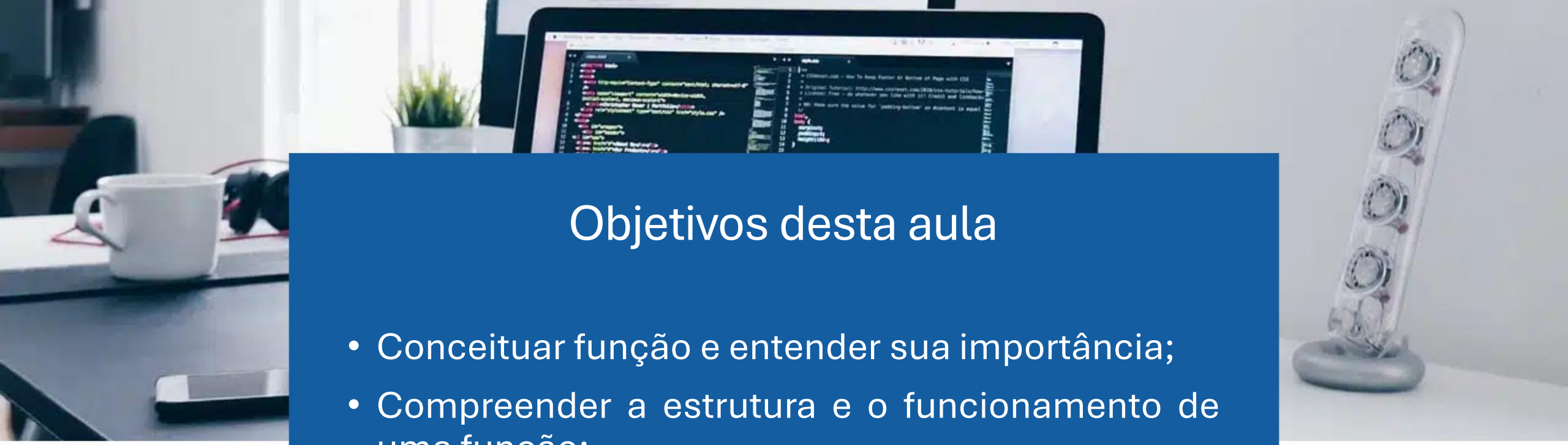
Aula #10:

Funções

Prof. Dr. Luiz Álvaro de Oliveira Júnior



Objetivos desta aula

- Conceituar função e entender sua importância;
- Compreender a estrutura e o funcionamento de uma função;
- Aplicar os conhecimentos de programação na solução de problemas;

Sumário

Funções

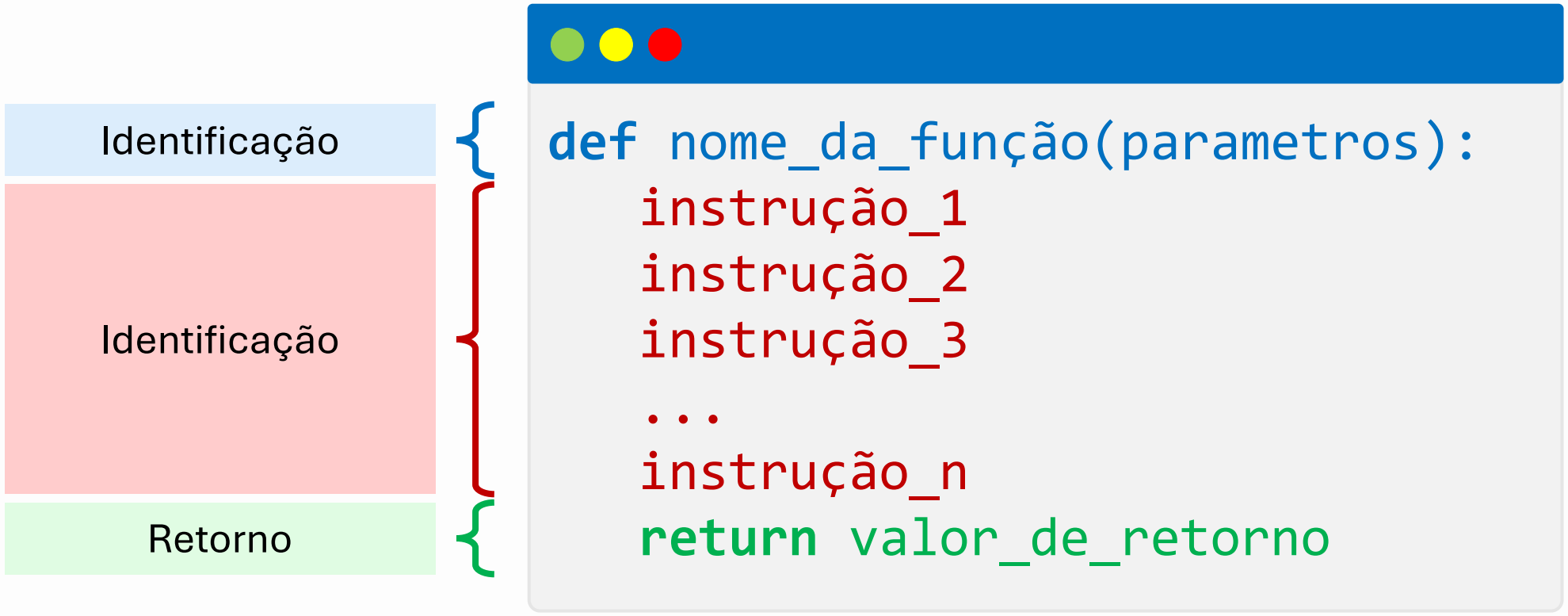
Um **função** é um bloco de código que executa um conjunto de instruções. A função é identificada por um **nome**, pode ou não receber valores de entrada, que chamamos de **parâmetros**, e retorna ou não um resultado.

