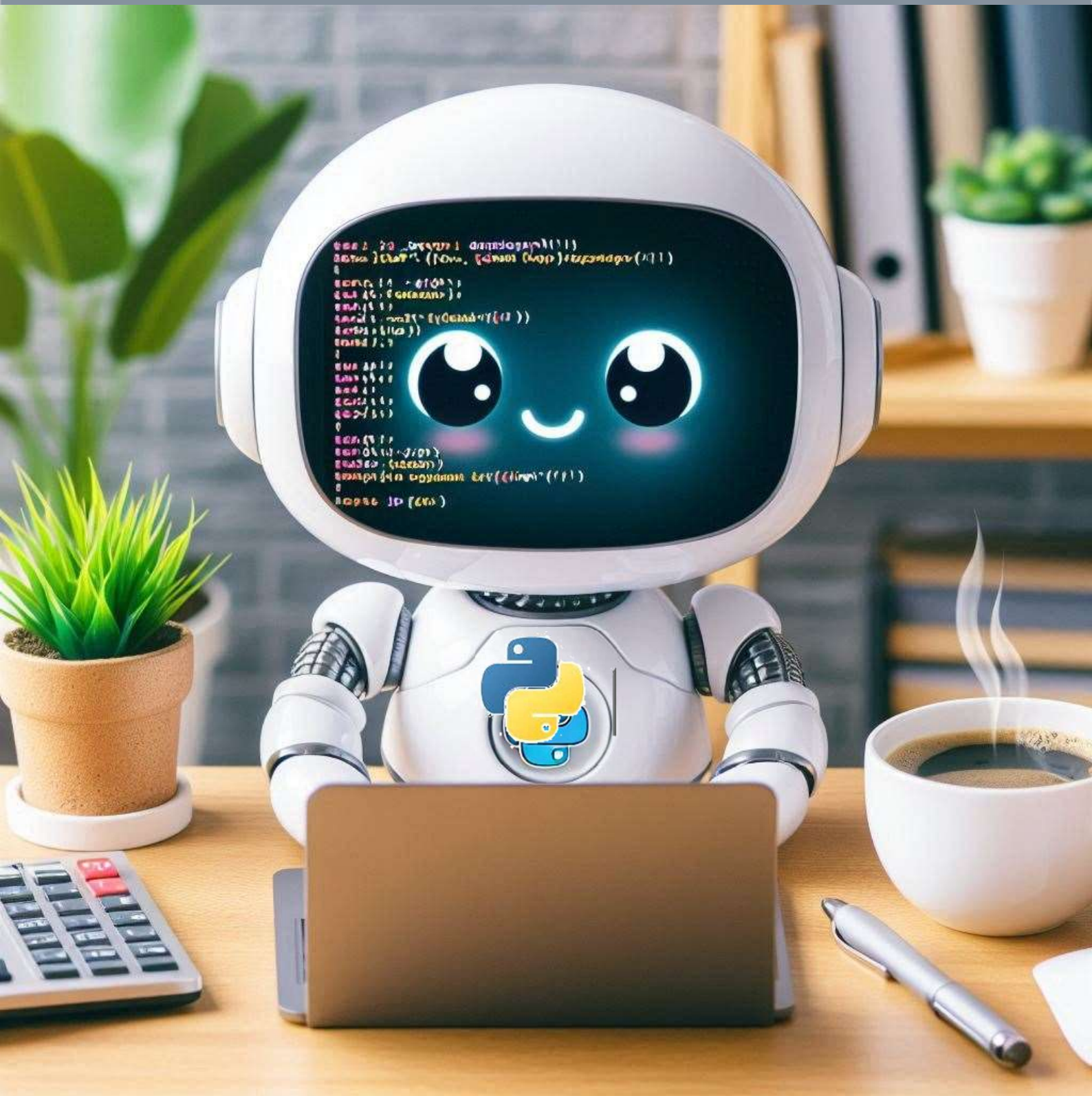


# Introdução ao Python: Comece Sua Jornada na Programação



Andrea Bezerra

# INTRODUÇÃO

Python é uma linguagem de programação poderosa e fácil de aprender. Este e-Book é uma introdução prática ao Python, guiando você através dos conceitos básicos com exemplos reais que você pode aplicar imediatamente. Vamos começar!



