

# PYTHON QUEST



**THIAGO GALVÃO**

# COMEÇANDO COM PYTHON

Se você é um jovem programador que já domina lógica de programação e algoritmos, então este e-book é para você. Vamos embarcar juntos em uma jornada épica para dominar os fundamentos do Python, a linguagem que tem conquistado o mundo da tecnologia e da cultura geek!

Se você já entende conceitos como loops, condicionais e estrutura de dados, está um passo à frente. Agora, seu desafio é aprender Python da melhor forma possível. Imagine que sua mente é como um mago em um RPG: você já conhece os feitiços básicos (lógica de programação), mas agora precisa dominar um novo grimório de encantamentos (Python). Então, afie sua varinha e prepare-se para a aventura!







# 01

## Sintaxe e Estruturas Básicas



# VARIÁVEIS E TIPOS DE DADOS

## Variáveis e Tipos de Dados: Seu Inventário Inicial

Em Python, criar uma variável é como guardar um item no seu inventário. Você simplesmente atribui um valor a um nome:



```
# Criando variáveis  
nome = "Alice"  
idade = 25  
saldo = 100.50
```

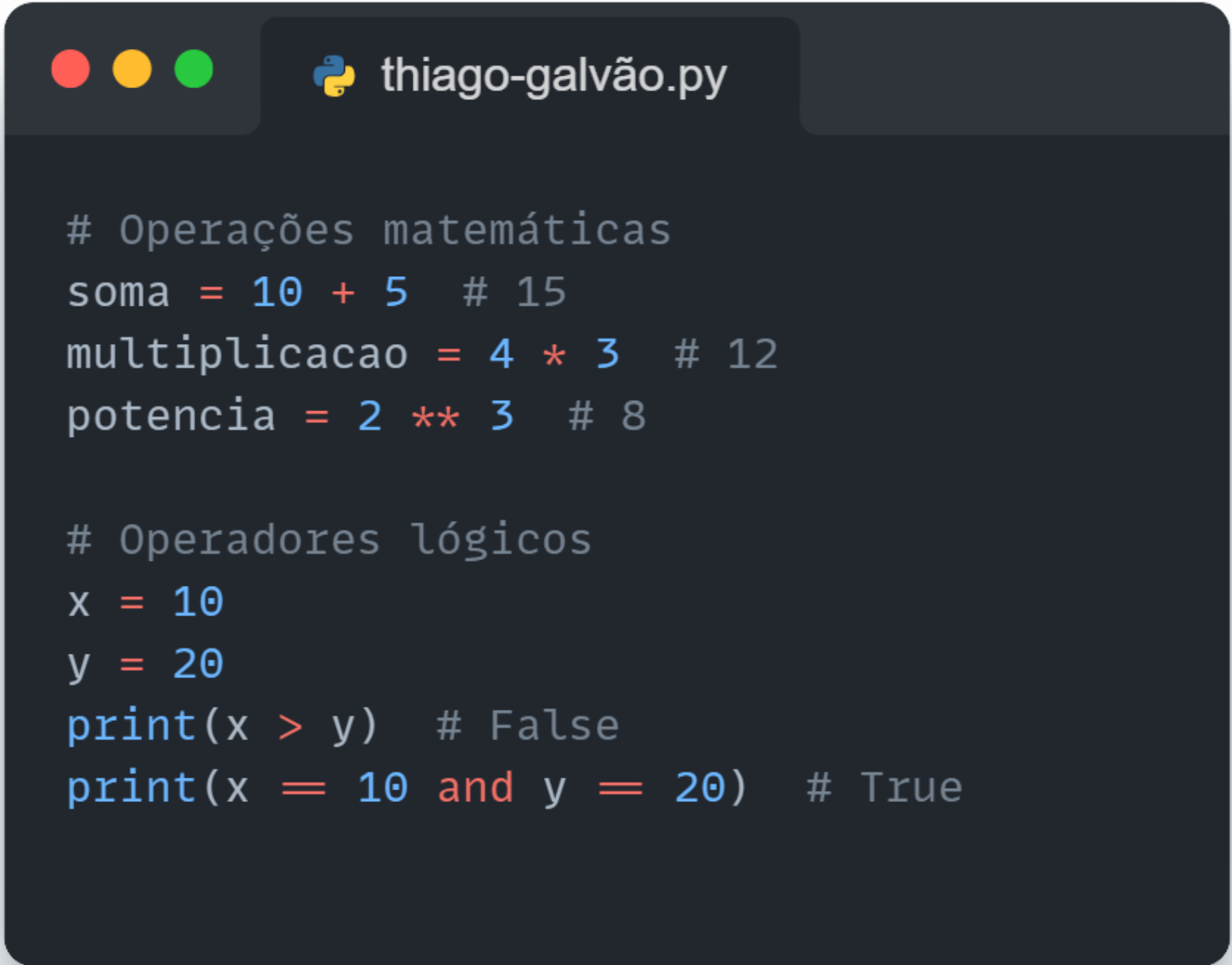
Python reconhece automaticamente o tipo de cada variável:

- *str* (string) para textos.
- *int* (inteiro) para números inteiros.
- *float* para números decimais.