As funções são utilizadas para modularizar, reutilizar e organizar melhor o código, promovendo clareza e facilitando a manutenção e o reaproveitamento do código.

A declaração de uma função em um código Python é feita usando a palavra reservada **def**.

Funções

A estrutura genérica de uma função é a seguinte:



Funções

Vamos analisar o exemplo abaixo:

```
def soma(x,y):
    x = int(input("Digite x: "))
    y = int(input("Digite y: "))
    soma = x + y
    return soma
```

Identificação da função:
Nome: soma (obrigatório)
Parâmetros: x e y (opcional)

Bloco de código:
Corpo da função (obrigatório).

Retorno:
Valor retornado pela função (opcional).

A palavra reservada **def** deve ser sempre utilizada. Ela indica ao interpretador que ali começa uma função.

Funções

As funções também utilizam indentação para delimitar o bloco de código, e pode haver várias camadas de indentação, dependendo das estruturas presentes no interior da função.

Estruturas de dados como ***strings***, **listas**, **arrays** etc., bem como estruturas de controle como sequenciais, condicionais e repetições, podem compor o corpo de uma função.

São essas estruturas e a lógica que as emprega que determinam o que a função faz e como ela funciona.

Observação: Toda **linha iniciada com def termina com “:”**!

Tipos de funções

As funções podem ser de quatro tipos:

Sem parâmetros e sem retorno: são funções que não recebem parâmetros nem retornam valores.

Sem parâmetros e com retorno: são funções que não recebem parâmetros, mas retornam valores.

Com parâmetros e sem retorno: são funções que recebem parâmetros, mas não retornam valores.

Com parâmetros e com retorno: são funções que recebem parâmetros e retornam valores.

Tipos de funções

Função que não recebe parâmetro e não retorna valor:

```
def mensagem():
    nome = input("Digite seu nome: ")
    texto = "Bom dia"
    print(f"{texto}, {nome}!")
```

Como identificar:

- após o nome da função, os parênteses ficam vazios.
- não aparece a palavra reservada **return**.

Tipos de funções

Função que não recebe parâmetro, mas retorna valor:

```
def mensagem():
    nome = input("Digite seu nome: ")
    frase = "Bom dia, " + nome + "!"
    return frase
```

Como identificar:

- após o nome da função, os parênteses ficam vazios.
- aparece a palavra reservada **return** seguindo-se o nome de uma variável. Neste caso, a função retorna a variável frase

Tipos de funções

Função que recebe parâmetro, mas não retorna valor:

```
def mensagem(texto):
    nome = input("Digite seu nome: ")
    frase = texto + nome + "!"
    print(frase)
```

Como identificar:

- após o nome da função, há pelo menos uma variável dentro dos parênteses.
- não há cláusula **return**.

Tipos de funções

Função que recebe parâmetro e retorna valor:

```
def mensagem(texto):
    nome = input("Digite seu nome: ")
    frase = texto + nome
    return frase
```

Como identificar:

- após o nome da função, há pelo menos uma variável dentro dos parênteses.
- aparece a palavra reservada **return** seguindo-se o nome de uma variável. Neste caso, a função retorna a variável frase

Invocando as funções

Depois de escrever nossas próprias funções nos códigos, para utilizá-las, precisamos **invocar** (ou **chamar**) essas funções, tal como fazemos com as funções que usamos das bibliotecas nativas do próprio Python ou daquelas desenvolvidas por terceiros.

A invocação de uma função depende do tipo de função, isto é, se ela recebe ou não parâmetros e se ela retorna ou não valores.

Para as funções que vimos agora há pouco, vejamos como realizar a invocação e qual é a saída do código.

