



# Introdução à linguagem Python

Aula 2 - Módulos, Condicionais e Funções

Por: Thiago Ribeiro da Silva

Estudante de Ciência da Computação (UFAL)

# O que são módulos?

- Módulos (ou bibliotecas) são, de modo básico, aglomerados de funções (funcionalidades) que podem ser chamados e usados em algum script python;
- Eles servem para otimizar os programas e evitar repetição de código.

## MATEMÁTICA

$\sin()$        $\tan()$

$\cos()$        $\log()$

$\text{floor}()$        $\text{ceil}()$   
 $\text{pow}()$

# O que são módulos?

- Para chamar (importar) um módulo para seu script, usamos o comando *import*;
- Ao usá-lo, para chamar qualquer função de um módulo, usa-se *nome\_do\_modulo.nome\_da\_funcao()*;
- Podemos chamar funções específicas com o comando: *from modulo import funcao*



```
1 import math
2
3 print(math.pow(2, 3)) #8
4
5 from math import log10
6
7 print(log10(10)) #1
```



# Alguns módulos internos

- Seguem alguns dos módulos internos encontrados no python

## Matemática (math)

FUNÇÕES	DESCRIÇÃO
ceil(x), floor(x)	arredonda para cima e para baixo, respectivamente
trunc(x)	retorna a parte fracionada de um float
pow(x), sqrt(x)	operação de potenciação e raiz quadrada
factorial(x)	fatorial de um número



# Alguns módulos internos

- Seguem alguns dos módulos internos encontrados no python

## Matemática (math)

FUNÇÕES	DESCRIÇÃO
degrees(x)	converte de radianos para graus
radians(x)	converte de graus para radianos
sin(x), cos(x), tan(x)	retornam seno, cosseno e tangente de um valor passado em radianos
pi, e	retornam o valor de pi e e



# Alguns módulos internos

- Seguem alguns dos módulos internos encontrados no python

Gerador de números pseudo-aleatórios (**random**)

FUNÇÕES	DESCRIÇÃO
random()	retorna um float entre 0.0 e 1.0
randint(a, b)	retorna um inteiro entre a e b
uniform(a, b)	retorna um float entre a e b



# Alguns módulos internos

- Seguem alguns dos módulos internos encontrados no python

Funções relacionadas ao tempo (**time**)

FUNÇÕES	DESCRIÇÃO
time()	retorna o tempo atual em segundos
sleep(x)	para a execução do algoritmo por x segundos

# Alguns exemplos



```
#### Encontrando a hipotenusa de um triângulo retângulo #####
from math import sqrt #importando a função raiz quadrada

cateto1 = float(input("cateto 1: "))
cateto2 = float(input("cateto 2: "))

hipotenusa = sqrt(cateto1 ** 2 + cateto2 ** 2)

print(f"A hipotenusa é {hipotenusa}")
```

```
↳ cateto 1: 3
    cateto 2: 4
    A hipotenusa é 5.0
```

# Alguns exemplos



```
#### Gerador de inteiros aleatórios ####  
from random import randint  
  
print("---- Determine o intervalo (fechado) ----\n")  
#O comando \n pula uma linha no print  
inicio = int(input("Inicio do intervalo: "))  
fim = int(input("Fim do intervalo: "))  
  
numero = randint(inicio, fim)  
  
print(f"Seu número aleatório é: {numero}")
```

↳ ---- Determine o intervalo (fechado) ----

```
Inicio do intervalo: 5  
Fim do intervalo: 14  
Seu número aleatório é: 12
```

# Alguns exemplos



```
#### Contando o tempo ####

from time import time, sleep

inicio = time()
print(f"antes do sleep: {inicio:.2f}")

sleep(2)
fim = time()
print(f"depois do sleep: {fim:.2f}")
print(f"diferença: {fim - inicio:.2f}")
```

```
antes do sleep: 1672002809.10
depois do sleep: 1672002811.10
diferença: 2.00
```

