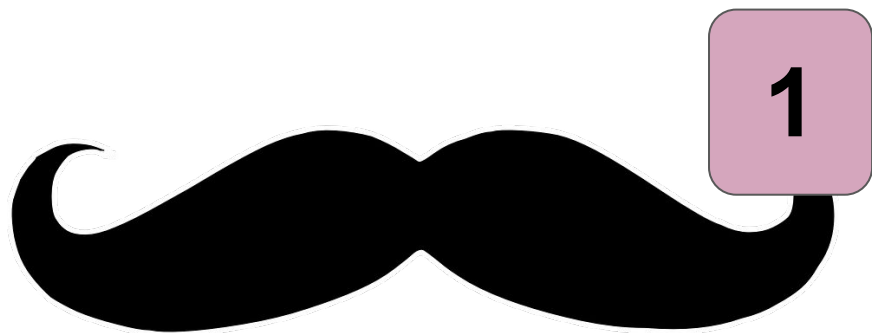
Ou a triste história do bigode



Variáveis -> Alias



Variáveis -> Alias



```
bigodinho = mustache()
```

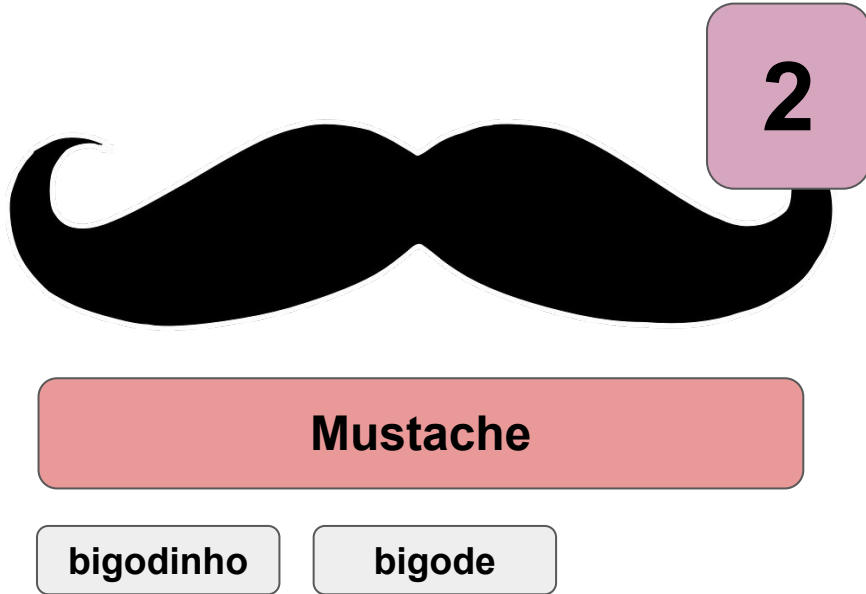
Mustache

bigodinho



Apelido carinhoso

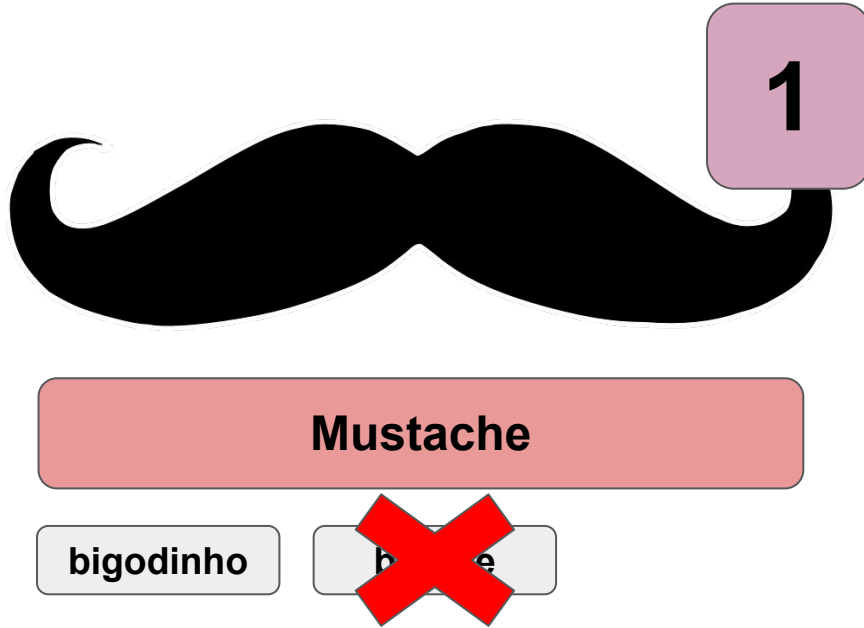
Variáveis -> Alias



```
bigodinho = mustache()
```

```
bigode = bigodinho
```

Variáveis -> Alias

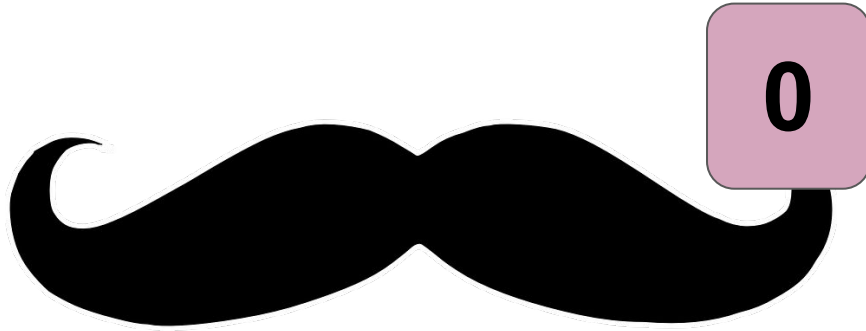


```
bigodinho = mustache()
```

```
bigode = bigodinho
```

```
del bigode
```

