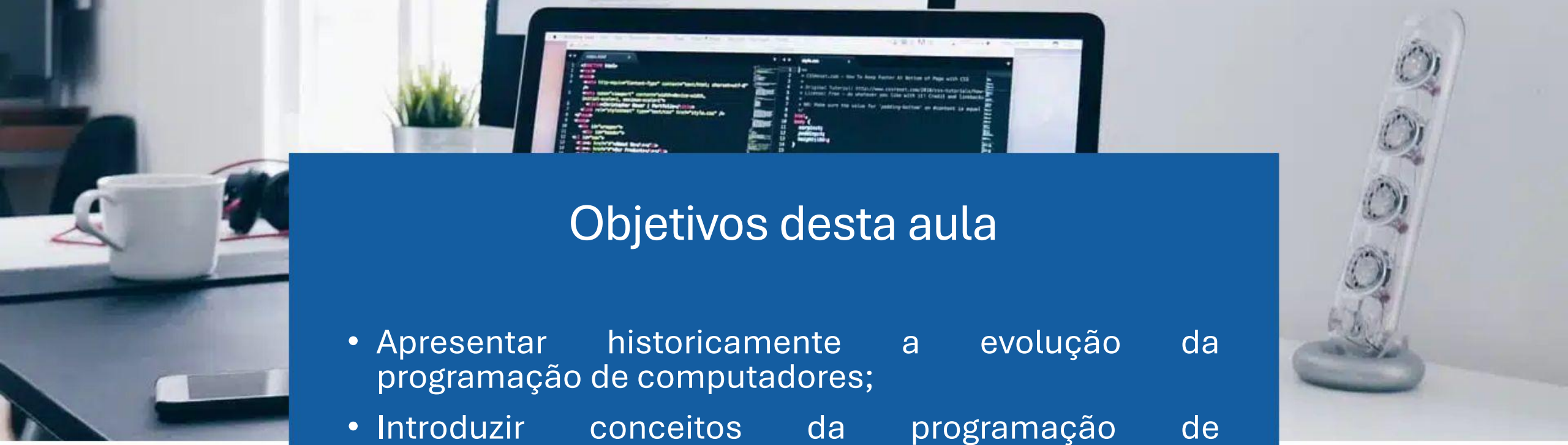
Aula #1:

Introdução à Programação com Python

Prof. Dr. Luiz Álvaro de Oliveira Júnior



Objetivos desta aula

- Apresentar historicamente a evolução da programação de computadores;
- Introduzir conceitos da programação de computadores e da linguagem Python;
- Exemplificar linguagens compiladas e linguagens interpretadas;
- Conhecer algumas ferramentas;
- Escrever algoritmos para resolver problemas simples.

Sumário

Alguns fatos históricos

01

Ada Lovelace escreve o 1º algoritmo para a Máquina Analítica de Babbage **(1842 – 1843)**

02

Código de Máquina (binário) foi usado nos primeiros computadores (ENIAC). A programação era feita com fios e chaves **(década de 1940)**.

03

Surge a linguagem **Fortran**, a primeira linguagem de alto nível **(década de 1950)**.

04

A linguagem **C** é criada, influenciando Python, Java e outras **(década de 1970)**.

Alguns fatos históricos

05

Guido von Rassum lança a linguagem Python **(década de 1990)**.

06

Linguagens modernas (JavaScript e Swift) dominam a programação web e mobile **(década de 2000)**.

07

Google lança o Tensorflow, um dos principais catalisadores da popularização e democratização do **machine learning** **(década de 2010)**.

08

O **Python** se consolida como a linguagem universal da análise de dados, automação e ciência computacional **(década de 2020)**.

Introdução à Programação

Programação é o processo de escrever instruções (códigos) que um computador pode entender e executar para realizar tarefas específicas.

Essas instruções são escritas em linguagens de programação (como Python, Java ou C++) e transformadas em comandos que o hardware do computador executar.

Os componentes básicos da programação são o **algoritmo**, as **linguagens** de programação e o **compilador/interpretador**.

Introdução à Programação

Algoritmos são sequências lógicas de passos para resolver algum problema.