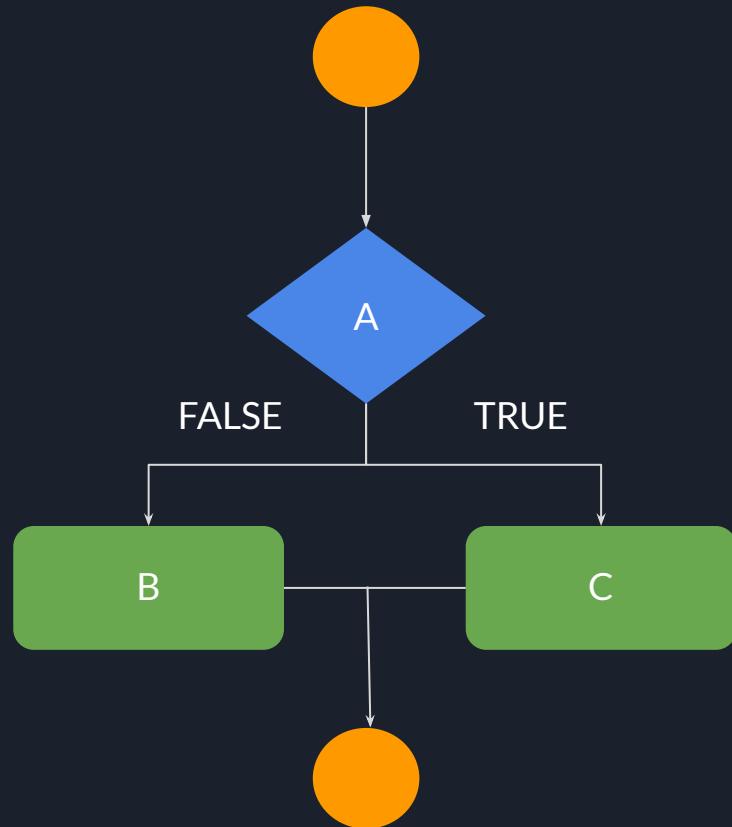
# Links para a documentação dos módulos

- Math: <https://docs.python.org/3/library/math.html>
- Random: <https://docs.python.org/3/library/random.html>
- Time: <https://docs.python.org/3/library/time.html>
- Outros: <https://docs.python.org/3/library/>



# Estrutura condicional

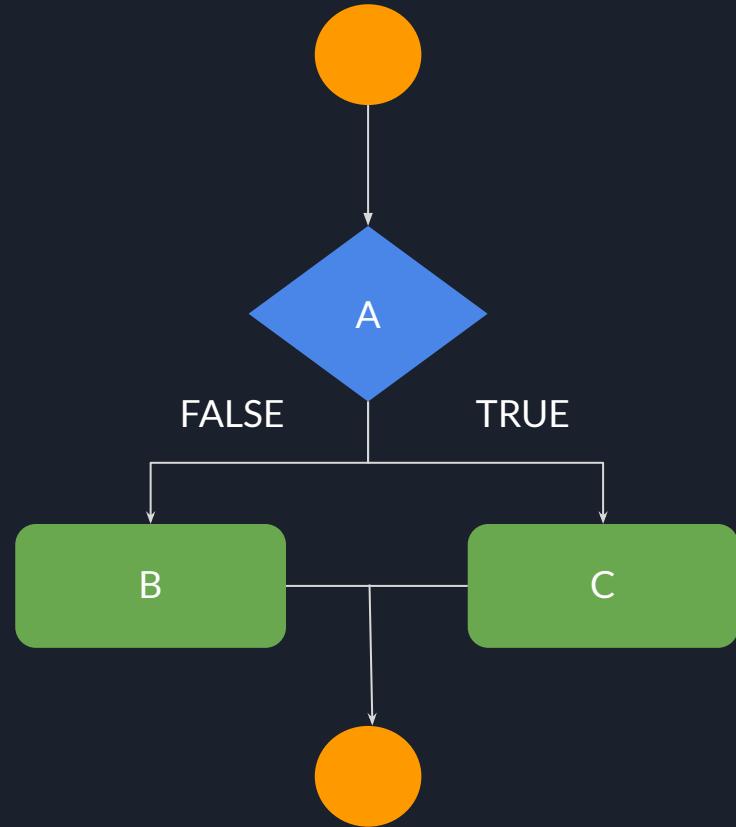
- Não são todas as vezes que algoritmos são lineares. Como vimos no algoritmo para escovar os dentes, existem situações em que surgem bifurcações ou até mesmo polifurcações em um caminho;
- Para tal, existem as estruturas de condição na programação.



