Laboratório de Programação em Python

Prof. Alisson Alves
Mestre em Ciência da Computação
(UFMG)

python™

SENAI

Agenda

- Funções Básicas do Software;
- Relações entre áreas;
- Introdução à Algoritmos;
- Introdução à Lógica;
- Introdução ao Visual Studio Code (VSCode)
- Introdução ao Python

python™

Visual Studio Code

SENAI

Entrada, Saída e Processamento

FUNÇÕES BÁSICAS DO SOFTWARE

python™

SENAI

Função do Software

1. O que faz um *software*?
   - **Processamento**!
2. O que um *software* precisa para processar?
   - Informações ou Dados!
3. Qual é o resultado de um processamento?
   - Informações, Dados ou Conhecimento!

python™

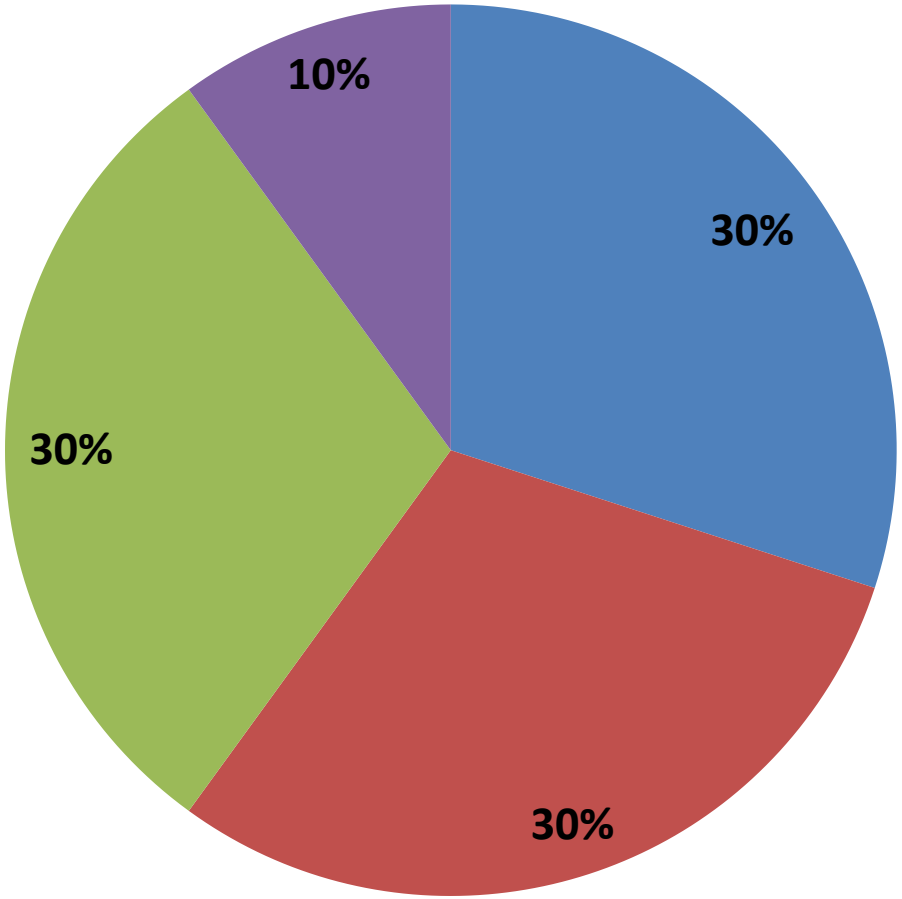
SENAI

RELAÇÕES ENTRE AS ÁREAS

python™

SENAI

