

INTRODUÇÃO AO PYTHON



OBJETIVO GERAL

- Apresentar os fundamentos do Python (história, contexto, versões e ambiente) e sua relevância na era da IA,
- Capacitar o aluno a reconhecer quando e como usar Python com eficiência.



OBJETIVOS ESPECÍFICOS

- Diferenciar linguagens interpretadas e compiladas
- Entender a linha do tempo das linguagens de programação
- Introduzir Python: filosofia, usos e ecossistema
- Revisar versões e compatibilidade
- Demonstrar um ambiente de programação Python



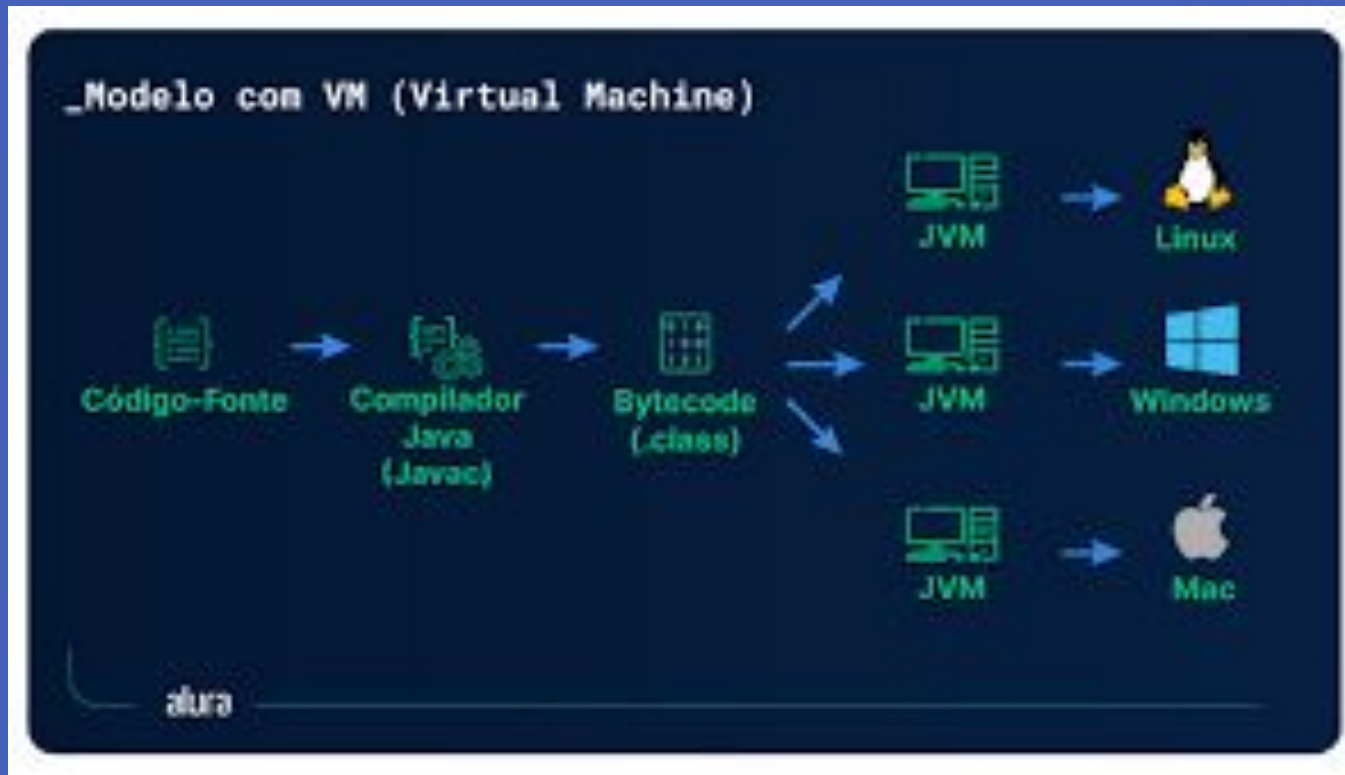
ROTEIRO

- Introdução (interpreted vs compiled)
- Desenvolvimento
 - Introdução ao Python
 - Histórico
 - Contexto
 - Versões
 - Ambiente de Programação
- Síntese e encerramento



INTRODUÇÃO – INTERPRETADA VS COMPILAGADA (CONCEITOS)

- Compilada: transforma o código em binário antes de executar (ex.: C/C++/Go/Rust).



