Programação em Python

Funções e argumentos (parâmetros)

Prof. Daniel Di Domenico
https://github.com/danidomenico/gex003_algprog

Slides cortesia da profa. Andrea Charão (UFSM) e do prof. João V. F. Lima (UFSM)

Funções em Python

- Inspiração em funções matemáticas!
  - Ex: f(x, y) = ax + by + c
- Funções pré-definidas: extensão dos comandos da linguagem:
  - Ex: matemática avançada (*math.sqrt()*), gráficos (*pyplot.plot()*), listas (*len()*, *lista.append()*), etc.
- Programador também pode definir funções:
  - def f(x, y)

Funções em Python

- Exemplo: calcular o triplo de um número:

```
#Função que calcula o tripo de um número
def triplo(x):
    return x*3 #retorna o resultado

#Código principal...
num = triplo(4)
print("Triplo de 4 é", num)
Triplo de 4 é 12
num = triplo(6)
print("Triplo de 6 é", num)
Triplo de 6 é 18
```

Funções em Python

- Exemplo: calcular o triplo de um número:

```
#Função que calcula o tripo
def triplo(x):
    return x*3 #retorna o re

#Código principal...
num = triplo(4)
print("Triplo de 4 é", num)
Triplo de 4 é 12
num = triplo(6)
print("Triplo de 6 é", num)
Triplo de 6 é 18
```

Define a função chamada "**triplo**", que recebe uma variável (**x**) e retorna um valor através do **return**.

Evoca/chama a função "**triplo**" previamente definida, passando como parâmetro o valor 4.

Funções em Python

- Exemplo: calcular o triplo de um número:

```
#Função que calcula o tripo de um número
def triplo(x):
    return x*3 #retorna o resultado

#Código principal...
num = triplo(4)
print("Triplo de 4 é", num)
Triplo de 4 é 12
num = triplo(6)
print("Triplo de 6 é", num)
Triplo de 6 é 18
```

ATENÇÃO: a execução do programa inicia do código principal (fora de funções)

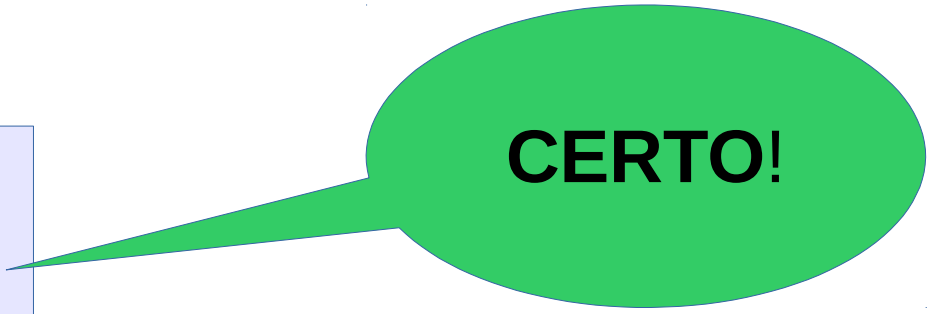
Funções em Python

- **Por que utilizar funções:**
  - Para permitir o reaproveitamento de código já construído (por você ou por outros programadores);
  - Para evitar repetições de um mesmo código;
  - Para facilitar alterações (manutenções) no código;
  - Para organizar o código, a fim de:
    - Evitar blocos de código muito extensos que tornam-se difíceis de entender;
    - Separar o programa em blocos que possam ser logicamente compreendidos de forma isolada.

Funções em Python

- Atenção ao **recuo (endentação!!!!)**;
- Não esquecer de recuar à direita:

```
def triplo(x):
    return x*3
```