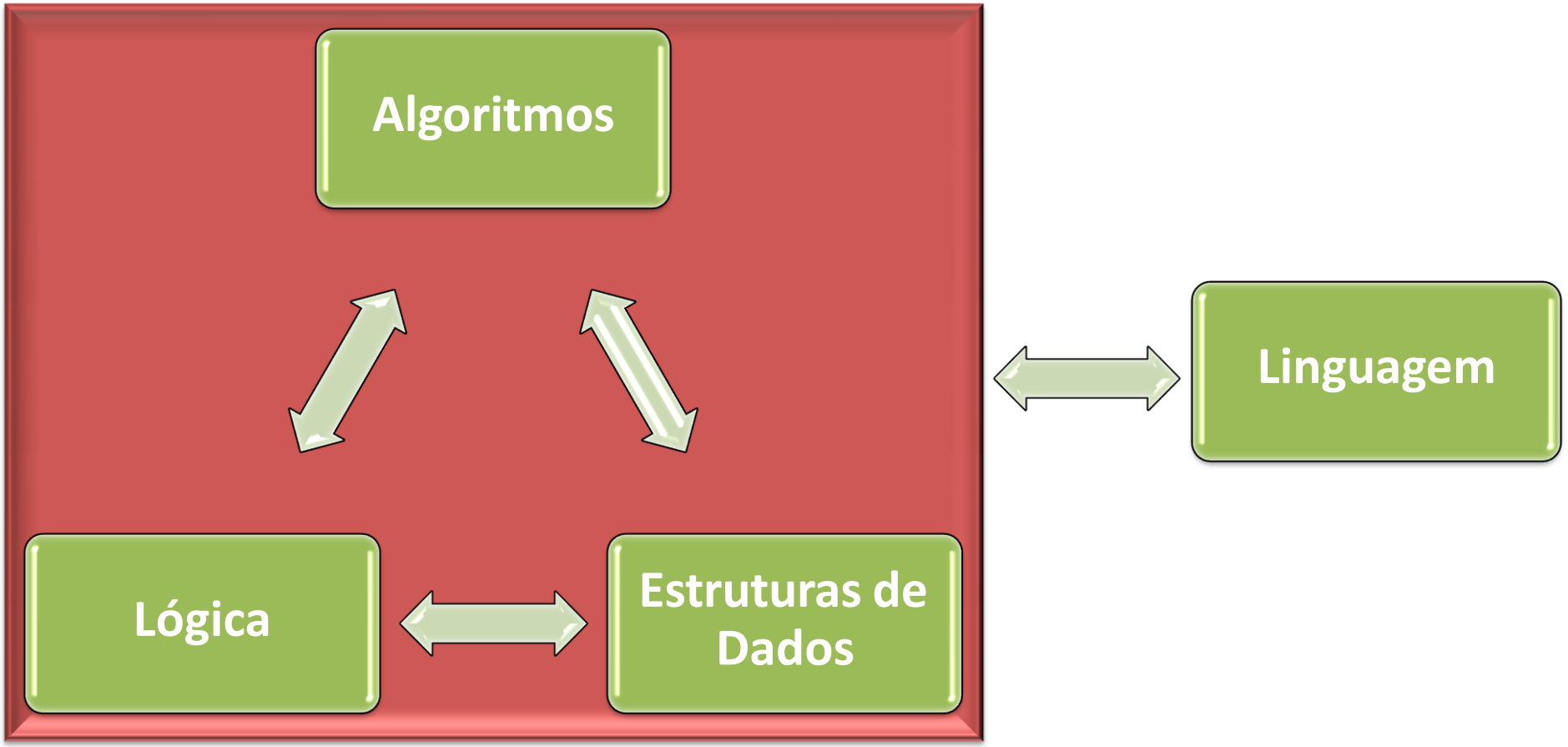
Composição do Software



python™

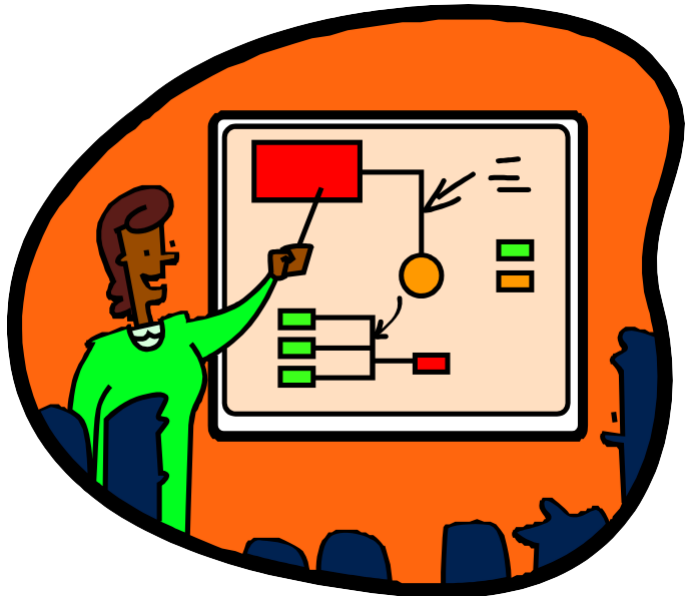
SENAI

Relações entre as Partes



python™

SENAI



INTRODUÇÃO À ALGORITMOS

python™

SENAI

Algoritmos

- Forma de descrever uma tarefa, função ou programa de computador;
- Pode ser representado por pseudocódigo ou fluxograma;
- Representa uma **sequência de passos finitos** e bem definidos que o computador deve executar a fim de **atingir** ou **obter** um resultado.

python™

SENAI

Algoritmos

- Informal: *"Procedimento computacional bem definido que toma algum valor ou conjunto de valores como **entrada** e produz algum valor ou conjunto de valores como **saída**".*
  - Cormen, T. H. Algoritmos – Teoria e Prática, 2001.