Variáveis -> Alias



Mustache

~~bigodinho~~

~~bigode~~

```
bigodinho = mustache()
```

```
bigode = bigodinho
```

```
del bigode
```

```
del bigodinho
```



Variáveis -> Alias



0

**Objeto
destruído**

tache()

= bigodinho

Mustache

~~bigodinho~~

~~bigodinho~~

del bigode

del bigodinho

Tudo é objeto

Um pouco sobre introspecção



Tudo é objeto

```
>>> num = 7  
>>> type(num)  
<class 'int'>
```

Atribuição de 7 a num

Retorna o tipo de dado
contido na variável

Objeto inteiro

Tudo é objeto

```
>>> num = 7  
>>> type(num)  
<class 'int'>
```

Atribuição de 7 a num

Retorna o tipo de dado
contido na variável

Objeto inteiro

**As estruturas de
dados tem atributos
e métodos**

Tudo é objeto

Retorna a lista de métodos do objeto

```
>>> type('a')  
      <class 'str'>  
>>> dir('a')  
>>> ['count', 'index', 'lower', 'replace', 'upper', '...']
```

Tipos numéricos

Int, Float, Complex



Números Inteiros

Tipo	Chamada	Resultado
Base 10	11	11
Base 2	0b11	3
Base 8	0o11	9
Base 16	0x11	17

```
>>> type(0x11)  
<class 'int'>
```

```
>>> print(0x11 + 0b11)  
20
```



Números de ponto flutuante e complexos

```
>>> type(11.4)  
<class 'float'>
```

```
>>> type(11.0 + 1j)  
<class 'complex'>
```

```
>>> 11 + 1j + 11.04  
(22.04+1j)
```

Operações com números [0]

$+$, $-$, $*$, $$, $/$, $//$, $\%$**

$$2 + 2 = 4$$

$$2 - 2 = 0$$

$$2 * 2 = 4$$

$$2 ** 2 = 4$$

$$2 / 2 = 1.0$$

$$2 \% 2 = 0$$

$$2 // 2 = 1$$

Operações com números [1]

$+$, $-$, $*$, $$, $/$, $//$, $\%$**

Toda operação de inteiros retorna inteiro*;

Toda operação com um float retorna float;

Toda operação com complexos retorna complexos

$2 / 2 = 1.0$

$2 // 2 = 1$

Exercício #3

Faça um programa para uma loja de tintas. O programa deverá pedir o tamanho em metros quadrados da área a ser pintada. Considere que a cobertura da tinta é de 1 litro para cada 3 metros quadrados e que a tinta é vendida em latas de 18 litros, que custam R\$ 80,00. Informe ao usuário a quantidades de latas de tinta a serem compradas e o preço total.



Exercício #4

Faça um programa que peça 2 números inteiros e um número float. Calcule e mostre:

- O produto do dobro do primeiro com metade do segundo .
- A soma do triplo do primeiro com o terceiro.
- O terceiro elevado ao cubo.