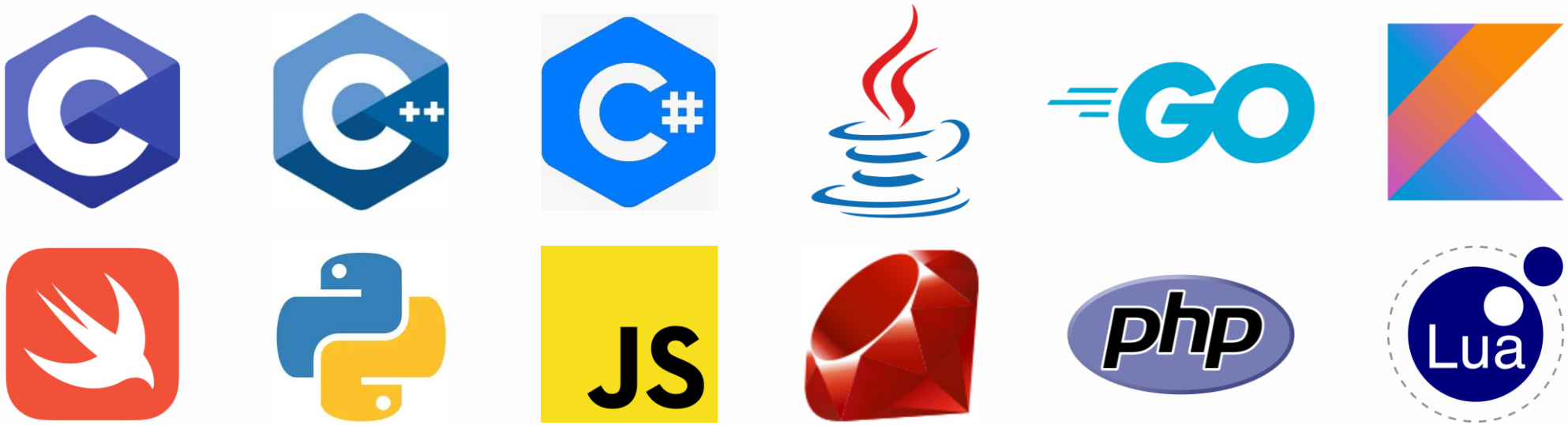
Exemplo: Somar dois números e mostrar o resultado.

Passo 1: Pegue o primeiro número;
Passo 2: Pegue o segundo número;
Passo 3: Some-os.
Passo 4: Mostre o resultado.

Introdução à Programação

Linguagens de programação são responsáveis por traduzir os algoritmos para instruções que o computador entenda.

Exemplos: C, C++, C#, Java, Go, Kotlin, Swift, Python, Javascript, Ruby, PHP, Lua, etc...



Introdução à Programação

Compiladores/interpretadores são programas de computador que convertem o código escrito em alguma linguagem de programação para algo que o computador consiga entender e executar.

Os **compiladores** analisam o código completo e o converte para linguagem de máquina (binário), gerando um executável. Por sua vez, os interpretadores **executam o código linha por linha**, traduzindo e executando as instruções em tempo real sem criar um arquivo executável separado.

Introdução à Programação

Compiladas ou interpretadas, as diferentes linguagens de programação seguem estratégias diferentes no processamento do código.

Na linguagem Python, o código é analisado e convertido em bytecode (um formato intermediário que a máquina virtual Python interpreta), sendo esse bytecode executado de forma sequencial (linha por linha) durante a execução do programa.

Algoritmos

Algoritmos são sequências lógicas e ordenadas de passos para resolver um problema ou realizar uma tarefa. Eles existem tanto na programação quanto no cotidiano (como receitas de bolo ou instruções de montagem de móveis).

Características de um Bom Algoritmo:

- **Precisão**: Passos claros e sem ambiguidade.
- **Finitude**: Deve ter um fim (não pode ser infinito).
- **Eficiência**: Resolver o problema com o mínimo de recursos.

Algoritmos

Exercício #1: Escrever o algoritmo em pseudocódigo que represente a tarefa de **trocar uma lâmpada**.

1. Desligar o interruptor de energia.
2. Pegar uma escada estável.
3. Subir na escada e girar a lâmpada queimada no sentido anti-horário.
4. Pegar uma lâmpada nova (mesmo modelo e potência).
5. Encaixar a lâmpada nova e girar no sentido horário.
6. Ligar o interruptor para testar.