python™

SENAI

Algoritmo – Passar Café

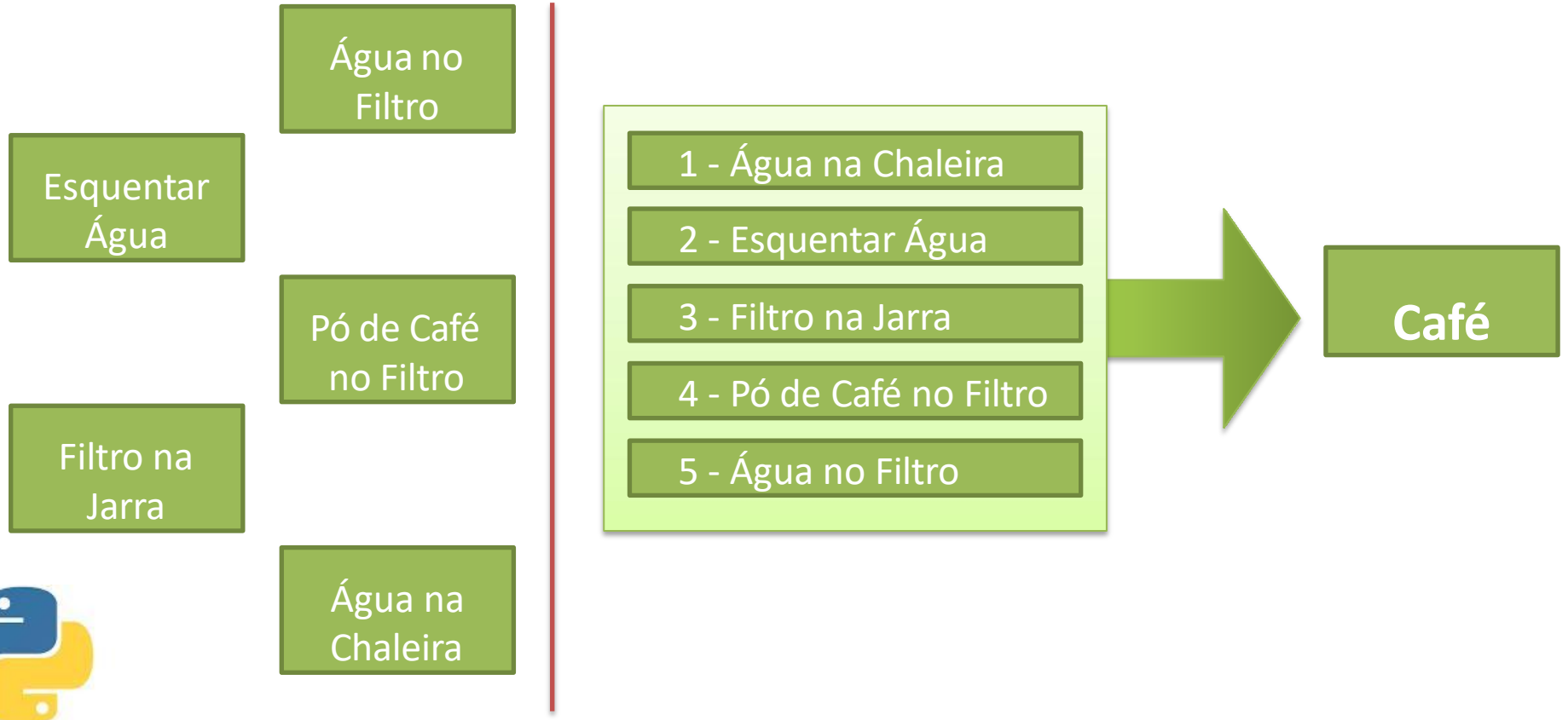
- Instruções Simples para “Passar Café”: Quais?

python™

SENAI

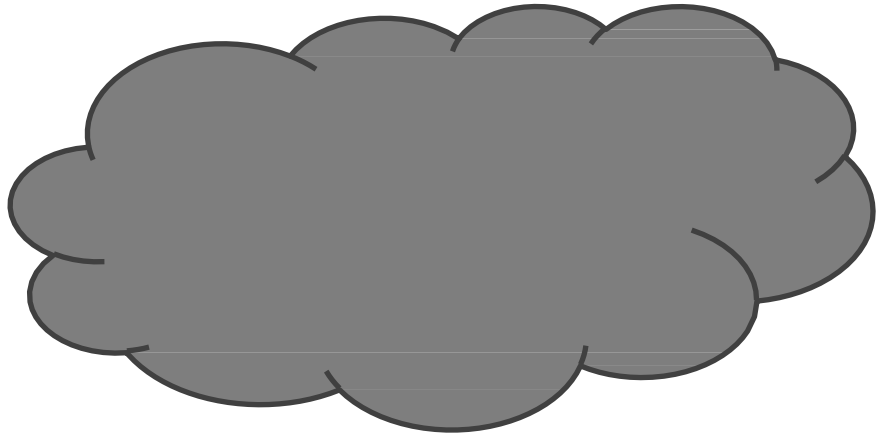
Algoritmo – Passar Café

- Algoritmo: “Passar Café”:



python™

SENAI



INTRODUÇÃO À LÓGICA

python™

SENAI

Lógica de Programação

"Lógica de programação é a técnica de encadear pensamentos para atingir determinado objetivo"

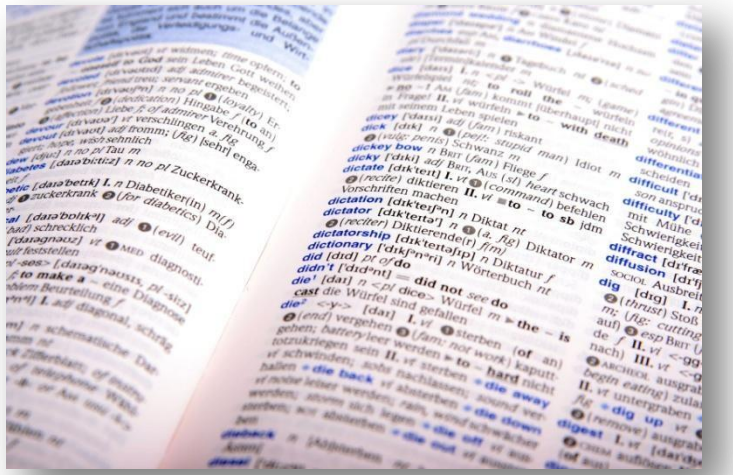
Paulo Sérgio de Moraes – Unicamp

- Importância?
  - Requisito para programar;
  - Grande diferencial entre os programadores;
  - Reconhecer, interpretar e estruturar o problema:
    - Solucionar o problema;

SENAI

Lógica de Programação

- Definições:
  - Instruções: (algoritmo)
    - Comandos ou ações;
    - Atividade parcial;
  - Sequência Lógica: (lógica)
    - Ordem das instruções;
    - Completar a atividade.

