

# PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática  
Lucas Sampaio Leite



## Motivação...

- Imagine que você está desenvolvendo um sistema para calcular descontos em compras de clientes.
- No início, você escreve algo assim:

```
valor = 250
desconto = valor * 0.1
total = valor - desconto
print("Total a pagar (Cliente 1):", total)
```

## Motivação...

- Imagine agora que o mesmo cálculo precisa ser repetido para cada cliente.
- Cada cliente pode receber taxas de desconto diferentes de acordo com o método de pagamento utilizado (ex.: 15% em vez de 10% para compras a vista).
- Em sistemas maiores, o desconto pode ser calculado em vários pontos diferentes do sistema, aumentando a chance de erros.

# Motivação...

- Pergunta para reflexão:
  - Como podemos resolver isso de forma mais organizada, reutilizável e fácil de manter?



## Motivação...

- Pergunta para reflexão:
  - Como podemos resolver isso de forma mais organizada, reutilizável e fácil de manter?



Solução: criar uma função que encapsule o cálculo do desconto.

# Funções

- Uma função é um bloco nomeado de instruções que executa uma operação específica.
- Ela é definida a partir de um identificador (nome) e de um conjunto de instruções que a compõem, podendo ser reutilizada posteriormente sempre que necessário.
- As funções permitem dividir o código em partes menores e mais organizadas, o que aumenta a legibilidade, facilita a manutenção e contribui para o reaproveitamento do código.



# Funções

- Dividir para Conquistar:
  - Estratégia muito usada na resolução de problemas.
  - Consiste em quebrar um problema complexo em subproblemas menores e mais simples.
  - Cada subproblema é resolvido de forma independente.
  - As soluções dos subproblemas são então combinadas para resolver o problema original.

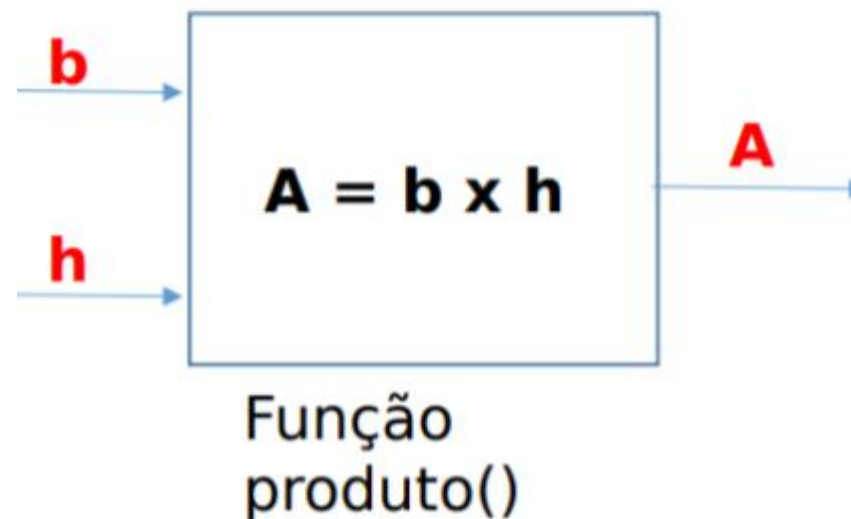
# Funções

- Em programação, usamos funções para aplicar essa mesma ideia:
  - Cada função resolve um subproblema específico.
  - O programa principal combina essas funções para resolver o problema completo.
  - Resultado: código mais organizado, reutilizável e fácil de manter.