Algoritmos

Exercício #2: Escrever o algoritmo em pseudocódigo que represente a tarefa de **preparar uma torrada** usando uma torradeira.

1. Pegar 1 fatia de pão de forma.
2. Passar manteiga dos dois lados do pão.
3. Inserir o pão na torradeira.
4. Ajustar o timer para o tempo desejado.
5. Pressionar o botão de ligar.
6. Esperar até ejetar.

Por que aprender a programar?

A programação é uma ferramenta poderosa para diversas áreas da engenharia, otimizando processos, análises e automações.

Ela permite, por exemplo, testar e avaliar modelos, estruturas, cenários, estratégias entre outras possibilidades a um custo bem menor que fazer a mesma coisa experimentalmente.

A linguagem Python

Nesta disciplina, usaremos a linguagem Python. A seguir algumas características que justificam a utilização dessa linguagem.

- O código fica mais claro e objetivo que em outras linguagens;
- Suporta tanto programação procedural (funções e sequências), quanto orientação a objetos (classes e heranças) e programação funcional;
- Tipagem dinâmica;
- Bibliotecas padrão poderosa;

A linguagem Python

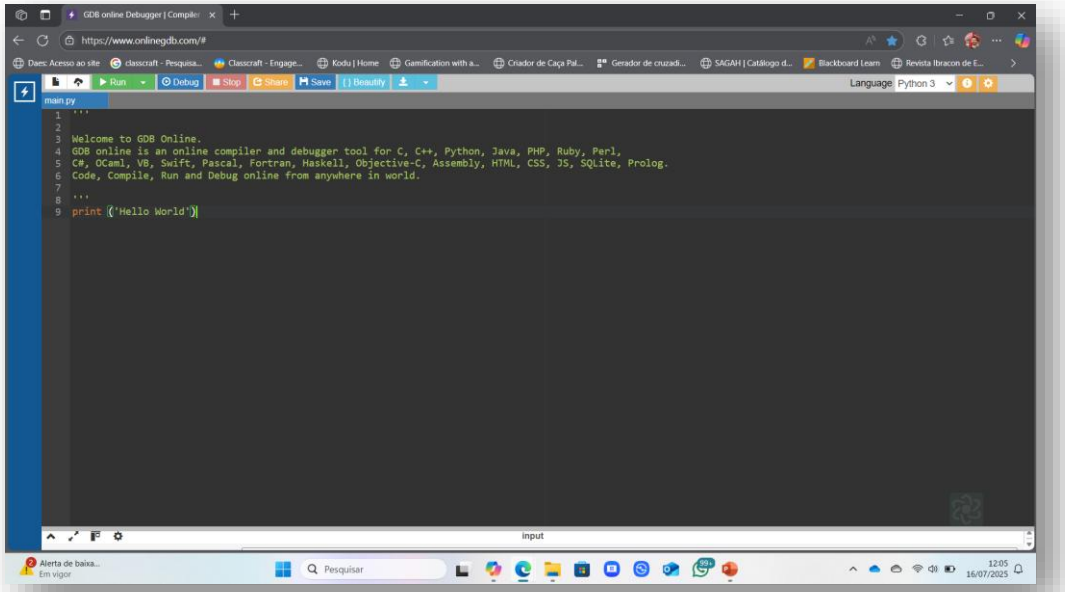
- Variedade de bibliotecas disponíveis (cerca de 125 mil);
- Multiplataforma, operando em Windows, macOS, Linux, dispositivos embarcados (Raspeberry Pi Pico, por exemplo);
- Grande disponibilidade de material de apoio desenvolvido pela própria comunidade de programadores;
- Open Source;
- Remove o lixo de programação automaticamente;
- Chama código ou interage com outras linguagens;