# 01

# INTRODUÇÃO AO PYTHON

---

Neste capítulo, você aprenderá os fundamentos da programação em Python. Vamos cobrir desde a instalação do Python até a execução do seu primeiro programa.

# INSTALANDO PYTHON

Para começar, você precisa instalar o Python. Acesse [python.org](https://python.org) e baixe a versão mais recente para o seu sistema operacional. Após a instalação, abra o terminal e digite **'python -version'** para verificar se tudo está funcionando corretamente.

## Passo a Passo

**1. Acesse o site oficial do Python:** [python.org](https://python.org).

**2. Baixe a versão mais recente do Python:** Escolha a versão compatível com seu sistema operacional (Windows, macOS ou Linux).

**3. Instale o Python:** Siga as instruções do instalador. Certifique-se de marcar a opção "Add Python to PATH" durante a instalação no Windows.

# SEU PRIMEIRO PROGRAMA

Vamos criar nosso primeiro programa em Python. Abra o seu editor de texto preferido e digite o seguinte código:



```
print("Olá, Mundo!")
```

## Explicação:

Este simples programa imprime a mensagem **"Olá, Mundo!"** na tela. O comando **'print'** é uma função embutida em Python que exibe a saída para o usuário.

# 02

## VARIÁVEIS E TIPOS DE DADOS

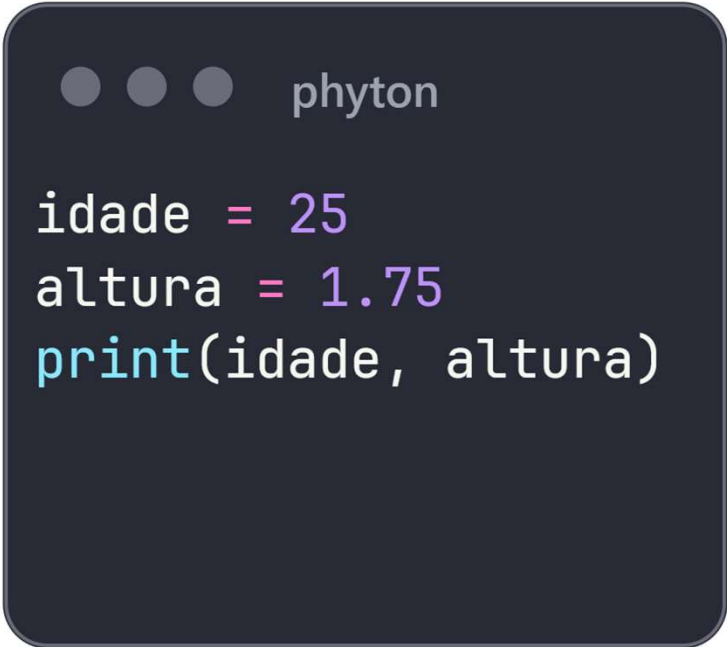
---

Variáveis são usadas para armazenar valores. Em Python, você não precisa declarar o tipo de variável, pois a linguagem é dinamicamente tipada.

# VARIÁVEIS

Python possui vários tipos de dados integrados. Aqui estão alguns dos mais comuns:

## Inteiros e Flutuantes



```
idade = 25
altura = 1.75
print(idade, altura)
```

### Explicação:

‘**idade**’ é um inteiro e ‘**altura**’ é um float (número decimal)

# VARIÁVEIS

## Strings

```
● ● ● phyton  
  
nome = "João"  
print(nome)
```

## Booleanos

```
● ● ● phyton  
  
is_estudante = True  
print(is_estudante)
```



# 03

## Estruturas de Controle

---

Aqui, vamos explorar as estruturas de controle de fluxo, como condicionais e loops, que permitem controlar o comportamento do seu programa.