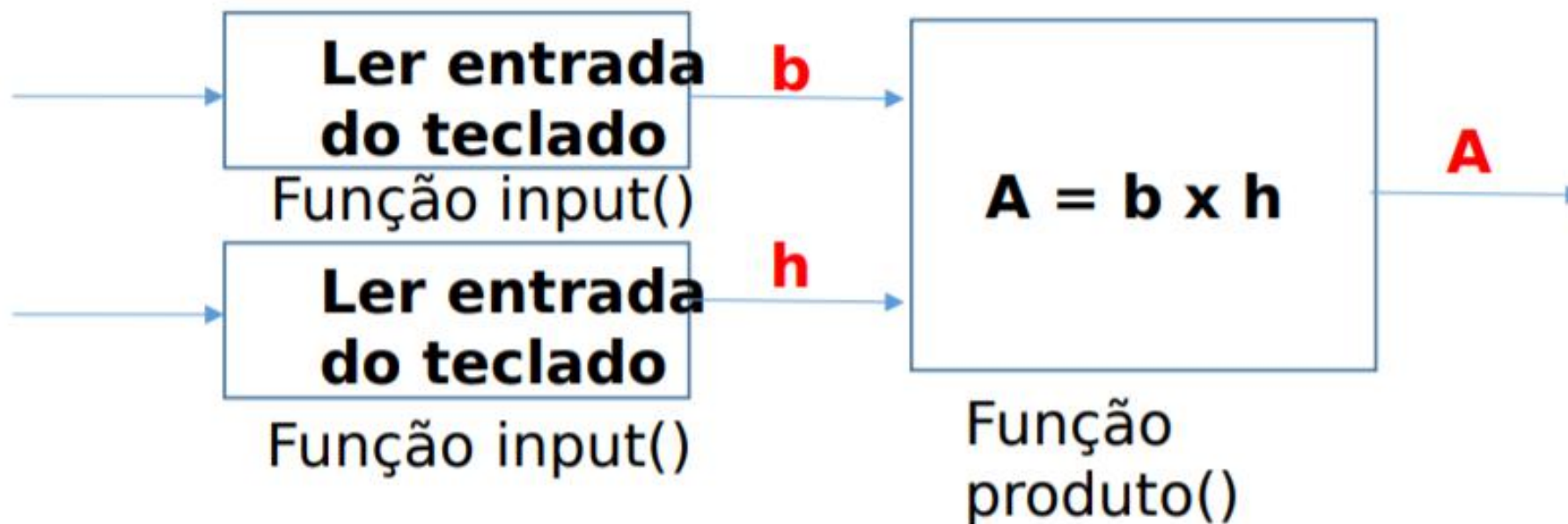
# Funções

- Uma mesma função pode ser reutilizada em diferentes contextos.
- Exemplo:
  - Uma função de multiplicação pode ser usada tanto para calcular a área de um retângulo quanto a de um triângulo.
  - Em ambos os casos, a solução do problema envolve a mesma operação: multiplicar valores.



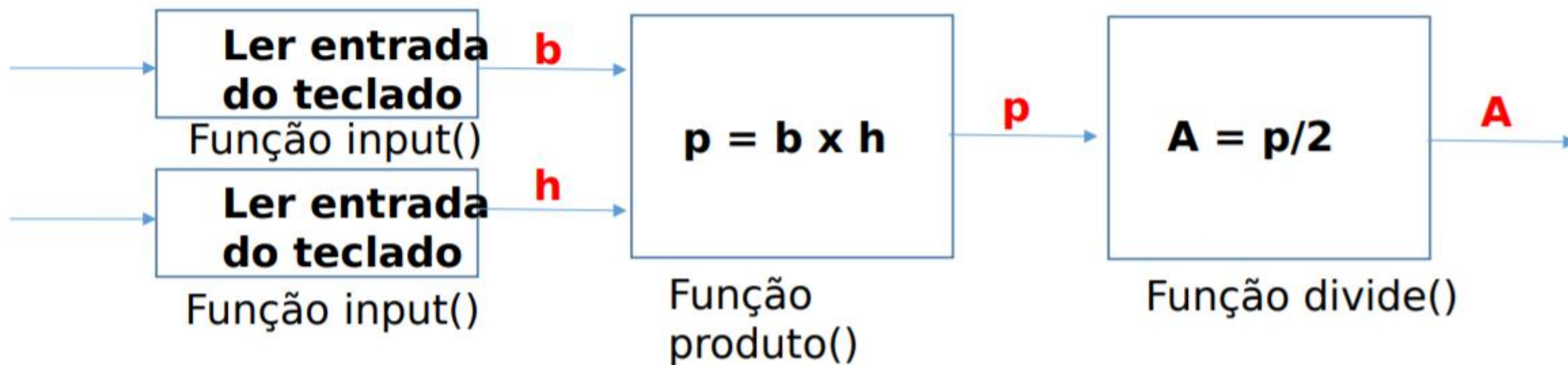
# Funções

- Exemplo do uso da função produto para o cálculo da área de um retângulo:



# Funções

- Exemplo do uso da função produto para cálculo da área de um triângulo:



# Funções

- `def` → palavra-chave usada para definir uma nova função.
- Identificador (nome da função):
  - Dá nome à função.
  - Usado para invocar a função no código.
  - Executa um conjunto de instruções para resolver parte do problema.
- Parâmetros/argumentos (entre parênteses):
  - Representam os valores de entrada da função.
  - Uma função pode ter nenhum, um ou vários parâmetros.