A linguagem Python

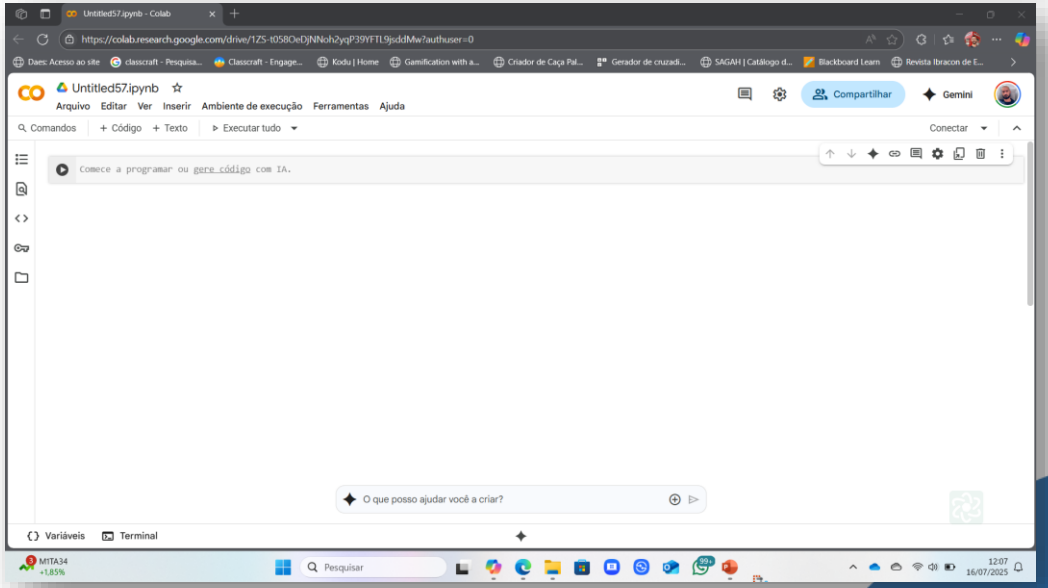
- Há diversas ferramentas computacionais que possuem o interpretador Python, em versões desktop ou online.
- Exemplos:
  - **Desktop (requer instalação)**: Visual Studio Code (VSCode) e Anaconda
  - **Online (não requer instalação)**: onlineGDB, Google Colab

A linguagem Python

Online GDB:
https://www.onlinegdb.com/#



Google colab:
https://colab.google



Conceitos importantes

Em Python, **estruturas de dados** são formas organizadas de armazenar e manipular valores de maneira eficiente. Elas são essenciais para representar coleções de informações, desde listas simples até conjuntos mais complexos e tabelas de dados. Aqui vai um panorama das principais

Entre as estruturas de dados do Python, destacam-se:

1. **String**: Imutável, estrutura para trabalhar com textos.
2. **Lista**: Mutável, ordenada, aceita elementos repetidos.

Conceitos importantes

3. **Tupla**: Imutável, ordenada, rápida.
4. **Dicionário**: Mutável, associa chaves a valores (mapeamento).
5. **Conjunto**: Mutável, não ordenado, sem elementos duplicados.
6. **Array**: Mutáveis, ordenados, processamento eficiente.

As cinco primeiras são **estruturas de dados nativas** (originadas da própria linguagem), enquanto a última é uma **estrutura de dados especializada ou avançada**, trazida por uma biblioteca externa e que fornece recursos especializados.

Conceitos importantes

```
string = “palavra”
lista = [1, 2, 3]
tupla = (1, 2, 3)
conjunto = {1, 2, 3}
dicionario = {"a": 1, "b": 2}
```

Conceitos importantes

Em Python, **palavras reservadas** são termos que têm um **significado especial para a linguagem**. Elas são usadas para definir a estrutura e o comportamento do código. Assim, elas **não podem ser usadas como nomes de variáveis, funções ou identificadores**.

True : valor booleano VERDADEIRO;

False: valor booleano FALSO;

and: operador lógico E;

or : operador lógico OU;

not: operador lógico NÃO;

Palavras reservadas

if: início de uma condição (SE);

elif: condição alternativa (SENÃO SE);

else: bloco executado se condições anteriores forem falsas (SENÃO);

for: loop para iterar em iterável;

while: loop condicional

break: interrompe loop;

continue: continua loop pulando para a próxima iteração;

pass: passa sem fazer nada;

Palavras reservadas

in: verifica se um elemento está contido em outro;

def: define uma função;

return: retorna um ou mais valores de dentro de uma função;

from: importa partes específicas de um módulo especificado;

import: importa um módulo;

as: atribui um nome alternativo a um módulo;

try: inicia bloco de tratamento de exceção;

except: captura erro durante a execução;

Palavras reservadas

finally: executa sempre, independente de erro;

global: declara que uma variável é global;

Há outras palavras reservadas, mas elas não são usadas no contexto da nossa disciplina.