# CONDICIONAIS

Estruturas condicionais permitem que o programa tome decisões baseadas em condições



```
idade = 20

if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")
```

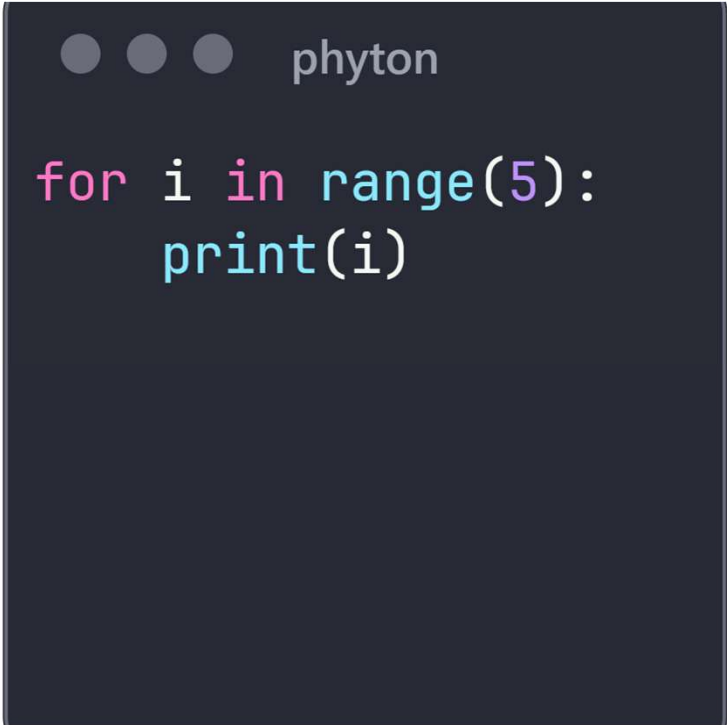
## Explicação:

O código verifica se a variável '**idade**' é maior ou igual a 18. Se for, imprime "Você é maior de idade." Caso contrário, imprime "Você é menor de idade."

# LOOPS

Loops são usados para executar um bloco de código várias vezes.

## Loop 'for'



```
● ● ● phyton  
  
for i in range(5):  
    print(i)
```

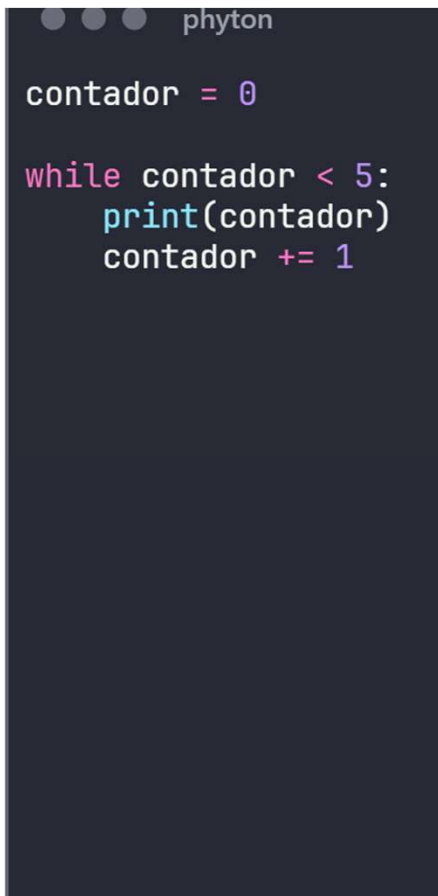
### Explicação:

Este programa imprime os números de 0 a 4. A função `range(5)` gera uma sequência de números de 0 a 4.

# LOOPS

Loops são usados para executar um bloco de código várias vezes.

## Loop 'while'

A screenshot of a terminal window titled 'phyton'. It contains the following Python code:

```
contador = 0

while contador < 5:
    print(contador)
    contador += 1
```

### Explicação:

Este loop **'while'** continua executando enquanto a condição **'contador < 5'** for verdadeira. O valor de **'contador'** é incrementado em 1 a cada iteração, imprimindo os números de 0 a 4.

