

# Lógica de Programação Algorítmica

Aula 9 - 18/05/2023



[https://bit.ly/23\\_1\\_logica\\_m](https://bit.ly/23_1_logica_m)



## O que temos para **hoje**?

- O que vimos até agora;
- Situação problema;
- Funções;
- Recursividade;
- Lista de exercícios;



# O que vimos até agora

---



## O que vimos até agora

- ⦿ Entrada de dados;
- ⦿ Declaração de variáveis;
- ⦿ Operações matemáticas;
- ⦿ Estruturas de sequência;
- ⦿ Estruturas de seleção;
- ⦿ Estruturas de repetição;
- ⦿ Funções;
- ⦿ Saída de dados;



# Situação problema



## Situação **problema**

- É muito comum, durante a construção de um programa, termos códigos que se repetem;
- Imagine que você tem um bloco de código que se repete em 5 partes do seu programa;
- Depois que você escreveu o programa, você verificou que esse código repetido possui um erro;



## Situação problema

- Você precisa, então, modificar os 5 locais onde esse código foi repetido para fazer a correção;
- E se, na verdade, não eram 5 locais, mas sim 6 e você esqueceu o último local onde estava o código errado?



# Funções





## Funções

- Função é um nome para um conjunto de comandos que pode ser chamado várias vezes em pontos diferentes do programa, sem a necessidade de repetição de código;