INTRODUÇÃO – INTERPRETADA VS COMPILADA (CONCEITOS)

- Interpretada: executa linha a linha por um interpretador (ex.: Python, JS).



EXEMPLOS

- Dia a dia: scripts de automação (interpretada) vs sistemas de alto desempenho (compilada).

- Python (interpretada):

```
print('Olá, mundo!')
```

- C (compilada):

```
#include <stdio.h>
```

```
int main(){ printf("Olá, mundo!\n"); return 0; }
```

VANTAGENS E DESVANTAGENS

- Interpretada

- Vantagens: portabilidade, ciclo rápido, ótima para protótipos e automação.
- Desvantagens: menor desempenho bruto, dependência do interpretador.

- Compilada

- Vantagens: alto desempenho, binário otimizado e distribuível.
- Desvantagens: ciclo de build/teste mais lento, menor flexibilidade.

LINHA DO TEMPO: SOFTWARE 1.0 → 3.0 (IA)

- Software 1.0 – Baixo nível (Assembly, C): foco em hardware e eficiência.
- Software 2.0 – Alto nível (Python, Java, JS): produtividade, bibliotecas e ecossistemas.
- Software 3.0 – IA/LLMs (ChatGPT, Copilot, APIs): geração/assistência de código e automação inteligente.



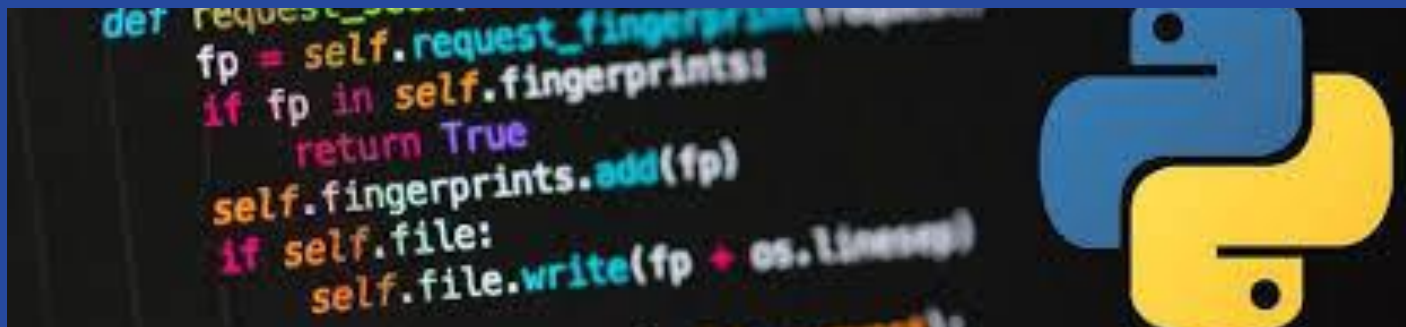
DESENVOLVIMENTO – PERGUNTA PROVOCADORA

- Ainda é relevante aprender codificação na era da IA? Qual a importância do Python nesse contexto?