# Estrutura condicional

- Sendo assim, dada uma condição, se for verdadeira, o algoritmo irá para um caminho, mas se for falsa, irá para outro diferente.
- No python, essa estrutura se dá da seguinte forma:

```
if <condicao> :  
    <execute isso>  
  
else:  
    <execute isso>
```



# Estrutura condicional - Exemplo 1

```
1 # Estrutura de condição simples
2 verdade = True
3
4 if verdade:
5     print("Verdade")
6 else:
7     print("Mentira")
```



# Operadores de comparação

- Para otimizar nossas condições, é importante conhecer alguns outros operadores

OPERADOR	DESCRIÇÃO	USO
> (maior que)	verifica se um valor é maior que outro	$x > 10$
< (menor que)	verifica se um valor é menor que outro	$x < 20$
== (igual a)	verifica se um valor é igual outro	$x == 15$
!= (diferente)	verifica se um valor é diferente de outro	$x != y$
>= (maior ou igual)	“ ” é maior ou igual a outro	$a >= b$
<= (menor ou igual)	“ ” é menor ou igual a outro	$15 <= 14.5$

# Estrutura condicional - Exemplo 2

```
[7] # Exemplo condicional comparativa

idade = int(input("Digite sua idade: "))

if idade >= 18:
    print("Maior de idade")
else:
    print("Menor de idade")
```

```
Digite sua idade: 18
Maior de idade
```



# Operadores lógicos

- Para otimizar nossas condições, é importante conhecer alguns outros operadores

OPERADOR	DESCRIÇÃO	USO
and	operador lógico “E”	$x > 10 \text{ and } x < 15$
or	operador lógico “OU”	$x > 1 \text{ or } x > 5$
not	operador lógico “NÃO”	$\text{not}(x > 15 \text{ and } x < 30)$



# Operadores lógicos

A	B	NOTA	A OR B	A AND B
True	True	False	True	True
True	False	False	True	False
False	True	True	True	False
False	False	True	False	False

# Estrutura condicional - Exemplo 3



```
# Exemplo condicional com lógica

idade = int(input("Digite sua idade: "))

if (idade >= 16 and idade < 18) or idade >= 70:
    print("Voto facultativo")
else:
    print("Voto obrigatório")
```

```
Digite sua idade: 70
Voto facultativo
```

# Estrutura condicional composta

- Escolhas quase nunca são binárias, portanto, as estruturas condicionais possuem recursos para  $n$  caminhos possíveis no algoritmo;
- “*se condição x for verdade, execute isso, se não, se condição y for verdade, execute isso, se nenhuma das duas for verdade, então execute isso.*”



```
1 # Estrutura de condição composta
2 cor = 'azul'
3
4 if cor == 'azul':
5     print("é azul")
6 elif cor == 'verde': # else if (elif)
7     print("é verde")
8 else:
9     print("Não é azul, nem verde")
```

# Estrutura condicional em linha

- Algumas linguagem possuem um operador chamado **ternário** usado para atribuir valor dada uma condição em linha única;
- condicao ? valor1 : valor 2;
- “*a condição é verdade? se sim, valor 1, se não, valor 2*”
- Python tem algo semelhante, da seguinte forma:
- valor1 **if** condicao **else** valor2



```
1 # operador ternário em python
2 idade = 18
3 eh_maior_de_idade = True if idade >= 18 else False
```

# Estrutura condicional - Exemplo 4



```
# Idade para votar

idade = int(input("Digite sua idade: "))

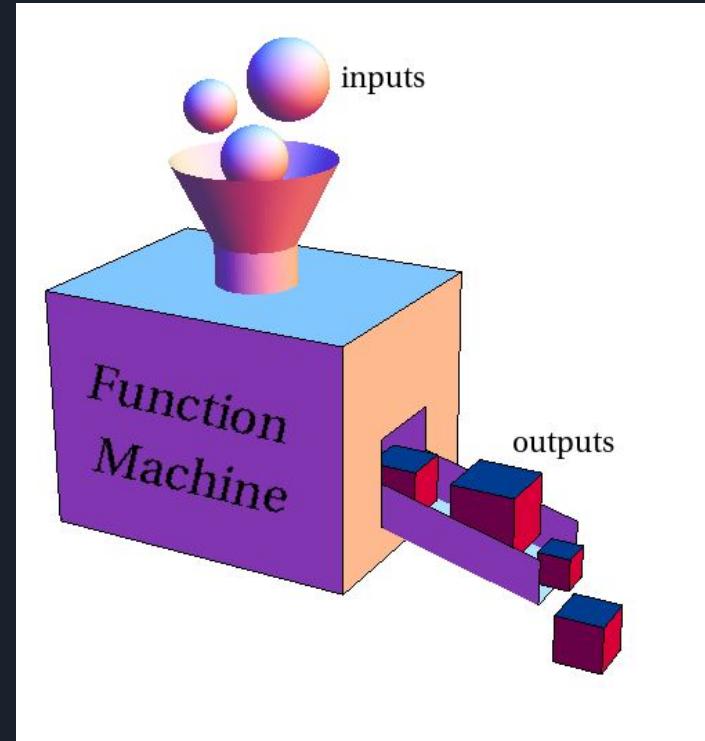
if idade < 16:
    print("Não pode votar")
elif (idade >= 16 and idade < 18) or idade >= 70:
    print("Voto facultativo")
else:
    print("Voto obrigatório")
```

Digite sua idade: 15

Não pode votar

# Entendendo funções

- Conhecemos funções matemáticas, que são um tipo de relação entre conjuntos. Sempre há uma entrada, um processamento e uma saída;
- A ideia de funções é extremamente semelhante na programação. Temos um trecho de código que é executado a partir do recebimento de parâmetros e pode retornar algo.