Invocando as funções

Para invocar uma função que **não recebe parâmetro nem retorna valor**, simplesmente **escrevemos o nome da função com os parênteses**.

```
def mensagem():
    nome = input("Digite seu nome: ")
    texto = "Bom dia"
    print(f"{texto}, {nome}!")

mensagem()
```

Saída

```
Bom dia, Luiz!
```

Invocando as funções

Para invocar uma função que **não recebe parâmetro mas retorna valor**, simplesmente **escrevemos o nome da variável que irá receber o valor e, em seguida, o nome da função.**

```
def mensagem():
    nome = input("Digite seu nome: ")
    frase = "Bom dia, " + nome + "!"
    return frase

texto = mensagem()
print(texto)
```

Saída

```
Bom dia, Luiz!
```

Invocando as funções

Para invocar uma função que **recebe parâmetro, mas não retorna valor**, simplesmente **escrevemos o nome da função e passamos o parâmetro entre os parênteses.**

```
def mensagem(parametro):
    nome = input("Digite seu nome: ")
    frase = parametro + nome + "!"
    print(frase)

texto = "Bom dia, "
mensagem(texto)
```

Saída

```
Bom dia, Luiz!
```

Invocando as funções

Para invocar uma função que **recebe parâmetro e retorna valor**, simplesmente **escrevemos o nome da variável que irá receber o valor, em seguida o nome da função e, dentro dos parênteses, o parâmetro.**

```
def mensagem(parametro):
    nome = input("Digite seu nome: ")
    frase = parametro + nome + "!"
    return frase

texto = "Bom dia, "
variavel = mensagem(texto)
```

Saída

```
Bom dia, Luiz!
```

Invocando as funções

Funções podem, ainda, retornar múltiplos valores. Neste caso, após a palavra reservada **return**, separamos por vírgulas as variáveis que serão retornadas. O **número de variáveis usados para receber deve ser igual ao número de valores retornados**.

```
def AreaPerimetro(b,h):
    area = b * h
    perimetro = (b + h) * 2
    return area, perimetro

A,P = AreaPerimetro(8,4)
print(f"Área: {A} m2\nPerímetro: {P} m")
```

Saída

```
Área: 32 m2
Perímetro: 24 m
```

Funções com variáveis globais

Da **Aula 2**, vamos lembrar o que são variáveis **locais** e **globais**:

Variáveis locais são aquelas que são **criadas dentro de funções** e **existem** localmente **somente nelas**, não sendo possível acessar essas variáveis externamente.

Variáveis globais são aquelas que **são criadas fora das funções** e podem ser **usadas em qualquer parte do código**. **Por padrão, toda variável criada fora de uma função é global e essa definição é implícita**.

Funções com variáveis globais

As funções não podem alterar valores das variáveis globais (externas a elas) exceto se tais variáveis forem listadas explicitamente na função como variáveis globais.

Para fazer isso, é necessário usar a palavra reservada **global** antes de listar as variáveis globais que a função pretende alterar. Sem a palavra reservada "**global**", o Python cria cópias locais dessas variáveis na função, o que pode causar erros ou comportamentos inesperados.

Vejamos um exemplo:

```
def AreaPerimetro(b, h):
    global cont
    cont += 1
    area = b * h
    perimetro = 2 * (b + h)
    return area, perimetro

cont = 0
n = int(input("Quantos cálculos quer fazer? "))

for i in range(n):
    b = float(input("Digite a base em m: "))
    h = float(input("Digite a altura em m: ") )
    A, P = AreaPerimetro(b, h)
    print(f"Área: {A} m2\nPerímetro: {P} m.")

print(f"Função invocada {cont} vez(es).")
```

Exercícios

Exercício #1

Escreva um programa em Python que peça a idade de uma pessoa e, em seguida, invoque uma função para avaliar se a pessoa pode dirigir ou não.

```
def CNH(x):
    if idade >= 18:
        print("Você já pode dirigir")
    else:
        print("Você é muito novo(a) para dirigir")

idade = int(input("Digite a idade: "))
CNH(idade)
```

Exercício #2

Escreva um programa que peça ao usuário dois números e, em seguida, pergunte qual operação o usuário deseja realizar: adição, multiplicação ou subtração. Cada operação deve ser realizada por uma função distinta.

As funções deverão receber parâmetros e retornar valor.

```
def adicao(x,y):
    return x + y

def subtracao(x,y):
    return x - y

def multiplicacao(x,y):
    return x * y

a,b = float(input("Digite a: ")), float(input("Digite b: "))
resp = input("Digite + para somar ou * para multiplicar: ")

if resp == "+":
    soma = adicao(a,b)
elif resp == "-":
    produto = subtracao(a,b)
elif resp == "*":
    produto = multiplicação(a,b)
else:
    print("Opção inválida.")
```

Exercício #3

Escreva um programa em Python para calcular o momento de uma força em relação a um ponto. Para isto, seu programa deve:

- Solicitar a intensidade da força;
- Chamar uma função para calcular o vetor direção a partir das coordenadas de dois pontos na linha de ação dessa força;
- Chamar uma função para calcular o vetor força;
- Chamar uma função para calcular o vetor posição;
- Chamar uma função para calcular o momento.