DESENVOLVIMENTO – INTRODUÇÃO AO PYTHON

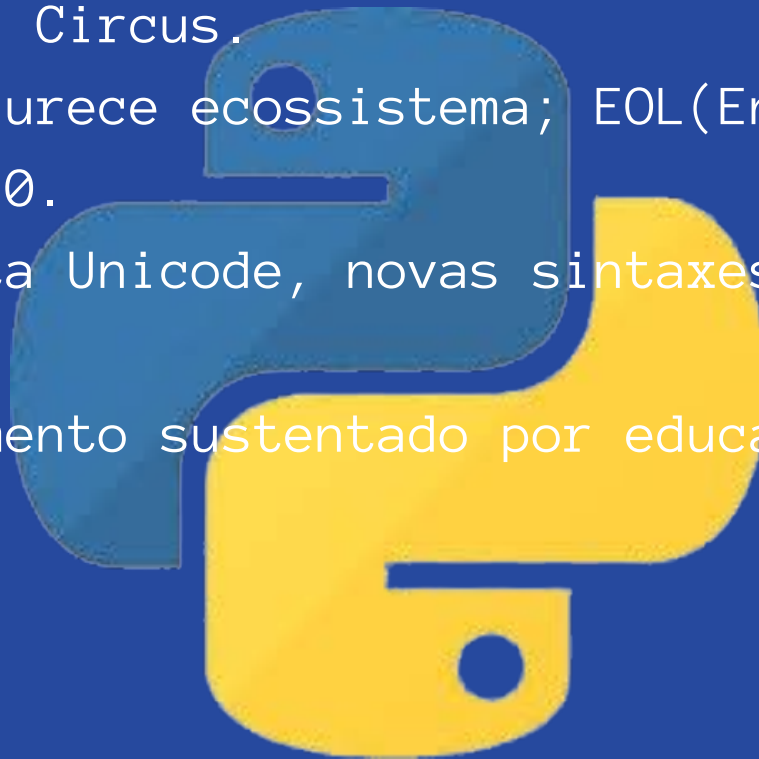
- Linguagem de alto nível, interpretada, multi-paradigma. Python é uma linguagem de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte.
- Filosofia: Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros.



DESENVOLVIMENTO – HISTÓRICO

(RESUMO)

- Primeira versão: Matemático Guido van Rossum (Centrum Wiskunde & Informatica – Amsterdã, 1991). O nome Python foi inspirado na série de comédia britânica Monty Python's Flying Circus.
- Python 2.x amadurece ecossistema; EOL(End of Life version) em 2020.
- Python 3.x adota Unicode, novas sintaxes e melhorias de performance.
- Futuro: Crescimento sustentado por educação, dados e IA.



DESENVOLVIMENTO – CONTEXTO ATUAL

- Python é onipresente em fluxos de dados e IA (NumPy, pandas, scikit-learn, PyTorch, TensorFlow), assim como em Infra/DevOps.
- Mercado: forte demanda, curva de aprendizado amigável.
- Projetado para ser intuitivo e fácil de programar (sem sacrificar o poder)
- Código aberto, com uma grande comunidade de pacotes e recursos
- Uma das linguagens de programação mais usadas no mundo
- Linguagem “Testado e Verdadeiro” que está em desenvolvimento há décadas
- Visualizações de alta qualidade
- Funciona na maioria dos sistemas operacionais e plataformas

DESENVOLVIMENTO – AMBIENTE DE PROGRAMAÇÃO

- IDEs (ambiente de desenvolvimento integrado)/editores: VS Code, PyCharm, Jupyter, IDLE.
- Fluxo típico: criar venv → instalar dependências → editar → executar → testar.
- Exemplo prático: Hello World em diversas IDEs (passo a passo nas próximas telas).



DESENVOLVIMENTO – AMBIENTE DE PROGRAMAÇÃO

- IDEs para teste (no curso utilizaremos Notebooks – Jupyter ou Google Colab):

Pycharm:

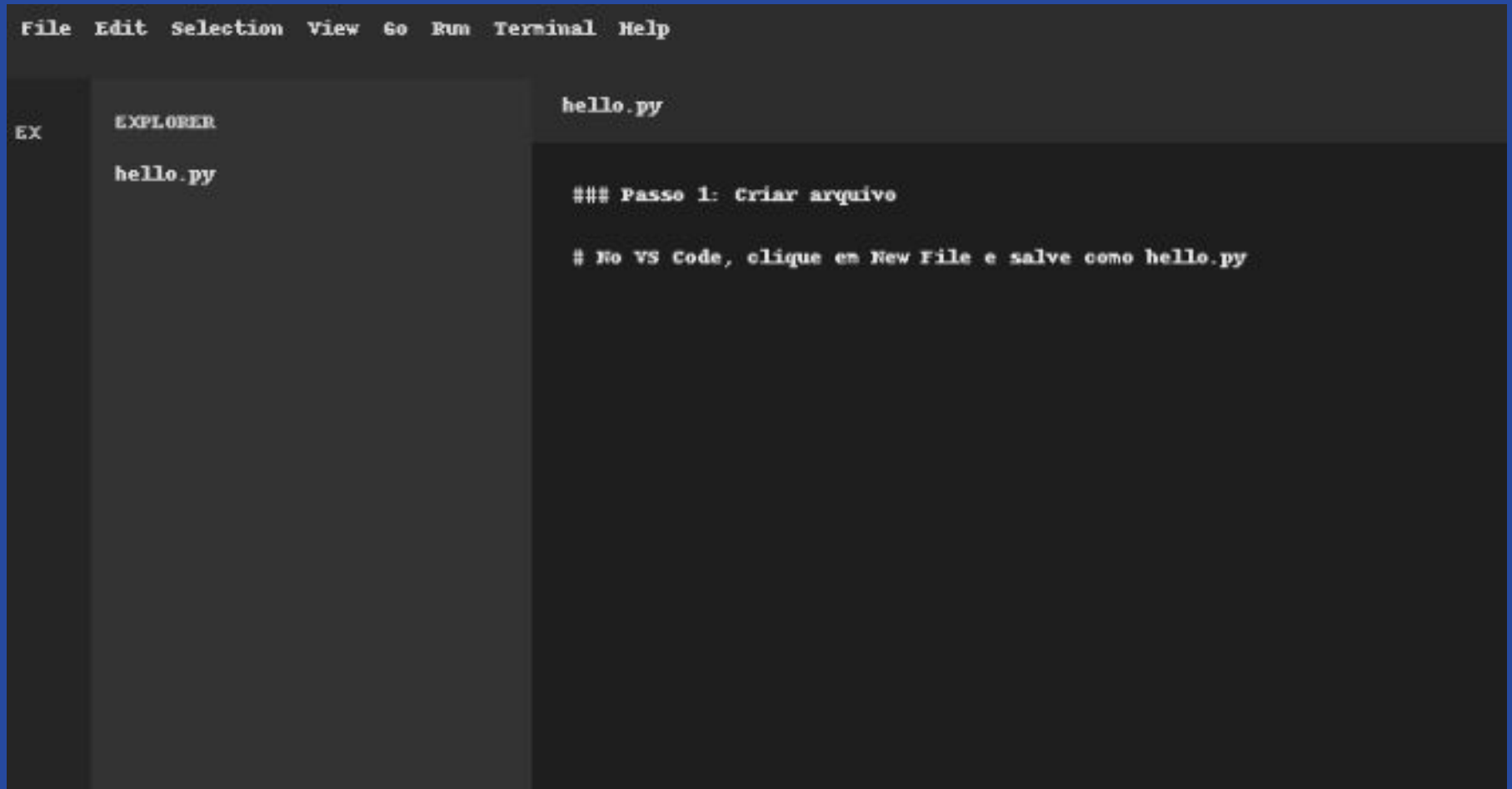
Jupyter:

<https://jupyter.org/try-jupyter/lab/>

Google Colab:

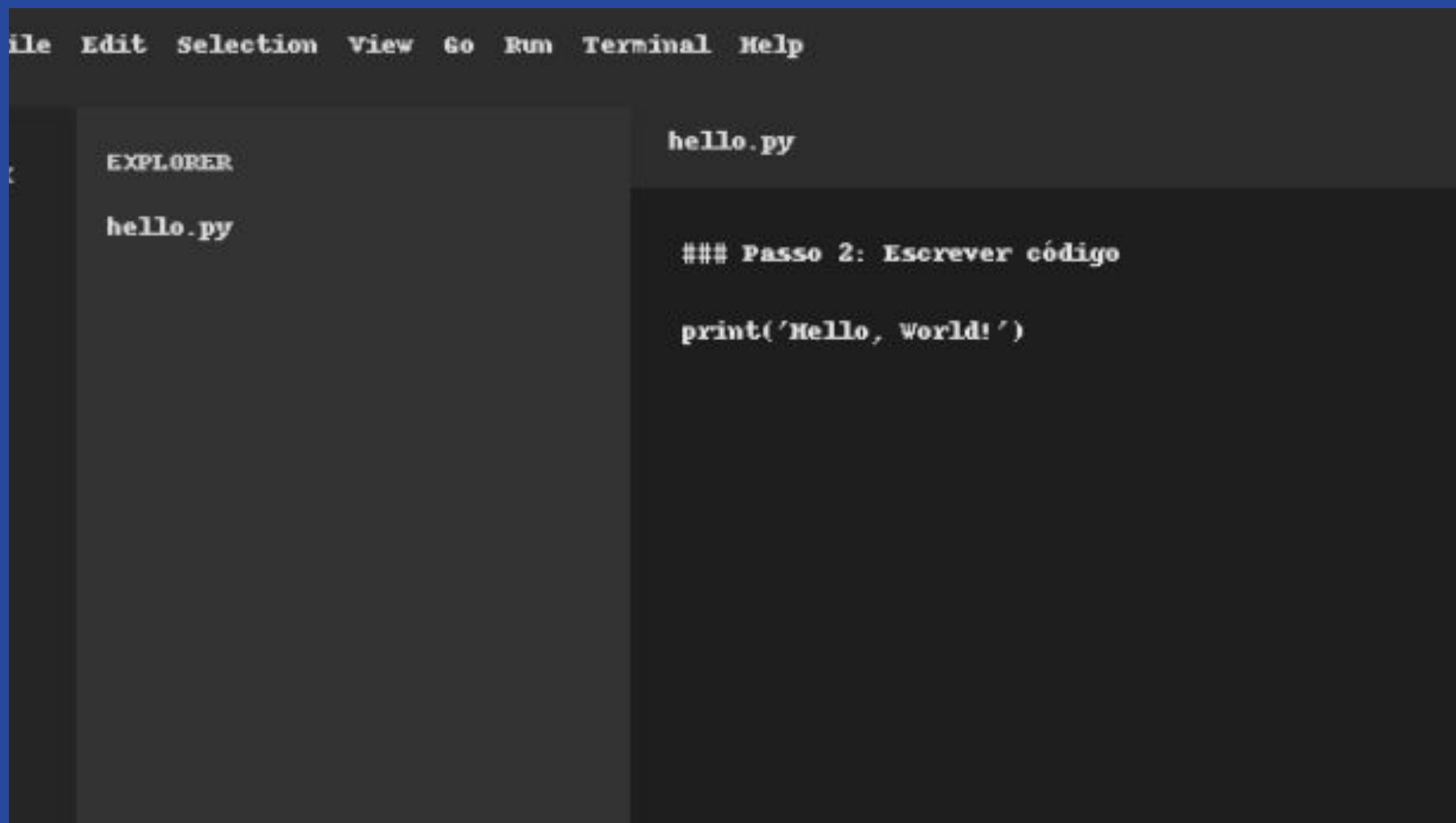
<https://colab.google/>

HELLO WORLD – PASSO 1



Criar um novo arquivo e salvar como hello.py.