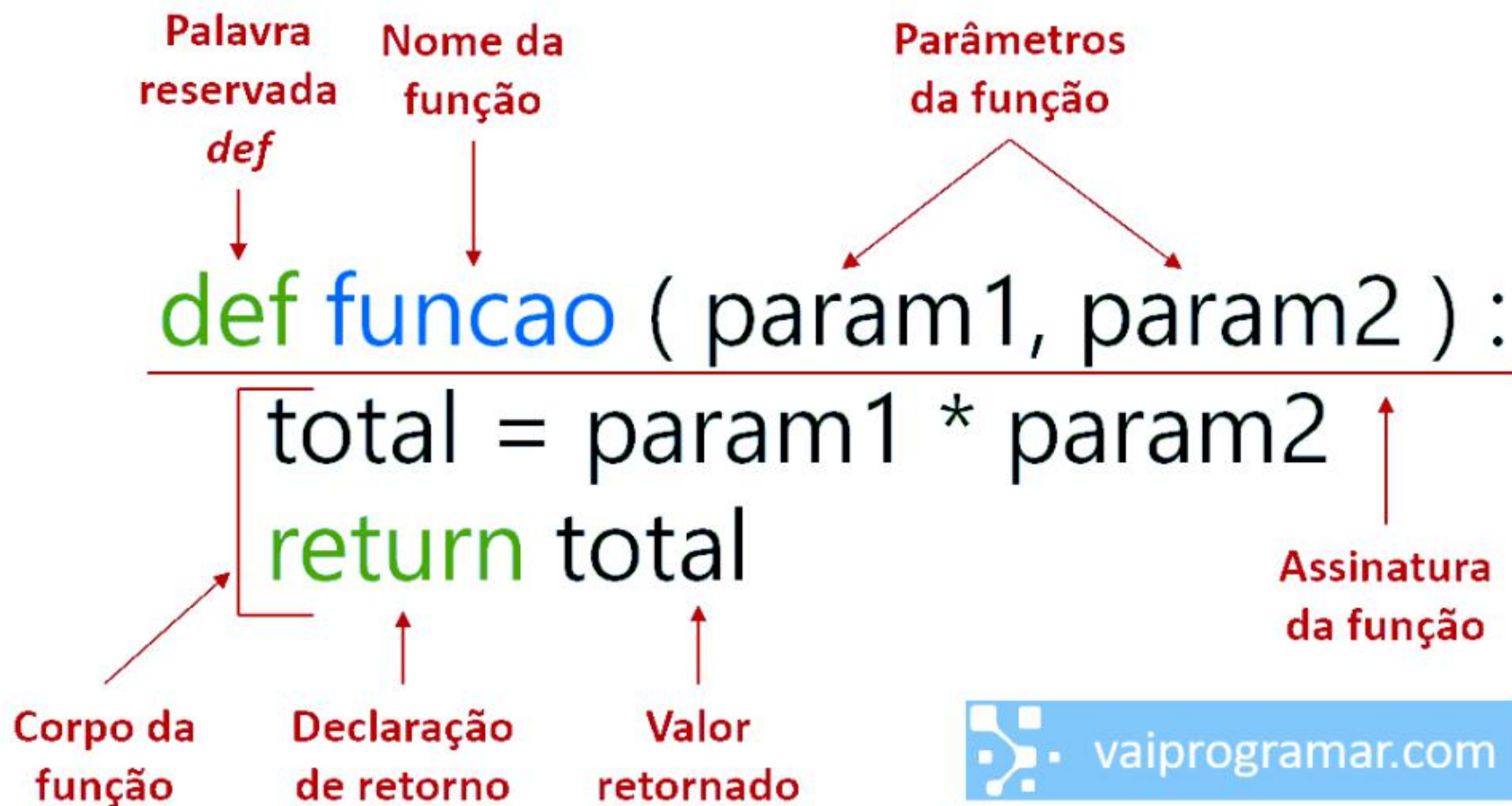
```
1  def identificador(parametros):  
2      #corpo da funcao  
3      #o corpo da funcao pode ter várias linhas  
4      #a indentação irá definir o corpo da funcao  
5      return #Uma função pode ter ou não retorno  
6  #este trecho está fora da função
```