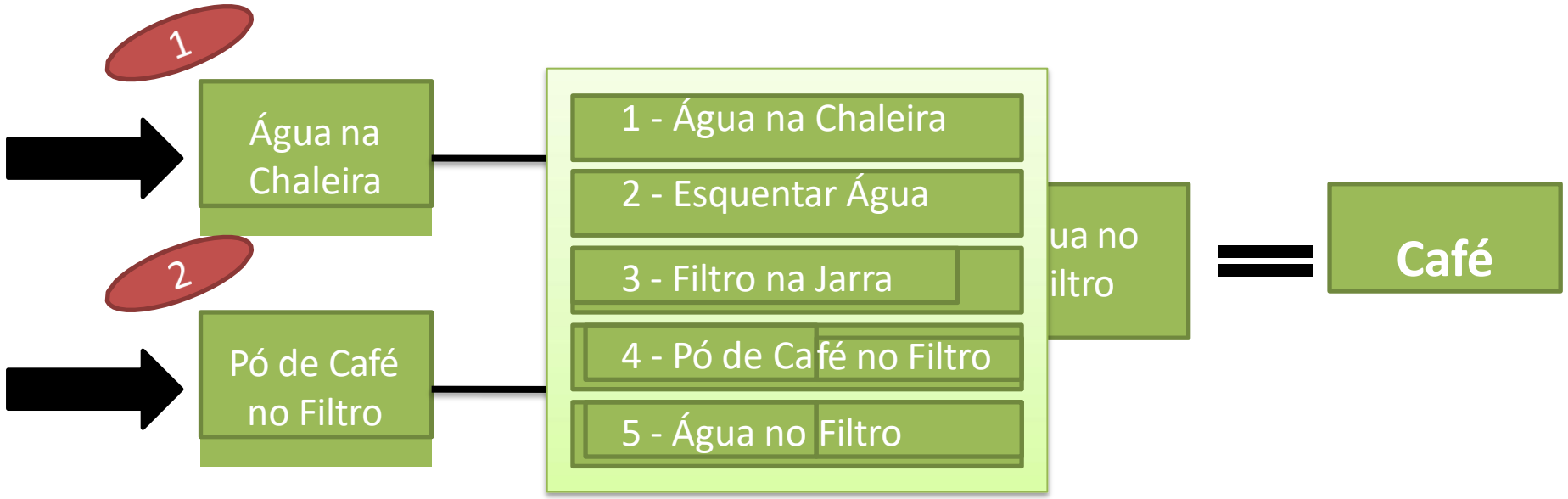


python™

SENAI

Lógica de Programação

- Sequência Lógica Simples para "*Passar Café*":

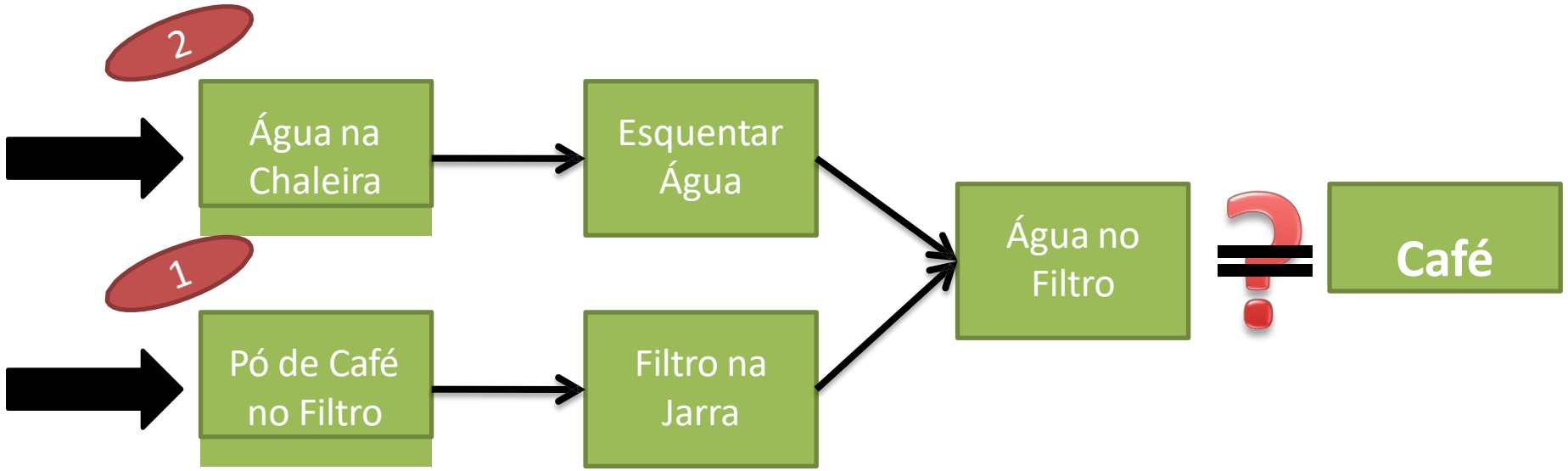


python™

SENAI

Lógica de Programação

- Sequência Lógica Simples para "*Passar Café*":