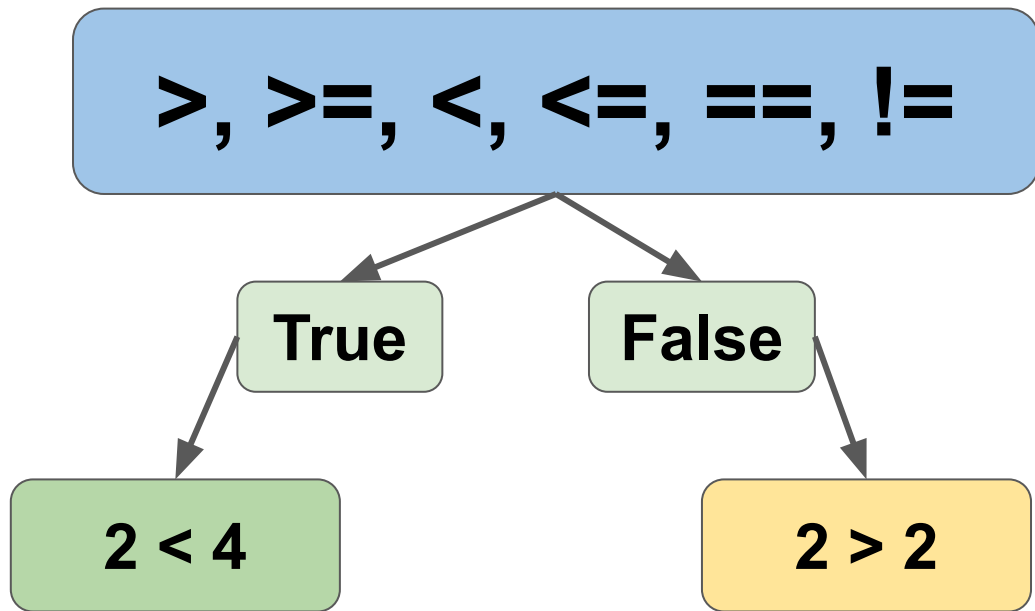


Tipos Booleanos

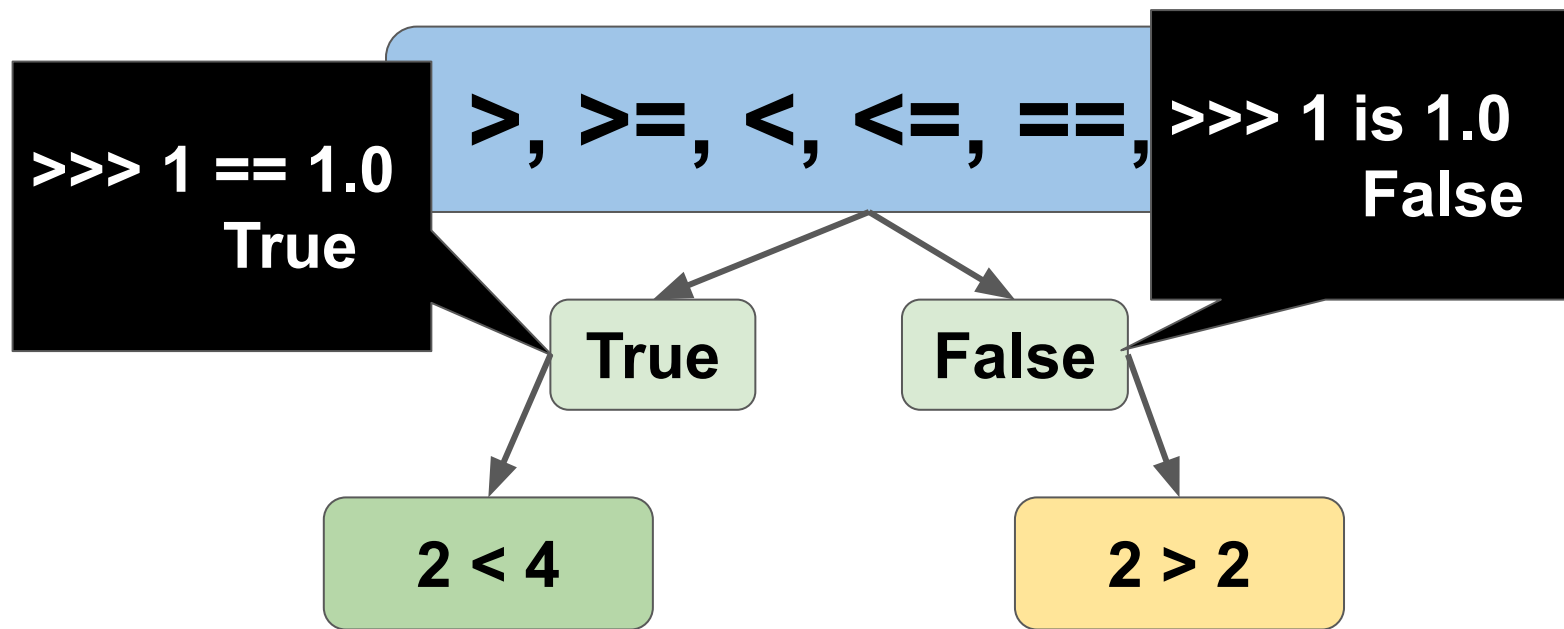
True, False, None e conectivos



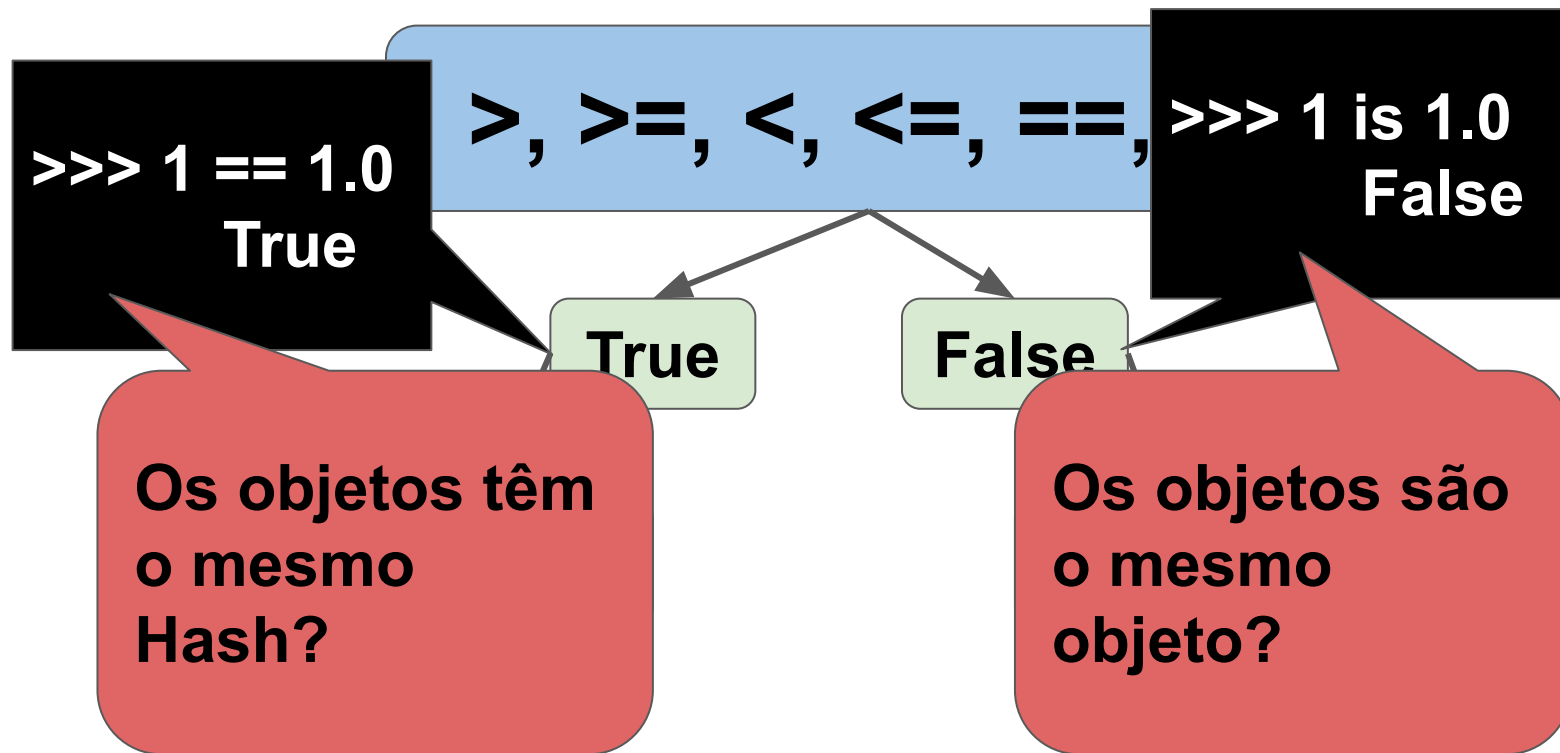
Operações com números [2] - Bool



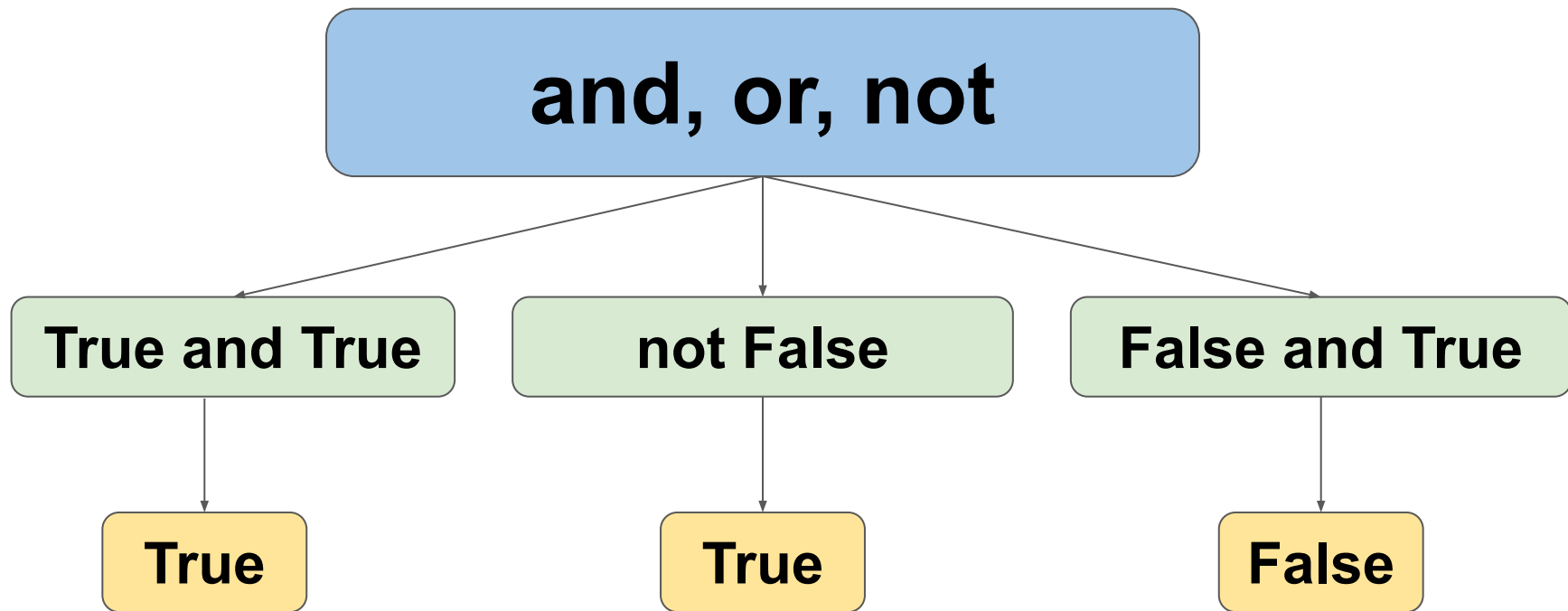
Operações com números [3] - Bool



Operações com números [3] - Bool



Conectores lógicos



Estruturas de decisão - if,elif,else

```
>>> x = 7
>>> y = 6
>>> if x == y:
    print('Mesmo valor')
elif x > y:
    print('x é maior que y')
else:
    print(f'Não sei resolver {x} e {y}')
```