# Funções

```
1  def identificador(parametros):
2      #corpo da funcao
3      #o corpo da funcao pode ter várias linhas
4      #a indentação irá definir o corpo da funcao
5      return #Uma função pode ter ou não retorno
6  #este trecho está fora da função
```

- Corpo da função:
  - Conjunto de instruções que define o que a função faz.
  - Pode conter uma ou várias linhas de código.
  - Dentro dele, podemos usar o comando return.
- return:
  - Indica o resultado produzido pela função.
  - Nem toda função precisa retornar um valor:
    - return None (retorno explícito sem valor).
    - return (encerra a função sem valor).
    - Ou simplesmente não usar return.

## Como Declarar uma Função em Python



# Funções

- Exemplo de função sem parâmetros e sem retorno:

```
def saudacao():  
    print("Olá, seja bem-vindo!")  
  
saudacao()
```



```
Olá, seja bem-vindo!
```



# Funções

- Exemplo com parâmetros, mas sem retorno:

```
def saudacao_pessoa(nome):  
    print("Olá, ", nome, "seja bem-vindo!")  
  
saudacao_pessoa("Lucas")
```



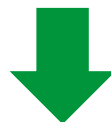
```
Olá, Lucas seja bem-vindo!
```



# Funções

- Exemplo com parâmetros e com retorno:

```
def soma(a, b):  
    return a + b  
  
resultado = soma(5, 3)  
print("Resultado da soma:", resultado)
```



Resultado da soma: 8

# Funções

- Exemplo sem retorno explícito (None):

```
def mostrar_dobro(n):  
    print("O dobro é:", n * 2)  
  
valor = mostrar_dobro(10)  
print("Valor retornado:", valor)
```



```
O dobro é: 20  
Valor retornado: None
```

# Funções

- Exemplo sem retorno explícito (None):

```
def mostrar_dobro(n):  
    print("O dobro é:", n * 2)  
  
valor = mostrar_dobro(10)  
print("Valor retornado:", valor)
```



```
O dobro é: 20  
Valor retornado: None
```

A função imprime diretamente o resultado.

Isso reduz a reutilização, porque o usuário não consegue facilmente usar o valor em outro cálculo.

# Funções

- Exemplo sem retorno explícito (None):

```
def mostrar_dobro(n):  
    print("O dobro é:", n * 2)  
  
valor = mostrar_dobro(10)  
print("Valor retornado:", valor)
```

Ao tentar capturar o valor retornado (valor), não há retorno explícito, então valor é None.

O dobro é: 20  
Valor retornado: None

# Funções

- Utilizando uma abordagem mais adequada:

```
def dobro(n):  
    return n * 2  
  
resultado = dobro(10)  
print("O dobro é:", resultado)
```



```
O dobro é: 20
```

# Funções

- Exemplo com mais de um parâmetro

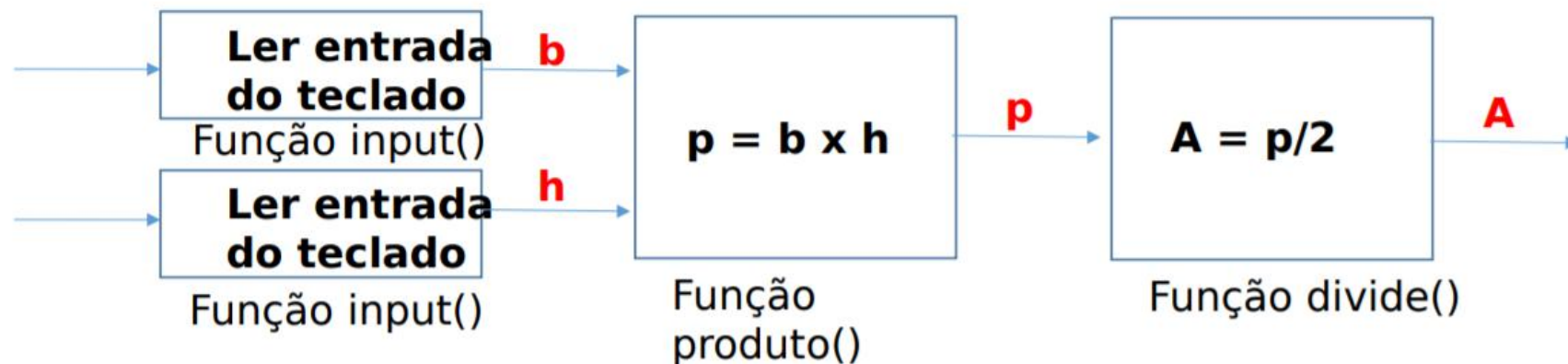
```
def calcular_imc(peso, altura):  
    return peso / (altura ** 2)  
  
print("IMC:", calcular_imc(70, 1.75))
```



```
IMC: 22.857142857142858
```