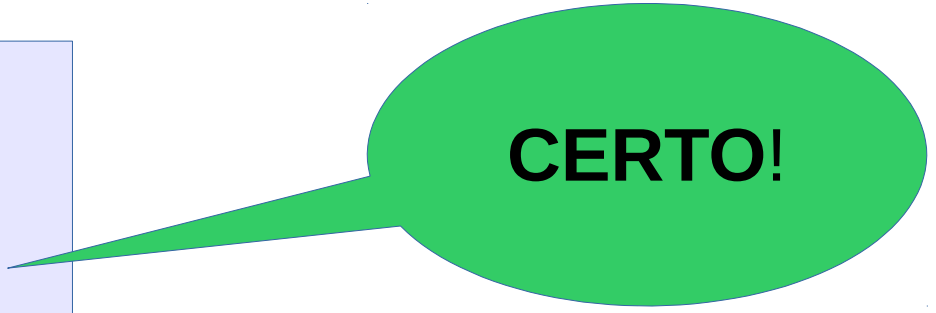



```
def triplo(x):
return x*3
```

Funções em Python

- As funções devem ser declaradas antes do código principal:

```
def triplo(x):
    return x*3

print(triplo(4))
```




```
print(triplo(4))

def triplo(x):
    return x*3
```

Funções em Python

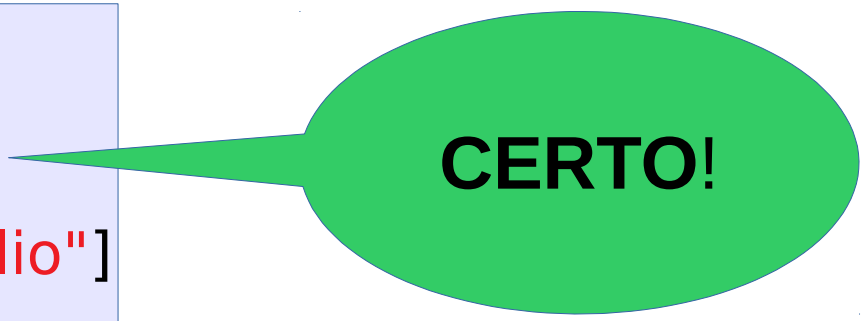
- Parâmetros e retorno:
  - Uma função pode ter nenhum (0) ou mais de 1 parâmetro:
    - **ATENÇÃO:** ao chamar a função, a ordem dos parâmetros deve ser condizente com a definição da função;
  - Uma função não é obrigada a retornar algo;

```
#Função que possui 2 parâmetros e nenhum retorno
def imprime_info(nome, idade):
    print("Nome:", nome)
    print("Idade:", idade)

#Código principal...
nome = "Jhon Travolta"
i = 52
imprime_info(nome, i) #Chamada da função
Nome: Jhon Travolta
Idade: 52
```

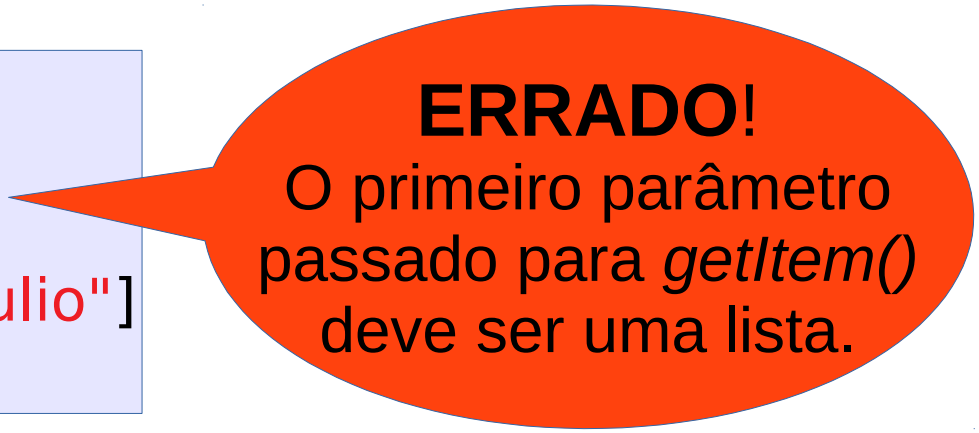
Funções em Python

- Parâmetros e retorno:
  - **ATENÇÃO:** os tipos dos dados passados por parâmetro devem ser compatíveis com os tipos esperados pela função:

```
def getItem(lista, idx):
    return lista[idx]

nomes = ["Carlos", "Rodrigo", "Julio"]
print(getItem(nomes, 1))
```




```
def getItem(lista, idx):
    return lista[idx]

nomes = ["Carlos", "Rodrigo", "Julio"]
print(getItem(1, nomes))
```