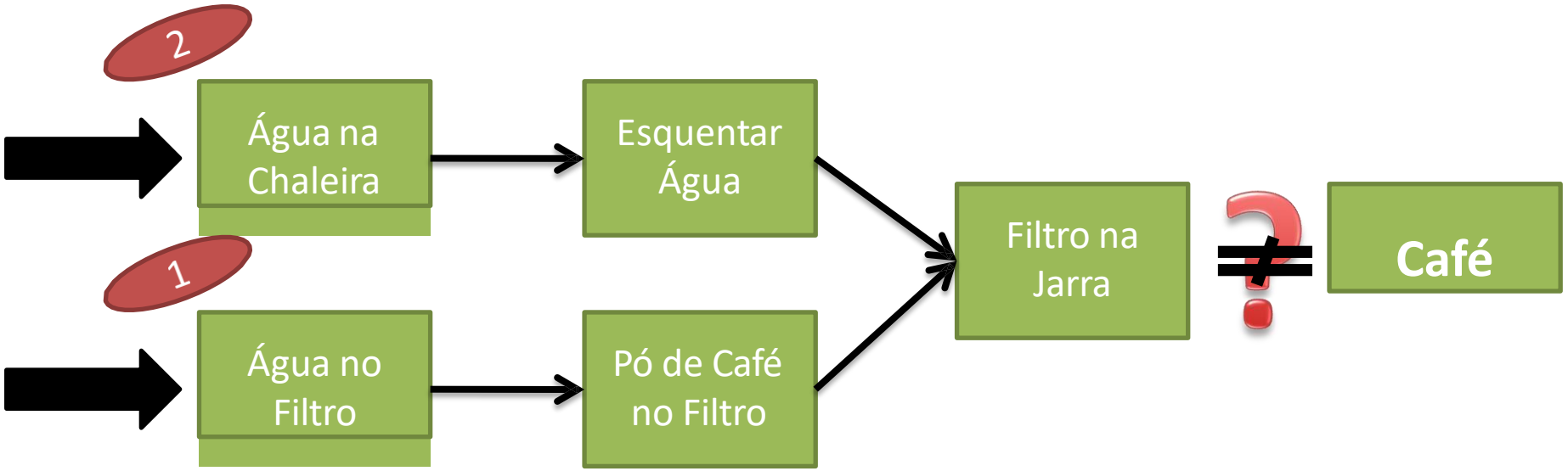


python™

SENAI

Lógica de Programação

- Sequência Lógica Simples para "*Passar Café*":



Funções de um programador

- Entender perfeitamente o problema;
- Escolher métodos para sua solução;
- Desenvolver um algoritmo baseado nos métodos;
- Codificar o algoritmo na linguagem de programação disponível.

SENAI

Dados na Computação

- O que é, ou pode ser, um “dado”?
  - É um número;
  - É um nome;
  - É um endereço;
  - É o valor de um produto;
  - É um *pixel* de uma imagem;
  - Pode ser muitas coisas...

python™

41 PEDRO

SENAI

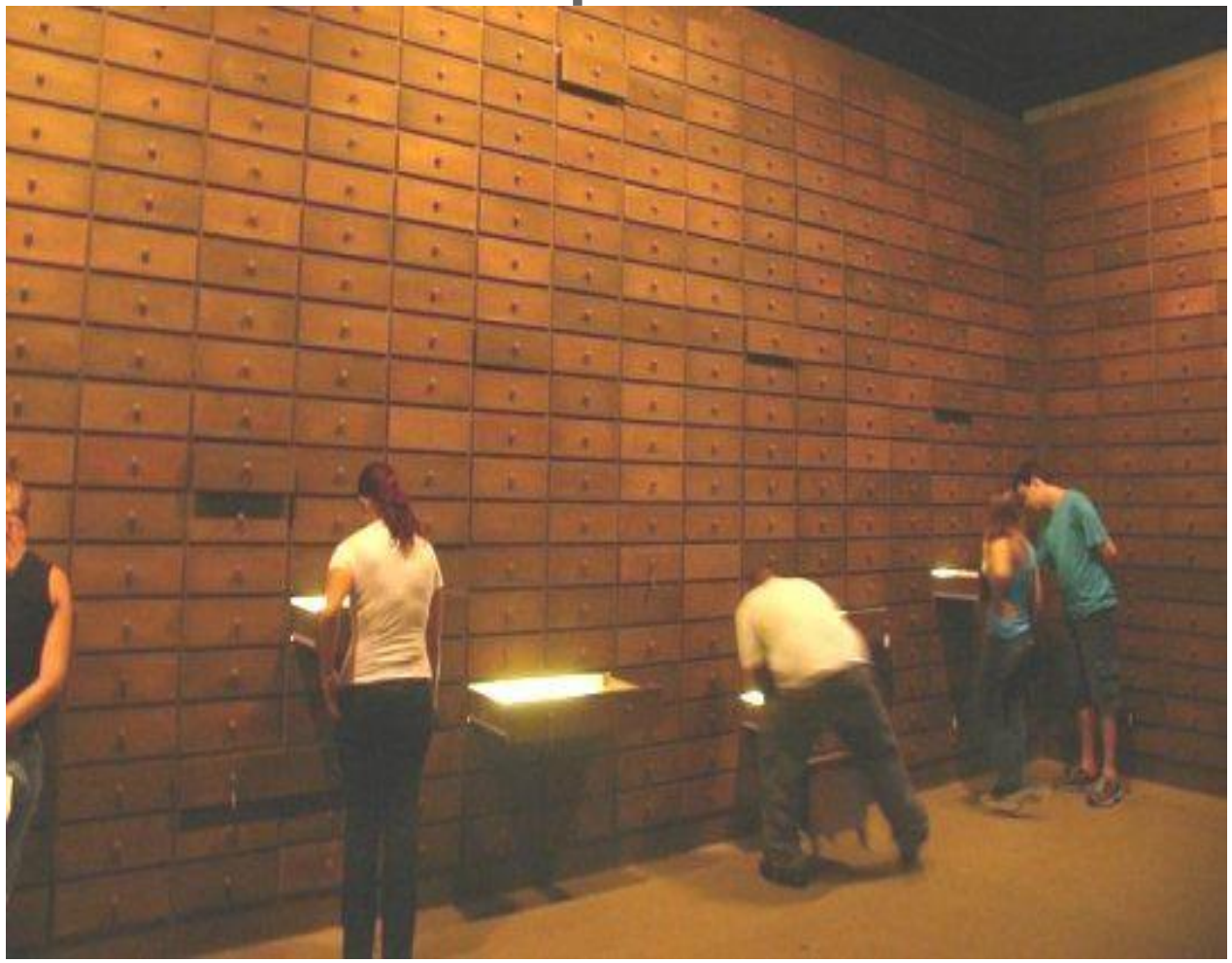
Como Armazenar Dados?