# OPERADORES MATEMÁTICOS E LÓGICOS

## Operadores Matemáticos e Lógicos: O Combate do Código

Python oferece operadores básicos para você manipular valores e tomar decisões:

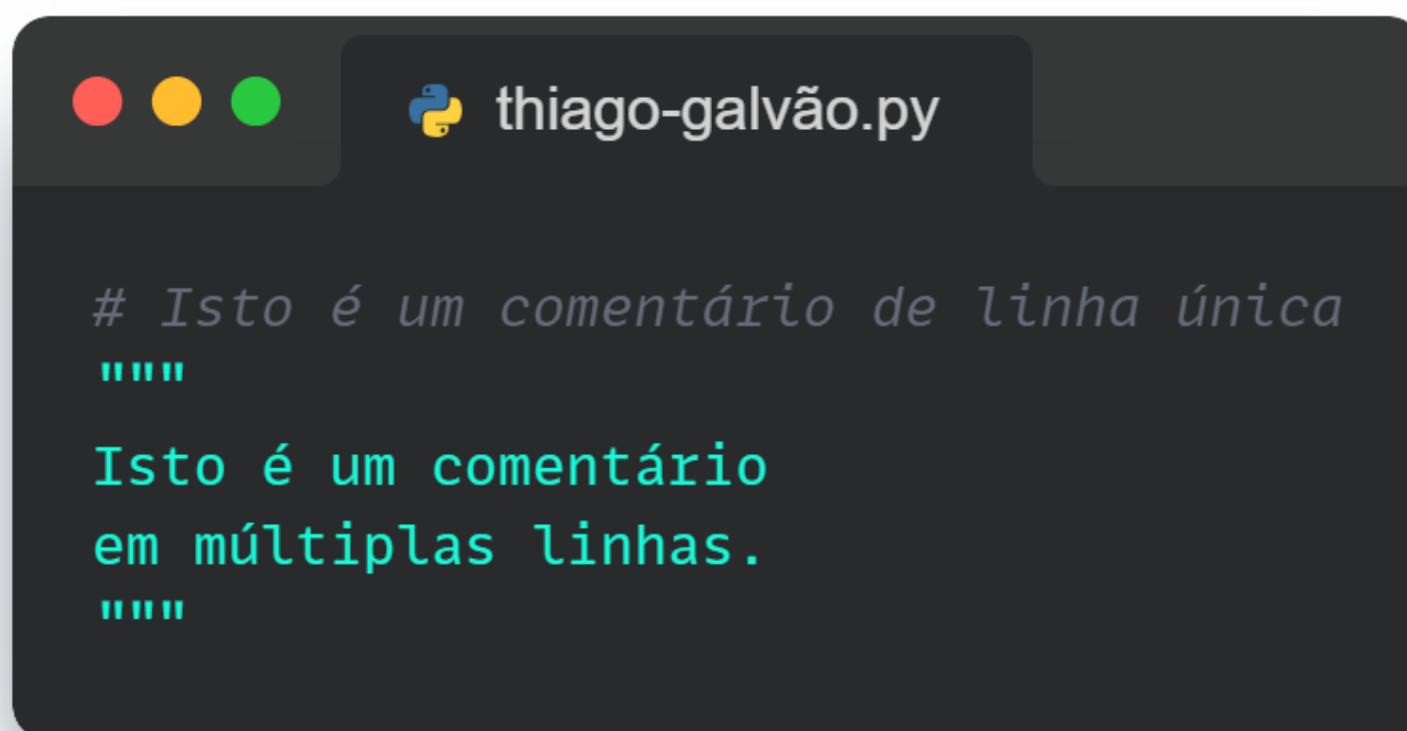


```
# Operações matemáticas
soma = 10 + 5 # 15
multiplicacao = 4 * 3 # 12
potencia = 2 ** 3 # 8

# Operadores lógicos
x = 10
y = 20
print(x > y) # False
print(x == 10 and y == 20) # True
```

## COMENTÁRIOS: NOTAS DO DIÁRIO DE AVENTURAS

Comentários ajudam a documentar o código, como anotações em um diário de viagem:



```
# Isto é um comentário de linha única
"""
Isto é um comentário
em múltiplas linhas.
"""
```



# 02

## Controle de Fluxo





# Controle de Fluxo

## Controle de Fluxo: A Arte das Decisões e Repetições

Python permite que seu código tome decisões e repita ações quando necessário.