Funções em Python

- Comando **return**:
  - Quando executado, interrompe o fluxo da função;

```
def soma(n1, n2):
    if n1 == 0 and n2 == 0:
        return 25000 #se executado, termina a execução
                     #da função
    total = n1 + n2
    return total;

print("Soma 0 + 0  =", soma(0, 0))
print("Soma 10 + 20 =", soma(10, 20))
Soma 0 + 0  = 25000
Soma 10 + 20 = 30
```

Funções em Python

- Funções podem chamar outras funções:
  - Todas devem estar declaradas antes do código principal do programa:

```
def soma(n1, n2):
    return n1 + n2

def imprime():
    print("Total:", soma(1, 2))

#Código principal...
imprime()
```

Total 3

Funções em Python

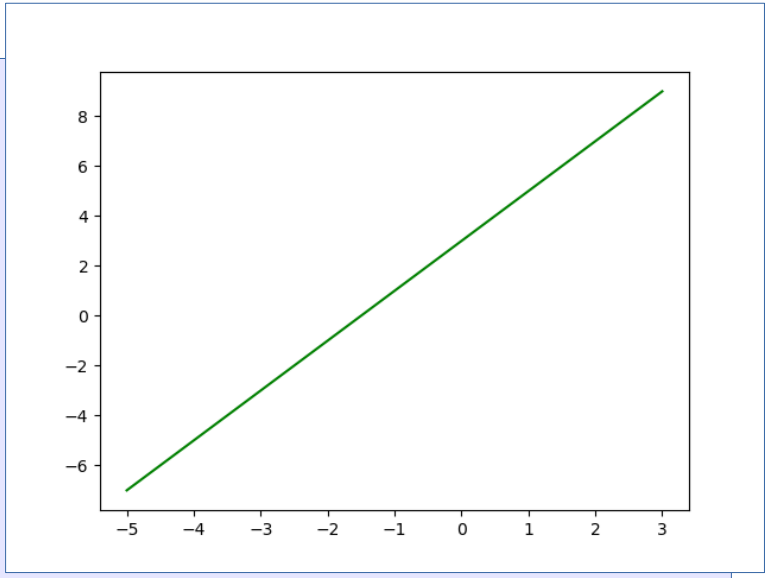
- Escopo de variáveis:
  - **Variável local:** definidas dentro do corpo de uma função;
  - **Variável global:** definidas fora (antes) de uma função:
    - Podem ser acessadas por todo o programa;

```
total_global = 0 #Variável global

def soma(n1, n2):
    global total_global        #Precisa redeclarar a variável com
    total_global = n1 + n2     #global para poder alterá-la

def imprime():                 #Para ler, não é necessário
    print("Total:", total_global)  #redeclarar a variável com global

soma(10, 20)
imprime()
Total: 30
print("Total global:", total_global) #Imprime a variável global
Total global: 30
```

Funções em Python

- Exemplo 1:
  - Função para calcular pontos de um gráfico:

```
import matplotlib.pyplot as pyplot
import numpy as np

def func(x): #f(x) = 2x + 3
    return 2*x + 3

valoresX = list(range(-5, 4))
valoresY = []

for x in valoresX: #calcula o resultado para todos os valores de X
    y = func(x) #chamada da função
    print("({}, {})".format(x, y))
    valoresY.append(y)

pyplot.plot(np.array(valoresX), np.array(valoresY), color='green')
pyplot.show() #mostra o gráfico
```



Funções em Python

- Exemplo 2:
  - Função para sortear um nome de uma lista:
    - Utiliza a biblioteca **random** para gerar um número aleatório.

```
import random

def sorteia_nome(lista):
    n = random.randrange(len(lista)) #Gera um número aleatório
    return lista[n] #Retorna a posição do número sorteado

nomes = ["Joãozinho", "Jhon Travolta", "Frederico",
         "Santiago", "Felisberto"]
sortudo = sorteia_nome(nomes)
print(sortudo)
```

Funções em Python

1) Faça um programa que leia o nome e duas notas de 5 alunos e mostre a média final de cada estudante. O cálculo da média deve ser realizado em uma função.

2) Altere o programa do Exemplo 2 para que calcule os valores de Y através da função $f(x) = x^2 + 2x + 3$.