# Funções



```
def produto(valor_1, valor_2):  
    resultado = valor_1 * valor_2  
    return resultado  
  
def divisao(dividendo, divisor):  
    if divisor == 0:  
        return  
    resultado = dividendo / divisor  
    return resultado
```

```
base = float(input("Digite o valor da base: "))  
altura = float(input("Digite o valor da altura: "))  
resultado = produto(base, altura)  
area = divisao(resultado, 2)
```

# Funções

```
def calcular_pagamento(qtd_horas, valor_hora):  
    if qtd_horas <= 40:  
        salario = qtd_horas * valor_hora  
    else:  
        hora_excedente = qtd_horas - 40  
        salario = 40 * valor_hora + (hora_excedente * (1.5 * valor_hora))  
    return salario  
  
resultado = calcular_pagamento(20, 100)  
print(resultado)  
resultado = calcular_pagamento(40, 100)  
print(resultado)  
resultado = calcular_pagamento(60, 100)  
print(resultado)
```

O que a função faz?


Qual a saída do programa?



# Funções

```
def calcular_pagamento(qtd_horas, valor_hora):  
    if qtd_horas <= 40:  
        salario = qtd_horas * valor_hora  
    else:  
        hora_excedente = qtd_horas - 40  
        salario = 40 * valor_hora + (hora_excedente * (1.5 * valor_hora))  
    return salario
```

```
resultado = calcular_pagamento(20, 100)  
print(resultado)  
resultado = calcular_pagamento(40, 100)  
print(resultado)  
resultado = calcular_pagamento(60, 100)  
print(resultado)
```



```
2000  
4000  
7000.0
```

# Exercícios

1. Crie um programa em Python que defina uma função capaz de receber três argumentos numéricos e retornar a soma desses três valores. No programa principal, chame essa função e exiba o resultado obtido.
2. Crie um programa em Python que defina uma função que receba um único argumento. A função deve retornar o caractere 'P' se o argumento for positivo e 'N' se o argumento for zero ou negativo. No programa principal, chame a função e exiba o resultado correspondente.
3. Crie um programa em Python que defina uma função capaz de receber um número inteiro como argumento e retornar o seu reverso. Por exemplo, se o número informado for 127, a função deve retornar 721. No programa principal, chame a função e exiba o resultado.

## Exercícios

4. Crie um programa em Python que solicite um número inteiro  $n$  ao usuário e utilize uma função para imprimir um padrão de acordo com a imagem fornecida. A função deve receber o valor  $n$  como argumento e imprimir o padrão até a  $n$ -ésima linha.

```
1
2  2
3  3  3
   . . . . .
n  n  n  n  n  n  . . . n
```

5. Crie um programa em Python que defina uma função capaz de calcular a potência  $a^b$  para valores inteiros de  $a$  e  $b$  informados pelo usuário. A função não deve usar o operador `**` nem funções da biblioteca `math`. Em seguida, chame a função e exiba o resultado.

# Exercícios

6. Crie um programa em Python que defina uma função capaz de retornar a quantidade de dígitos de um número inteiro informado pelo usuário. A função deve receber o número inteiro como argumento e não deve realizar conversão de tipos (por exemplo, para string). Chame a função e exiba o resultado.
7. DESAFIO: Crie um programa em Python com uma função para converter um número inteiro informado pelo usuário em seu equivalente na numeração romana. A função deve receber o número como argumento e retornar a representação romana correspondente. Solicite o número ao usuário, chame a função e exiba o resultado.

# Dúvidas



# PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática  
Lucas Sampaio Leite