### Condicionais (*if*, *elif*, *else*)

```
idade = 18
if idade >= 18:
    print("Você pode entrar!")
elif idade == 17:
    print("Quase lá!")
else:
    print("Volte mais tarde!")
```

### Loops (*for* e *while*)

```
# Loop for para percorrer uma lista
itens = ["Espada", "Poção", "Escudo"]
for item in itens:
    print(f"Peguei {item}!")

# Loop while para contar até 5
contador = 1
while contador <= 5:
    print(contador)
    contador += 1
```





# 03

## Funções e Modularização



# Funções e Modularização

Funções e Modularização: Reutilizando Código.

Funções ajudam a organizar o código e evitam repetições desnecessárias.

A code editor window with a dark background and light-colored text. The window has a title bar with three colored circles (red, yellow, green) and a Python logo followed by the filename 'thiago-galvão.py'. The code inside the editor is as follows:

```
def saudar(nome):  
    return f"Olá, {nome}!"  
  
print(saudar("Thiago"))
```



# 04

## Estruturas de Dados Avançadas



# Estruturas de Dados Avançadas

## Estruturas de Dados Avançadas: Organizadores de Informações

### Listas

```
thiago-galvão.py

personagens = ["Gandalf", "Aragorn", "Legolas"]
print(personagens[1]) # Aragorn
```

### Tuplas

```
thiago-galvão.py

tipo_pokemon = ("Fogo", "Água", "Terra")
print(tipo_pokemon[0]) # Fogo
```

### Dicionários

```
thiago-galvão.py

pokedex = {"Pikachu": "Elétrico", "Charmander": "Fogo"}
print(pokedex["Pikachu"]) # Elétrico
```





# 05

## Manipulação de Arquivos



# Manipulação de Arquivos

## Manipulação de Arquivos: Guardando Progresso

```
thiago-galvão.py

# Escrevendo em um arquivo
with open("save_game.txt", "w") as arquivo:
    arquivo.write("Level 10 - XP: 5000")

# Lendo um arquivo
with open("save_game.txt", "r") as arquivo:
    print(arquivo.read())
```



# 06

## Programação Orientada a Objetos (POO)



# Programação Orientada a Objetos (POO)

## Programação Orientada a Objetos (POO): Criando Personagens no Código

```
class Personagem:
    def __init__(self, nome, classe):
        self.nome = nome
        self.classe = classe

    def apresentar(self):
        return f"Eu sou {self.nome}, um {self.classe}!"

heroi = Personagem("Link", "Guerreiro")
print(heroi.apresentar())
```





# 07

## Bibliotecas Populares



# Bibliotecas Populares

## Bibliotecas Populares: Ferramentas Extras.

Gerando números aleatórios com *random*

```
thiago-galvão.py

import random
print(random.randint(1, 100)) # Número aleatório entre 1 e 100
```

Trabalhando com APIs usando *requests*

```
thiago-galvão.py

import requests
resposta = requests.get("https://pokeapi.co/api/v2/pokemon/pikachu")
print(resposta.json()["name"]) # pikachu
```



# CONCLUSÃO



## Bibliotecas Populares

Dominar Python é como aprender uma nova linguagem num universo fantástico. Com prática e paciência, você logo estará criando seus próprios projetos e explorando novas possibilidades.

Continue estudando, praticando e, acima de tudo, se divertindo com a jornada. Afinal, programar também é uma forma de criar sua própria aventura!

Agora, é hora de pegar seu teclado, abrir o terminal e começar a codar.

Que a força do Python esteja com você!







# OBRIGADO!



# OBRIGADO POR CHEGAR ATÉ AQUI!

---

Esse Ebook foi gerado por IA, e diagramado por humano.  
O passo a passo se encontra no meu Github

Esse conteúdo foi gerado sob fins didáticos de construção,  
não foi realizado uma validação cuidadosa humana no  
conteúdo e pode conter erros gerados por uma IA.



<https://github.com/thiago-galvao/ia-ebook>