HELLO WORLD – PASSO 2



```
File Edit Selection View Go Run Terminal Help

EXPLORER

hello.py

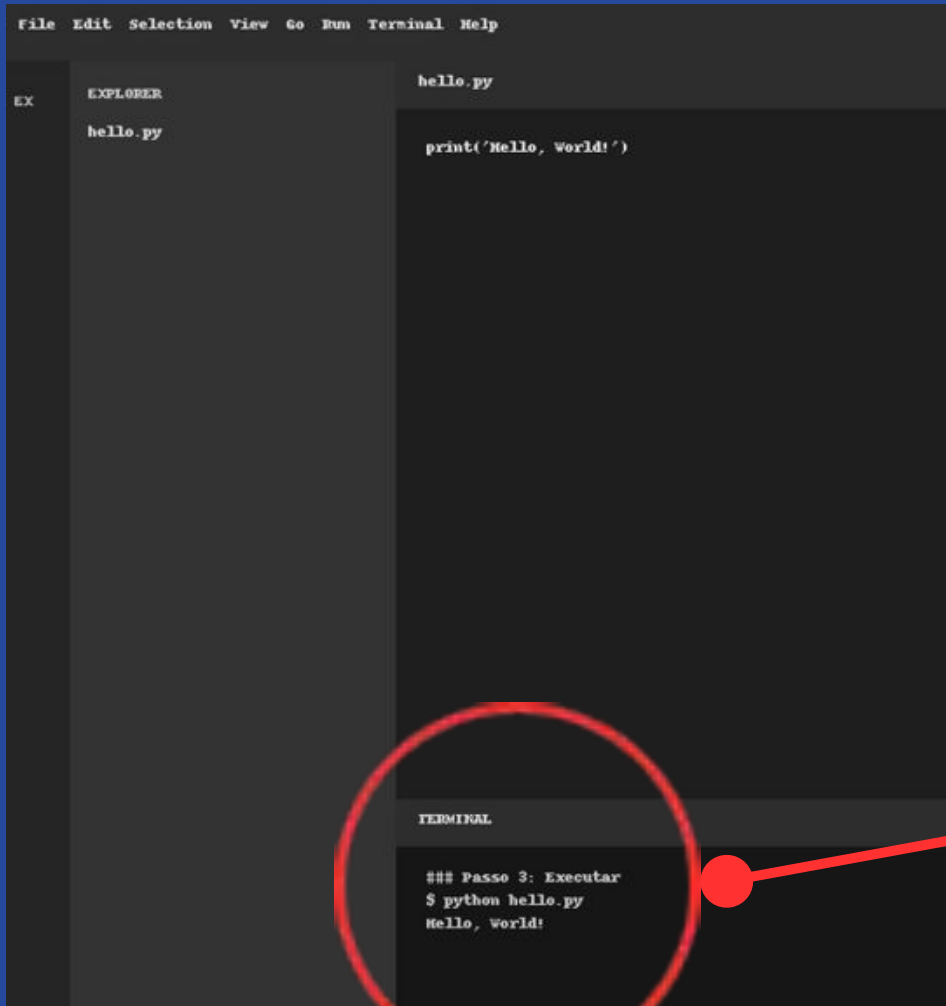
hello.py

### Passo 2: Escrever código

print('Hello, World!')
```

Digitar o código: `print('Hello, World!')` e salvar.