# Entendendo funções

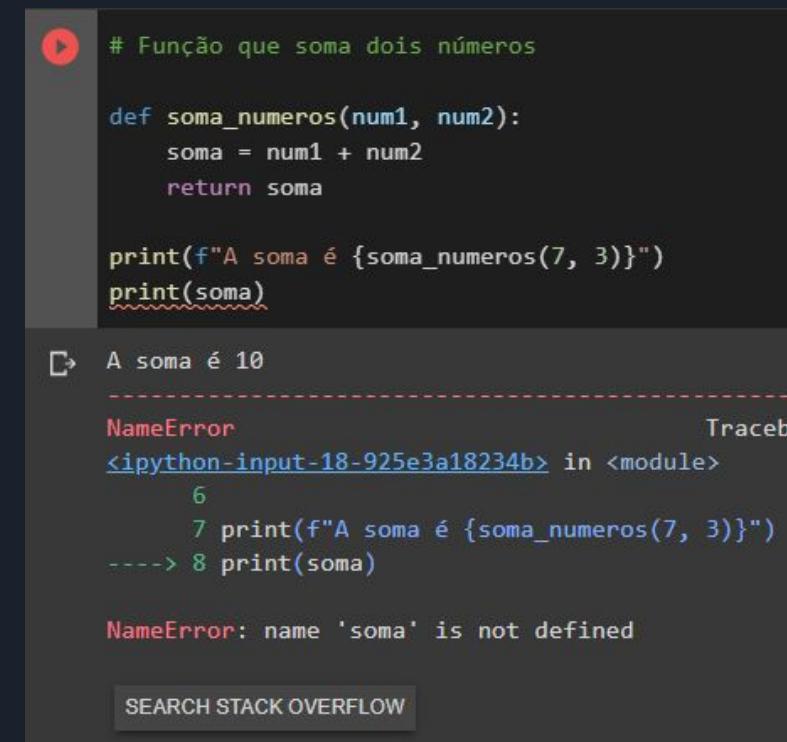
- A estrutura de funções em python é simples: temos que nomeá-la e então, entre parênteses, são colocados os parâmetros, separados por vírgula. Dentro do bloco de função, é escrito o processamento e é possível retornar um resultado como finalização.



```
1 # Função que soma dois números
2
3 def soma_numeros(num1, num2):
4     soma = num1 + num2
5     return soma
```

# Entendendo escopo de variáveis

- **Escopo** se refere ao local de existência de uma variável. Quando criamos uma nova variável dentro de uma função, está só existe dentro daquela função e, portanto, se tentarmos chamá-la fora, ocorrem erros.



The screenshot shows a Jupyter Notebook cell with the following code:

```
# Função que soma dois números

def soma_numeros(num1, num2):
    soma = num1 + num2
    return soma

print(f"A soma é {soma_numeros(7, 3)}")
print(soma)
```

The output of the cell is:

```
A soma é 10
-----
NameError: name 'soma' is not defined
```

Below the cell, there is a button labeled "SEARCH STACK OVERFLOW".

# Funções - Exemplo 1



```
# Programa que soma dois números

def soma_numeros(num1, num2):
    soma = num1 + num2
    return soma

num1 = int(input("Digite o primeiro numero: "))
num2 = int(input("Digite o segundo numero: "))

print(f"Resultado: {num1} + {num2} = {soma_numeros(num1, num2)}")
```

```
↳ Digite o primeiro numero: 14
Digite o segundo numero: 16
Resultado: 14 + 16 = 30
```

## Funções - Exemplo 2



```
#Programa que calcula a área do círculo
from math import pi

def area_circulo(raio):
    return pi * (raio ** 2)

r = float(input("Digite o raio do círculo: "))

print(f"A área do círculo é: {area_circulo(r):.2f}")
```

```
Digite o raio do círculo: 3.5
A área do círculo é: 38.48
```

Obrigado!