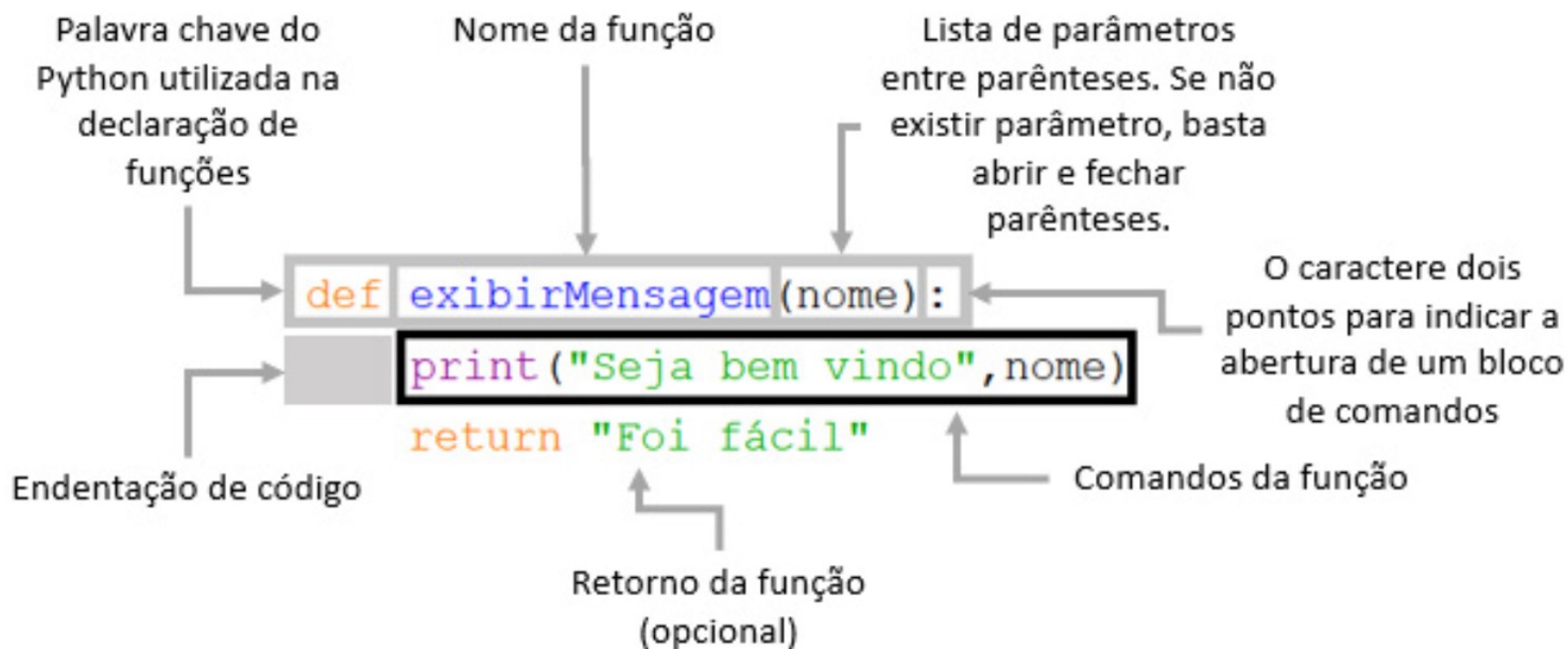


## Funções

- As funções também servem para deixar o código mais fácil de ser lido;



## Funções





## Funções

```
slide12.py > ...  
1  ∨ def exibirMensagem(nome):  
2      print("Seja bem vindo "+nome)  
3      return "Foi fácil"  
-
```



## Funções

○ O que acontece?



## Funções

- As funções precisam ser chamadas;



## Funções

 slide16.py > ...

```
1  ∨ def exibirMensagem(nome):  
2      print("Seja bem vindo "+nome)  
3      return "Foi fácil"  
4  
5  mensagem = exibirMensagem("Fabrício")  
6  print(mensagem)
```



## Funções

- Escreva uma função que receba como parâmetros dois números e escreva o resultado da soma entre eles;





## Funções

 slide17.py > ...

```
1  def soma(x,y):  
2      z = x + y  
3      print(z)  
4  
5  soma(2,3)
```



## Funções

- Escreva uma função que receba como parâmetros dois números e retorne o resultado da soma entre eles;



## Funções

 slide19.py > ...

```
1  def soma(x,y):  
2      z = x + y  
3      return z  
4  
5  resultado = soma(8,9)  
6  print(resultado)
```



## Funções

- Funções importantes que já vimos: input, print, range...



## Funções

- Exercício: crie uma função que receba como parâmetro um número e retorne se esse número é par;



## Funções

 slide22.py > ...

```
1  √ def is_par(n):  
2  √     if n % 2 == 0:  
3      |         return True  
4  √     else:  
5      |         return False  
6  
7  x = input("Digite um número: ")  
8  x = int(x)  
9  print(str(x)+" é par? "+str(is_par(x)))
```



## Funções

- Exercício: crie uma função que receba como parâmetro um número e retorne a soma de todos os números de 0 até esse número (incluindo esse número);



## Funções

slide24.py > ...

```
1  def somatoria(n):
2      soma = 0
3      for i in range(n+1):
4          soma = soma + i
5      return soma
6
7  n = input("Digite um número: ")
8  n = int(n)
9  soma = somatoria(n)
10 print("A somatória de todos os número de 0 até "+str(n)+" é: "+str(soma))
```





## Funções

- Exercício: crie uma função que receba como parâmetro um número e retorne se este número é primo;



## Funções

```
slide26.py > ...
1
2  ∨ def is_primo(n):
3      contador = 2
4  ∨  while contador < n:
5  ∨      if n % contador == 0:
6          return False
7          contador = contador + 1
8
9      return True
10
11  n = input("Digite um número: ")
12  n = int(n)
13  print(str(n)+" é primo? "+str(is_primo(n)))
```



# Recursividade



## Recursividade

- Funções recursivas são funções que chamam a si mesmas de maneira direta ou indireta;
- Não há nenhum benefício em termos de desempenho ao usar funções recursivas em Python, já que laços podem resolver o problema com mais eficiência.



## Recursividade

- Uma função recursiva tem duas partes:
  - Condição de parada: onde a recursão para e os valores são calculados;
  - Condição de recursão: onde a recursão acontece;



## Recursividade

- Exemplo: desenvolva uma função recursiva que calcule o fatorial de um número;



## Recursividade

 slide31.py > ...

```
1  def fatorial(n):
2      if n == 1 or n == 0:
3          return 1
4      return n * fatorial(n - 1)
5  retorno = fatorial(5)
6  print(retorno)
```



# Lista de exercícios





## Lista de **exercícios**

- Acessar o arquivo lista4.pdf no teams da disciplina;
- Resolver as questões;



# Obrigado!

*Alguma pergunta?*

Contato:

☎ 040601692@prof.unama.br