Estruturas

Parênteses não são necessários,
nem mesmo com conectivos, ex:
if x>y or x<y

```
>>> x = 7
>>> y = 6
>>> if x == y:
    print('Mesmo valor')
elif x > y:
    print('x é maior que y')
else:
    print(f'Não sei resolver {x} e {y}')
```

Condições Falsas

0

False

None

if

0

:

Não entrará nesse bloco

Condições Falsas - Sequências vazias

‘ ’

[]

{ }

if

[]

:

Não entrará nesse bloco

Exercício #5

Faça um programa para a leitura de duas notas parciais de um aluno. O programa deve calcular a média alcançada por aluno e apresentar:

- A mensagem "Aprovado", se a média alcançada for maior ou igual a sete;
- A mensagem "Reprovado", se a média for menor do que sete;
- A mensagem "Aprovado com Distinção", se a média for igual a dez.



Exercício #6

Faça um programa que pergunte o preço de três produtos e informe qual produto você deve comprar, sabendo que a decisão é sempre pelo mais barato.



Strings

O básico necessário



Inicialização

```
brian = 'ROMANES EUNT DOMUS'
```

```
brian = "ROMANES EUNT DOMUS"
```

```
brian = """ROMANES  
EUNT  
DOMUS"""
```

Concatenação (mat)

```
>>> brian + brian  
      ROMANES EUNT  
      DOMUSROMANES EUNT  
      DOMUS
```

```
>>> brian * 2  
      ROMANES EUNT  
      DOMUSROMANES EUNT  
      DOMUS
```


Inicialização

```
brian = 'ROMANES EUNT DOMUS'
```

```
brian = "ROMANE
```

```
brian = """ROMANES  
EUNT  
DOMUS"""
```

Multilinha

Concatenação

```
>>> brian + brian  
      ROMANES EUNT  
      DOMUSROMANES EUNT  
      DOMUS
```

```
>>> brian * 2  
      ROMANES EUNT  
      DOMUSROMANES EUNT  
      DOMUS
```

Métodos - Strings

```
>>> a = 'Abracadabra'
```

```
>>> a.count('a') #4
```

```
>>> a.index('c') #4
```

```
>>> a.partition('c')  
# ('Abra', 'c', 'adabra')
```

```
>>> a.replace('a','c')  
# 'Abrcccdcbrc'
```

```
>>> a.lower()  
# 'abracadabra'
```

```
>>> a.split('a')  
# ['Abr', 'c', 'd', 'br', '']
```

Métodos - Strings

```
>>> a = 'Abracadabra'
```

```
>>> 'a' in 'abracadabra'  
True
```

retorna uma
Tupla

```
a.count('a') #4
```

```
>>> a.partition('c')  
# ('Abra', 'c', 'adabra')
```

```
>>> a.lower()  
# 'abracadabra'
```

```
>>> a.index('c') #4
```

```
>>> a.replace('c', 'd')  
# 'Abrccadabra'
```

```
>>> a.split('a')  
# ['Abr', 'c', 'd', 'br', '']
```

retorna uma
Lista

Formatação - strings

```
>>> nome = 'Eduardo'; idade = 25
```

F-string

```
>>> f'Nome: {nome}, Idade: {idade}'  
# 'Nome: Eduardo, Idade 25'
```

str.format()

```
>>> 'Nome: {}, Idade: {}'.format(nome, idade)  
# 'Nome: Eduardo, Idade 25'
```

Estilo antigo

```
>>> 'Nome: %s, Idade: %s' % (nome, idade)  
# 'Nome: Eduardo, Idade 25'
```



Exercício #7

Faça um programa que receba uma string e responda se ela tem alguma vogal, se sim, quais são? E também diga se ela é uma frase ou não.



Exercício #8

Faça um programa que receba uma data de nascimento (15/07/87) e imprima

‘Você nasceu em <dia> de <mês> de <ano>’



Laços de repetição



While

```
>>> while BOOL:  
    do  
else:  
    do
```

```
>>> while 3 < 4:  
    print("Três é menor")
```

```
>>> while True:  
    # ???
```

```
>>> x = 0  
>>> while x > 10:  
    x = input("um num: ")
```


While

Executa o else quando acabar o while, mas não com um break

```
>>> while True:
    do
else:
    do
```

```
>>> while 3 < 4:
    print("Três é menor")
```

Entrada do teclado

```
>>> while True:
    # ???
```

```
>>> x = 0
>>> while x > 0:
    x = input("um num: ")
```

Exercício #9

Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.



Exercício #10

Faça um programa que leia 5 números e informe o maior número.