# 04

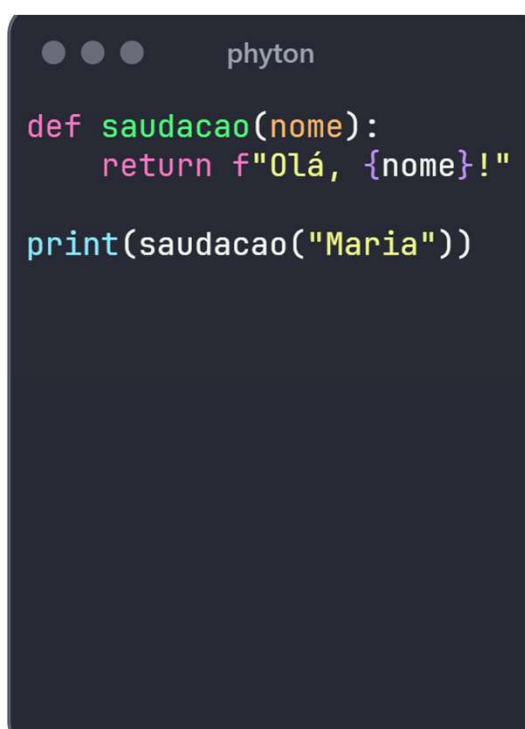
## FUNÇÕES E MÓDULOS

---

Funções e módulos ajudam a organizar e reutilizar código. Neste capítulo, você aprenderá a criar suas próprias funções e a utilizar módulos padrão do Python

# FUNÇÕES

Funções são blocos de código reutilizáveis que realizam uma tarefa específica.



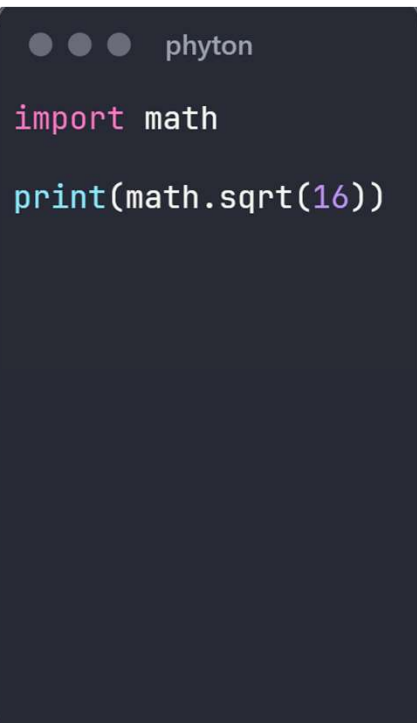
```
def saudacao(nome):  
    return f"Olá, {nome}!"  
  
print(saudacao("Maria"))
```

## Explicação:

A função '**saudação**' recebe um parâmetro 'nome' e retorna uma mensagem de saudação. A função é chamada com o argumento "Maria", resultando na mensagem "Olá, Maria!".

# MODULOS

Módulos são arquivos Python que contêm definições e instruções. Você pode importar módulos para usar suas funcionalidades



```
import math  
print(math.sqrt(16))
```

## Explicação:

Importamos o módulo '**math**' e usamos a função '**sqrt()**' para calcular a raiz quadrada de 16.

# 05

## TRABALHANDO COM DADOS

---

Manipular dados é uma habilidade essencial em programação. Vamos explorar listas, dicionários e a manipulação de arquivos.



# LISTAS

Listas armazenam múltiplos itens em uma única variável.

A screenshot of a Python terminal window with a dark background. The window title is "phyton". It contains the following code: 

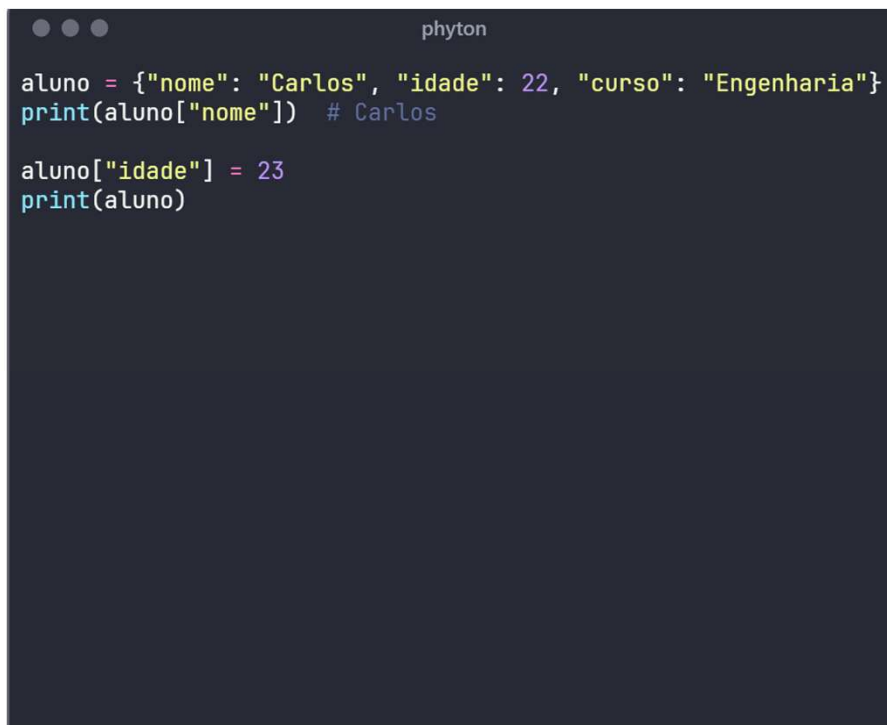
```
frutas = ["maçã", "banana", "cereja"]  
print(frutas[1]) # banana  
  
frutas.append("laranja")  
print(frutas)
```