- Dados são representados em pequenas estruturas e em geral são chamados de **variáveis** ou **constantes**!

- Tudo que é armazenado dentro do computador, permanece em alguma **memória**!

SENAI

A Memória do Computador



Como Armazenar Dados?

- **Variáveis**: armazenam dados em **caráter temporário** e tem conteúdo dinâmico, ou seja, podem ser acessadas ou alteradas a qualquer momento;
- **Constantes**: são dados estáticos, o seu conteúdo pode ser acessado a qualquer momento, mas será definido no momento de sua criação e **não poderá** ser mais alterado. Ex: **PI** = 3.14159 ou **GRAVIDADE** = 9.8

Introdução ao Python

- Criado por **Guido van Rossum** no final dos anos 80 e lançado em **1991** (versão 0.9.0).
- Nome inspirado no grupo de comédia britânico **Monty Python**.
- Projetado para ser uma linguagem de **fácil leitura e escrita**, com **sintaxe clara e concisa**.
- Evoluiu ao longo dos anos e se tornou **uma das linguagens mais populares do mundo**.

📌 *Simples, poderoso e versátil!*

SENAI

Características

▼ Legibilidade

Python utiliza uma sintaxe clara e simples, o que facilita a leitura e compreensão do código. Utiliza indentação (espaços ou tabulações) para delimitar blocos de código, o que promove um estilo de programação estruturado e legível.

▼ Tipagem dinâmica

Em Python, não é necessário declarar explicitamente o tipo de dados das variáveis. Python infere automaticamente o tipo de dados com base no valor atribuído a uma variável, o que simplifica a escrita de código.

▼ Multiplataforma

Python pode ser executado em diferentes sistemas operacionais, como Windows, macOS e Linux, sem necessidade de modificar o código. Isso o torna uma linguagem versátil e portátil.

▼ Ampla biblioteca padrão

Python vem com uma extensa biblioteca padrão que fornece uma grande quantidade de módulos e funções para realizar diversas tarefas, como manipulação de arquivos, conexão a bancos de dados, processamento de texto, entre outros.

Características

▼ Interpretado

Python é uma linguagem interpretada, o que significa que o código é executado linha por linha em tempo real. Isso permite um ciclo de desenvolvimento rápido e facilita a depuração do código.

▼ Comunidade ativa

Python conta com uma comunidade de desenvolvedores grande e ativa que contribui com bibliotecas, *frameworks* e ferramentas adicionais. Isso significa que você encontrará uma grande quantidade de recursos e suporte disponíveis.

Aplicações

Ciência de dados

Python é a linguagem preferida para análise de dados e ciência de dados devido a bibliotecas como NumPy, Pandas e Matplotlib.

Automatização de tarefas

Python pode ser utilizado para automatizar tarefas repetitivas, como processamento de arquivos, web *scraping* e testes de *software*.

Desenvolvimento web

Python é amplamente utilizado no desenvolvimento web *backend*, com *frameworks* populares como Django e Flask.

Inteligência artificial e *machine learning*

Python é a escolha principal para projetos de IA e *machine learning*, graças a bibliotecas como TensorFlow e Scikit-learn.

Desenvolvimento de jogos

Python é utilizado no desenvolvimento de jogos simples, especialmente com bibliotecas como Pygame.

SENAI

Baixar e instalar Python

Para começar a programar em Python, primeiro você deve baixar e instalar Python no seu computador. Siga estes passos:

1. Vá ao site oficial do Python: https://www.python.org/downloads/Links to an external site.
2. Na seção "Download" você encontrará a última versão estável do Python. Selecione a versão adequada para o seu sistema operacional (Windows, macOS ou Linux).
3. Baixe o instalador do Python correspondente ao seu sistema operacional.
4. Uma vez baixado, execute o instalador. Certifique-se de marcar a opção "Add Python to PATH" durante o processo de instalação no Windows. Isso permitirá que você execute Python a partir da linha de comando.
5. Siga as instruções do instalador e espere a instalação ser concluída.

Seu primeiro programa Python

- Abra seu IDE ou editor de texto preferido e crie um novo arquivo.
- Nomeie o arquivo como "ola_mundo.py". A extensão ".py" indica que é um arquivo de Python.
- No arquivo, escreva o seguinte código:

```
print ("Olá, Mundo!")
```

- Salve o arquivo e execute o programa. Se estiver utilizando um IDE, procure a opção "Run" ou "Execute".