For

```
>>> for e in <iter>:  
    do  
else:  
    do
```

```
>>> for e in "grupy":  
    print(e)
```

```
>>> for e in [1,2,3]:  
    print(e)
```

```
>>> for i,e in enumerate([1,2,3]):  
    print(i,e)
```

```
# 0 1  
# 1 2  
# 2 3
```



For

```
>>> for e in [1,2,3]:  
    do  
else:  
    do
```

**Executa o else
quando acabar o
while, mas não
com um break**

```
>>> for e in "grupy":  
    print(e)
```

```
>>> for e in [1,2,3]:  
    print(e)
```

```
>>> for i,e in enumerate([1,2,3]):  
    print(i,e)
```

```
# 0 1  
# 1 2  
# 2 3
```



Exercício # 11

Faça um programa que itera em uma string e toda vez que uma vogal aparecer na sua string print o seu nome entre as letras

```
string = bananas
```

```
b
```

```
eduardo
```

```
n
```

```
eduardo
```

```
n
```

```
...
```



Exercício # 12

Faça um programa que receba uma string, com um número de ponto flutuante, e imprima qual a parte dele que não é inteira

EX:

n = '3.14'

resposta: 14



Listas

Sim, elas aceitam tudo



Elas aceitam todos os tipos de objeto - Listas

```
>>> a = [1, 1. + 1j, "eduardo", [1,2,3], (1,2)]
```

```
>>> a[0] # 1  
>>> a[1] # 1. +j  
>>> a[2] # "eduardo"  
>>> a[3] # [1,2,3]
```

```
>>> a[3][0] # 1  
>>> a[1][1] # 2  
>>> a[4][0] # 1  
>>> a[2][-1] # "o"
```

Slice - Listas

```
>>> n = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> n[0]      # 0  
>>> n[6:]     # [6, 7, 8, 9]  
>>> n[:-6]    # [0, 1, 2, 3]  
>>> n[::2]    # [0, 2, 4, 6, 8]
```

[De onde : até onde: de quanto em quanto]

[2 : 10 : 3]



```
5] # [0, 1, 2, 3]  
>>> n[::2] # [0, 2, 4, 6, 8]
```

Slice - Listas

```
>>> matriz = [ [0, 1, 2], [3, 4, 5], [7, 8, 9] ]
```

```
>>> matriz[0]                # [0, 1, 2]
```

```
>>> matriz[0][1:]           # [1, 2]
```

```
>>> matriz3d = [ [ [0, 0, 0] ], [ [0, 0, 0] ] ]
```

```
>>> matriz3d[0][0][0]       # 0
```

Métodos - Listas

```
>>> x = [1, 2, 3]
```

```
>>> x.append(4)  
# [1, 2, 3, 4]
```

```
>>> x.remove(2)  
# [1, 3]
```

```
>>> x.insert(4, 0)  
# [1, 2, 3, 0]
```

```
>>> x.pop()  
# 3 | # [1, 2]
```

```
>>> x.count(2)  
# 1
```

```
>>> x.reverse()  
#None | # [3, 2, 1]
```

Métodos - Listas

Fila

```
>>> x = [1, 2, 3]
```

```
>>> x.append(4)  
# [1, 2, 3, 4]
```

```
>>> x.insert(4, 0)  
# [1, 2, 3, 0]
```

```
>>> x.count(2)  
# 1
```

Lista

```
>>> x.remove(2)  
# [1, 3]
```

```
>>> x.pop()  
# 3 | # [1, 2]
```

```
>>> x.reverse()  
#None | # [3, 2, 1]
```

Pilha

Exercício #13

Faça um programa que: Dada uma lista [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] e um número inteiro, imprima a tabuada desse número.



Exercício #14

Faça uma programa que dada a entrada de uma lista ele faça o cálculo acumulativo da mesma:

Exemplo de entrada: [1, 2, 3, 4]

Exemplo de saída: [1, 3, 6, 10]



Exercício #15

Faça um programa que dada a entrada de uma lista o programa calcule a combinação de dois elementos e nos retorne as combinações em uma nova lista.

Exemplo de entrada: [1, 2, 3, 4]

Exemplo de saída: [[1, 2], [1, 3], [1, 4], [2, 3], [2, 4], [3, 4]]