HELLO WORLD – PASSO 3



```
File Edit Selection View Go Run Terminal Help
```

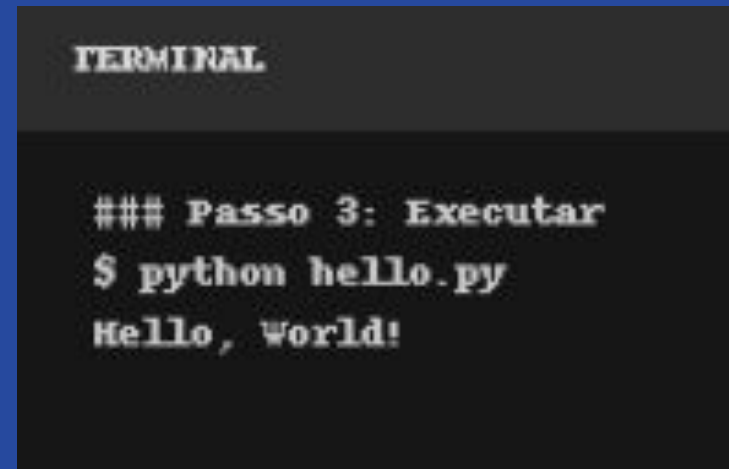
EXPLORER

hello.py

```
print('Hello, World!')
```

TERMINAL

```
### Passo 3: Executar  
$ python hello.py  
Hello, World!
```

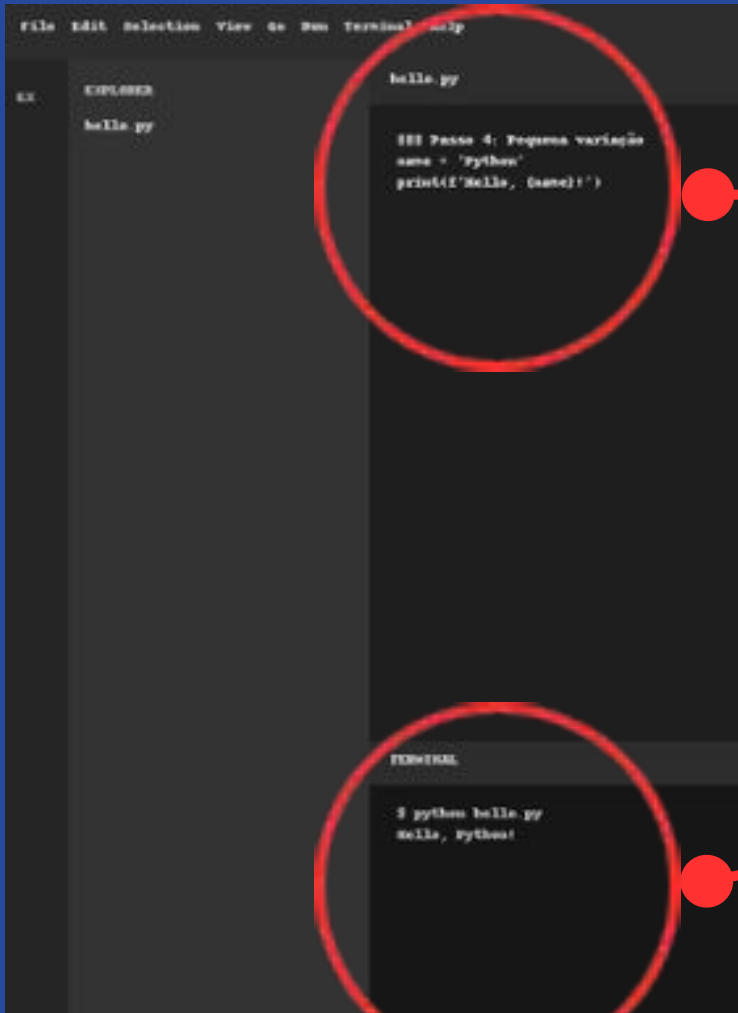


```
TERMINAL
```

```
### Passo 3: Executar  
$ python hello.py  
Hello, World!
```

Executar pelo terminal integrado: `python hello.py` → saída 'Hello, World!'.

HELLO WORLD – PASSO 4



hello.py

```
### Passo 4: Pequena variação  
name = 'Python'  
print(f'Hello, {name}!')
```

TERMINAL

```
$ python hello.py  
Hello, Python!
```

Variar com f-strings: `print(f'Hello, {name}!')`.

SÍNTESE – POR QUE APRENDER A CODAR (E PYTHON) AINDA IMPORTA?

- Entender > copiar: saber programar permite avaliar, adaptar e depurar respostas da IA.
- Python é a “cola” entre dados, APIs e modelos (pipelines de ML, automações, integrações).
- Prototipagem relâmpago: de um script a um MVP (Minimum Viable Product) em horas.
- Empregabilidade: Python é requisito transversal (dados, IA, web, automação, pesquisa).
- Conclusão: A IA amplia o alcance de quem programa – aprender Python torna você mais produtivo e criativo.

REFERÊNCIAS BIBLIOGRÁFICAS

- VAN ROSSUM, G.; DRAKE, F. L. Python 3 Reference Manual. CreateSpace, 2009.
- DOWNEY, A. B. Think Python (2e). Green Tea Press, 2015.
- MCKINNEY, W. Python for Data Analysis (3e). O'Reilly, 2022.
- PEP 8 – Style Guide for Python Code. Python Software Foundation.
- NUMPY, PANDAS, SCIPY, SCIKIT-LEARN, PYTORCH, TENSORFLOW – Documentações oficiais.