Você verá que a mensagem "Olá, Mundo!" é impressa na tela.

Conceitos básicos da sintaxe em Python

Indentação

No Python, a indentação (espaços ou tabulações no início de uma linha) é utilizada para delimitar blocos de código. Diferente de outras linguagens que utilizam chaves ou palavras-chave, o Python utiliza a indentação para determinar o escopo das declarações. Por exemplo:

```
if condition:

    # Bloco de código se a condição for verdadeira

    instrucao1
    instrucao2

else:

    # Bloco de código se a condição for falsa

    instrucao3
    instrucao4
```

SENAI

Importante

É fundamental manter uma indentação consistente em todo o código para evitar erros de sintaxe.

Conceitos básicos da sintaxe em Python

Comentários

Os comentários são linhas de texto no código que são ignoradas pelo interpretador do Python. Eles são utilizados para explicar ou documentar o código. No Python, os comentários de uma única linha começam com o símbolo #, enquanto os comentários de várias linhas são delimitados por três aspas """ . Por exemplo:

```
# Este é um comentário de uma única linha

"""
Este é um comentário
de várias linhas
"""
```

SENAI

Conceitos básicos da sintaxe em Python

Sensibilidade a maiúsculas e minúsculas

Python distingue entre maiúsculas e minúsculas. Portanto, variável, Variável e VARIÁVEL são consideradas variáveis diferentes.

Ponto e vírgula

Diferente de outras linguagens, o Python não requer o uso de ponto e vírgula (;) ao final de cada instrução. No entanto, se você desejar escrever várias instruções em uma única linha, pode separá-las com um ponto e vírgula. Por exemplo:

```
instrucao1; instrucao2; instrucao3
```

Operadores Matemáticos em Python

Operador	Descrição	Exemplo	Resultado
`+`	Adição	`5 + 3`	`8`
`-`	Subtração	`10 - 4`	`6`
`*`	Multiplicação	`7 * 2`	`14`
`/`	Divisão (retorna float)	`10 / 2`	`5.0`
`//`	Divisão inteira (retorna int)	`10 // 3`	`3`
`%`	Módulo (resto da divisão)	`10 % 3`	`1`
`**`	Exponenciação	`2 ** 3`	`8`
`()`	Parênteses (precedência)	`(2 + 3) * 4`	`20`

SENAI

Aritmética - Operadores aritméticos (matemáticos).

Expressão Algébrica	Expressão em Python	Exemplo de Código	Resultado
$a + b$	`a + b`	`resultado = 5 + 3`	`8`
$a - b$	`a - b`	`resultado = 10 - 4`	`6`
$a \times b$	`a * b`	`resultado = 7 * 2`	`14`
$\frac{a}{b}$	`a / b`	`resultado = 10 / 2`	`5.0`
a^b	`a ** b`	`resultado = 2 ** 3`	`8`
$\frac{a}{b}$ (divisão inteira)	`a // b`	`resultado = 10 // 3`	`3`
$a \mod b$	`a % b`	`resultado = 10 % 3`	`1`
$\frac{a+b}{c}$	`(a + b) / c`	`resultado = (5 + 3) / 2`	`4.0`
$a \times (b + c)$	`a * (b + c)`	`resultado = 2 * (3 + 4)`	`14`

Álgebra: $$m = \frac{a + b + c + d + e}{5}$$

```
m = (a + b + c + d + e) / 5
```

```
a = 10
b = 20
c = 30
d = 40
e = 50
m = (a + b + c + d + e) / 5
print("A média é:", m)
```

Regras de precedência de operador

1. As operações de **multiplicação**, **divisão** e de **resto** são aplicadas primeiro. Se uma expressão contiver várias dessas operações, elas serão aplicadas **da esquerda para a direita**. Os operadores de multiplicação, divisão e resto têm o mesmo nível de precedência.
2. As operações de **adição** e **subtração** são aplicadas em seguida. Se uma expressão contiver várias dessas operações, os operadores serão aplicados da **esquerda** para a **direita**. Os operadores de adição e subtração têm o mesmo nível de precedência.

Precedência	Operador	Descrição
1	`()` (parênteses)	Agrupamento de expressões
2	`**`	Exponenciação
3	`*`, `/`, `//`, `%`	Multiplicação, divisão, divisão inteira, módulo
4	`+`, `-`	Adição e subtração

Regras de precedência de operador

Operador(es)	Operação(ões)	Ordem de avaliação (precedência)
*	Multiplicação	Avaliado primeiro. Se houver vários operadores desse tipo, eles são avaliados da *esquerda para a direita*.
/	Divisão	
%	Resto	
+	Adição	Avaliado em seguida. Se houver vários operadores desse tipo, eles são avaliados da *esquerda para a direita*.
-	Subtração	
=	Atribuição	Avaliado por último.

```
resultado1 = 10 + 2 * 3 - 8 / 4 % 3
# Passo a passo:
# 1. Multiplicação: 2 * 3 = 6
# 2. Divisão: 8 / 4 = 2.0
# 3. Resto: 2.0 % 3 = 2.0
# 4. Adição e subtração: 10 + 6 - 2.0 = 14.0
print("Resultado 1:", resultado1)  # Saída: 14.0
```

Álgebra: $z = pr\%q + w/x - y$

```
z = p * r % q + w / x - y
    6   1   2   4   3   5
```

Regras de precedência de operador

Passo 1. `y = 2 * 5 * 5 + 3 * 5 + 7`

`2 * 5 = 10` *(Em primeiro lugar a multiplicação da extremidade esquerda)*

Passo 2. `y = 10 * 5 + 3 * 5 + 7`

`10 * 5 = 50` *(Multiplicação da esquerda)*

Passo 3. `y = 50 + 3 * 5 + 7`

`3 * 5 = 15` *(Multiplicação antes da adição)*

Passo 4. `y = 50 + 15 + 7`

`50 + 15 = 65` *(Adição da esquerda)*

Passo 5. `y = 65 + 7`

`65 + 7 = 72` *(última operação)*

Fig. 2.11 Cálculo de um polinômio do segundo grau.