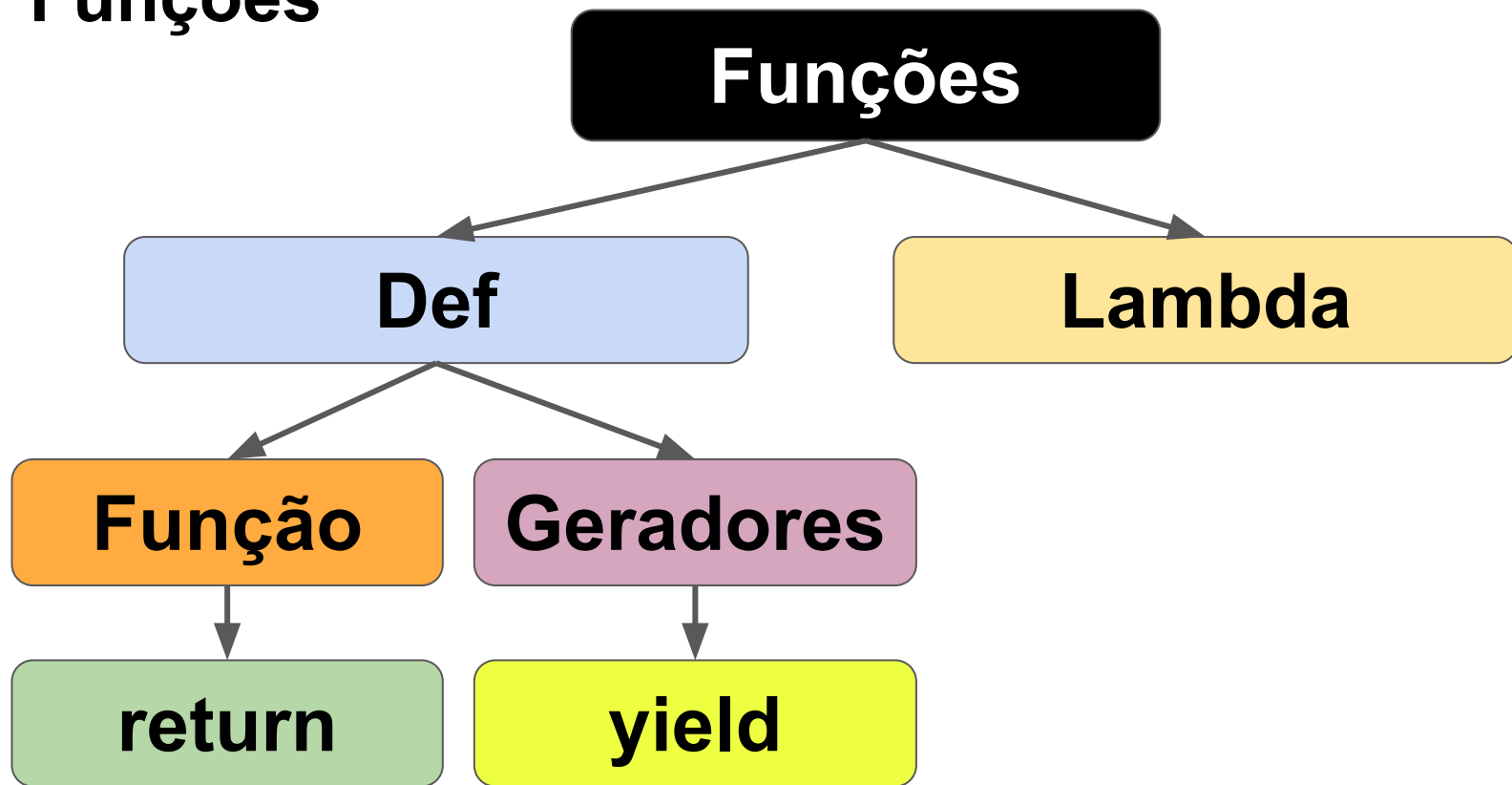


Funções

Elas também são objetos



Funções



Padrões[0] - Funções

Função nomeada

```
def nome (args):  
    return args
```

Função anônima

```
lambda args: op(args)
```

Função geradora

```
def nome (args):  
    yield args
```

Padrões[2] - Funções

```
def func(a, b=1, *c, **d):  
    return a, b, c, d
```

```
func(*[1, 2, 3, 4, 5], x=7 )  
# 1 2 (3, 4, 5) {'x': 7}
```

Empacotar

Dicionário

Desempacotar

Exercício #16

Faça um programa, com uma função, que calcula a média de uma lista.

Funções embutidas que podem te ajudar:

- `len(lista)` -> calcula o tamanho da lista
- `sum(lista)` -> faz o somatório dos valores



Exercício #17

Faça um programa, com uma função, que calcula a mediana de uma lista.

Funções embutidas que podem te ajudar:

- `sorted(lista)` -> ordena a lista



Exercício #18

Faça um programa, com uma função que dado uma lista e uma posição da mesma faça o quartil dessa posição.

```
p_index = int(p * len(lista))
```



Exercício #19

Faça um programa, com uma função, que calcule a dispersão de uma lista

Funções embutidas que podem te ajudar:

- `min(lista)` -> retorna o menor valor
- `max(lista)` -> retorna o maior valor



Tuplas

Elas não são só listas imutáveis



Métodos - Tuplas

```
>>> x = (1, 2, 3)
```

```
>>> x.count(2)  
# 1
```

```
>>> x.index(2)  
# 1
```

```
>>> x = 1  
>>> y = 2  
>>> x,y = y,x
```

```
>>> print(x)  
# 2  
>>> print(y)  
# 1
```

“Empacotamento” - Tuplas

```
>>> x = ( 1, 2, 3, 4, 5 )
```

```
>>> *a, b, c = x  
# (1, 2, 3) 4 5
```

```
>>> a, *b, c = x  
# 1 (2, 3, 4) 5
```

```
>>> a, b, *c = x  
# 1, 2, (4, 5)
```

```
>>> *a, b, *c = x  
# (1, 2) 3 (4, 5)
```



“Empacotamento” - Tuplas

```
>>> x = ( 1, 2, 3, 4, 5 )
```

```
>>> *a, b, c = x  
# (1, 2, 3) 4 5
```

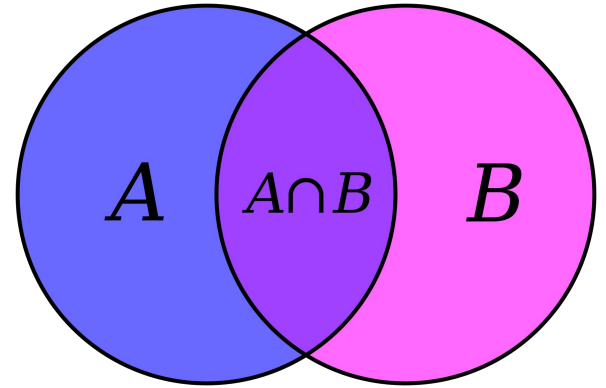
```
>>> a, *b, c = x  
# 1 (2, 3, 4) 5
```

```
>>> a, b, *c = x  
# 1, 2, (4, 5)
```