Variáveis em Python

Todas as variáveis ou constantes são estruturadas por um **tipo de dado**!

- **Idade**: Representada por um **número inteiro**
  - idade = 25 *# tipo int*
- **Peso**: Representado por um **número fracionário** (ponto flutuante).
  - peso = 68.5 *# tipo float*
- **Letra**: Representada por um **caractere** (string de comprimento 1).
  - letra = 'A' *# tipo str*
- **Texto/String:** Representado por uma **sequência de caracteres**.
  - texto = "Bem-vindo ao Python!" *# tipo str*
- **Boolean (Boleano):** Representado por um **valor lógico**: **True** ou **False/**Utilizado para expressar condições ou estados lógicos.
  - ativo = True *# tipo bool*

SENAI

Leitura e Escrita de Dados em Python

Em Python, a entrada e saída de dados nos permite **interagir** com o usuário e manipular arquivos.

Podemos **solicitar** informações ao usuário, **mostrar** resultados na tela e ler ou **escrever** dados em arquivos externos.

SENAI

Leitura de Dados com input()

A função input() lê dados do usuário como uma string.

Exemplo:

```
nome = input("Digite seu nome: ")
print("Olá,", nome)
```

Problema: Como ler números (int, float) e booleanos (bool)?

Leitura de Dados com input()

Importante

A função input() sempre retorna uma cadeia de texto. Se você deseja trabalhar com outros tipos de dados, como números inteiros ou flutuantes, deve realizar uma conversão explícita utilizando funções como int() ou float().

python™

SENAI

Leitura de Números Inteiros (int)

Convertendo para int

Use int() para converter a entrada em um número inteiro.

Exemplo:

```
idade = int(input("Digite sua idade: "))
print("Daqui a 5 anos, você terá", idade + 5, "anos.")
```

Cuidado: Se o usuário digitar algo não numérico, ocorrerá um erro (ValueError).

Leitura de Números Decimais (float)

Convertendo para float

Use float() para converter a entrada em um número decimal.

Exemplo:

```
altura = float(input("Digite sua altura (em metros): "))
print("Sua altura é", altura, "metros.")
```

Aplicação: Cálculos matemáticos com precisão decimal.

Leitura de Texto (str)

Trabalhando com Strings

A função input() já retorna uma string por padrão.

Exemplo:

```
cidade = input("Digite sua cidade: ")
print("Você mora em", cidade + "!")
```

Dica: Use métodos de string (strip(), upper(), etc.) para manipular o texto.

Leitura de Booleanos (bool)

Convertendo para bool

Use bool() para converter a entrada em um valor booleano (True ou False).

Exemplo:

```
resposta = input("Você gosta de Python? (sim/não): ").lower() == "sim"
print("Resposta:", resposta)
```

Observação:

Strings não vazias são convertidas para True.

Strings vazias são convertidas para False.

Exibindo Dados com print()

A função print() exibe dados na tela.

Exemplo:

```
nome = input("Insira seu nome: ")
idade = input("Insira sua idade: ")
print("Olá, " + nome + "!")
print("Você tem " + idade + " anos.")
```

Recurso: Use f-strings para formatação avançada.

```
print("Olá, mundo!")  # Texto simples
print("Idade:", 25)   # Texto e variável
altura = 1.75
print(f"Sua altura é {altura} m")  # f-string (formatação moderna)
```

Podemos utilizar a f-string (formatação de cadeias) para inserir variáveis diretamente dentro de uma cadeia de texto.

Exemplo Integrado

Programa que lê nome, idade, altura e uma resposta booleana, e exibe uma mensagem personalizada:

```
1    nome = input("Digite seu nome: ")
2    idade = int(input("Digite sua idade: "))
3    altura = float(input("Digite sua altura (em metros): "))
4    gosta_python = input("Você gosta de Python? (sim/não): ").lower() == "sim"
5
6    print(f"Olá, {nome}! Você tem {idade} anos, {altura}m de altura.")
7    print("Gosta de Python?", gosta_python)
```

Exercícios

1. Faça um programa que imprima a mensagem: “É preciso fazer todos os programas para aprender”.

2. Crie um programa que leia um número com casa decimal e imprima a terça parte deste número.

3. Faça um programa que receba quatro números inteiros, calcule e mostre a soma desses números

4. Faça um programa que receba duas notas, calcule e mostre a média ponderada dessas notas, considerando peso 7 para a primeira e peso 3 para a segunda

5. Crie um programa que leia dois números inteiros e imprima o dividendo, divisor, quociente e resto.

Conclusões

- Algoritmos nos mostram os passos;
- Lógica se preocupa com a coerência;
- Com Python podemos colocar tudo em prática.

Veremos cada um destes itens mais a fundo.