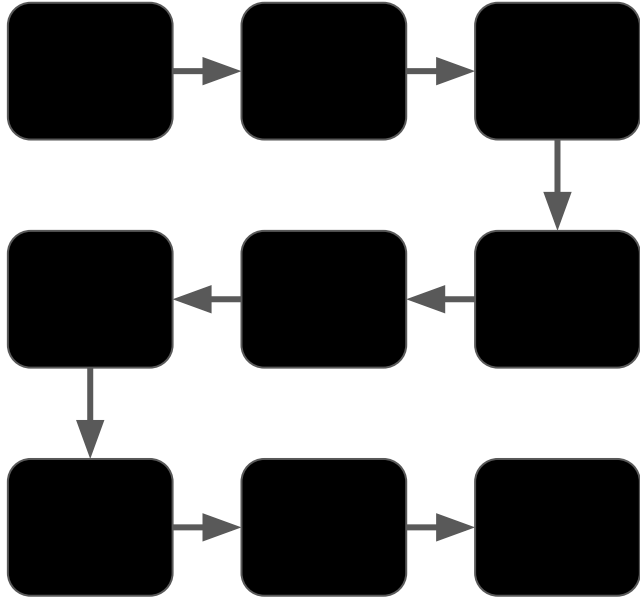
```
>>> a, b, *c = x  
# 1, 2, (4, 5)
```

Conjuntos

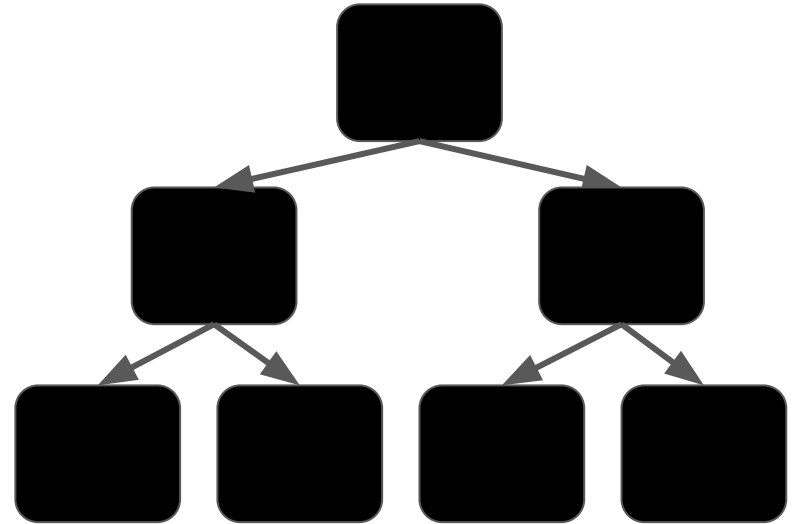
Valores “fixos”



Que raios são hashable? - Conjuntos



Litas e Tuplas



Conjuntos e dicionários



Métodos - Conjuntos

```
>>> x = {1, 2, 3}; y = {3, 4, 5}
```

```
>>> x.union(y)  
# {1, 2, 3, 4, 5}
```

```
>>> x.intersection(y)  
# {3}
```

```
>>> x.difference(y)  
# {1, 2}
```

```
>>> x.update(y)  
# {1, 2, 3, 4, 5}
```

```
>>> x.discard(1)  
# {2, 3}
```

```
>>> x.pop()  
# {2, 3}
```


Métodos - Conjuntos

```
>>> x = {1, 2, 3}; y = {3, 4, 5}
```

Retorna um novo

```
>>> x.union(y)  
# {1, 2, 3, 4, 5}
```

```
>>> x.difference(y)  
# {1, 2}
```

Remove o 1º

```
>>> x.discard(1)  
# {2, 3}
```

```
>>> x.intersection(y)  
# {3}
```

Atualiza x

```
>>> x.update(y)  
# {1, 2, 3, 4, 5}
```

Remove o 1º

```
>>> x.pop()  
# {2, 3}
```

Dicionários



Como eles funcionam? - Dicionários

- A chave necessariamente deve ser hashable
- Os valores podem ser qualquer coisa
- Similar à um JSON

```
Pessoa = {  
    'nome': 'eduardo',  
    'cargo': 'programador',  
    'função': lambda f, *args: f(args),  
    'saldo': {'dia 5': 150, 'dia 10': [0, -100, -500]}  
}
```



Como uso esse negócio? - Dicionários

```
>>> x = {'Maria': 50, 'Juana': 25}
```

```
>>> x['Maria']  
# 50
```

```
>>> x.keys()  
# ['Juana', 'maria']
```

```
>>> x['Maria'] = 10
```

```
>>> x.values()  
# [50, 25]
```

```
>>> x.popitem()  
# ('Juana', 25)
```

```
>>> x.setdefault('Carlos')  
# {'Juana': 25, 'maria': 100, 'Carlos':  
  None}
```

Exercício #20

Baseando-se nos exercícios passados, monte um dicionário com os seguintes seguintes chaves:

lista, somatório, tamanho, maior valor e menor valor



Exercício #21

Dada uma lista de entradas de usuário de números inteiros, construa um dicionário com a lista padrão, a lista dos valores elevados ao quadrado e a lista dos valores elevados ao cubo



XOXO

Dúvidas?

@dunossauro