## Explicação:

Este código cria uma lista de frutas e imprime o segundo item da lista ("banana"). Em seguida, adiciona "laranja" à lista e a imprime novamente.

# DICIONARIOS

Dicionários são coleções de pares chave-valor.

A screenshot of a terminal window titled 'python'. The code inside the terminal is as follows:

```
aluno = {"nome": "Carlos", "idade": 22, "curso": "Engenharia"}  
print(aluno["nome"]) # Carlos  
  
aluno["idade"] = 23  
print(aluno)
```

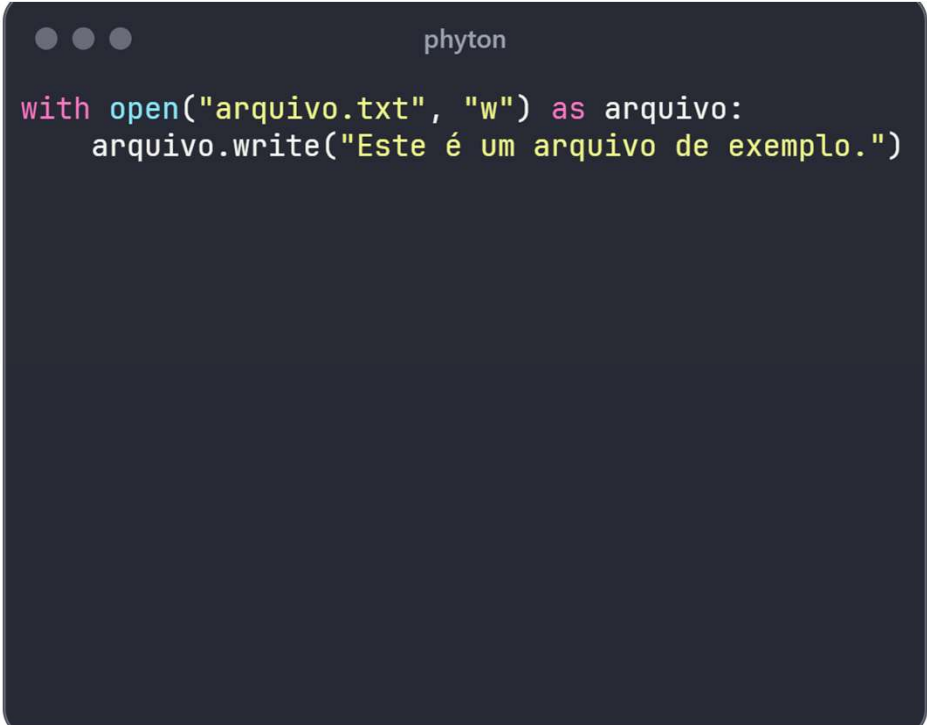
## Explicação:

Este código cria um dicionário chamado 'aluno' com informações sobre um estudante. Imprime o nome do aluno ("Carlos") e depois atualiza a idade para 23.

# MANIPULAÇÃO DE ARQUIVOS

Você pode ler e escrever arquivos usando Python.

**Escrever em um arquivo.**

A screenshot of a dark-themed terminal window titled 'phyton'. It contains two lines of Python code: 'with open("arquivo.txt", "w") as arquivo:' and 'arquivo.write("Este é um arquivo de exemplo.")'.

```
phyton  
  
with open("arquivo.txt", "w") as arquivo:  
    arquivo.write("Este é um arquivo de exemplo.")
```

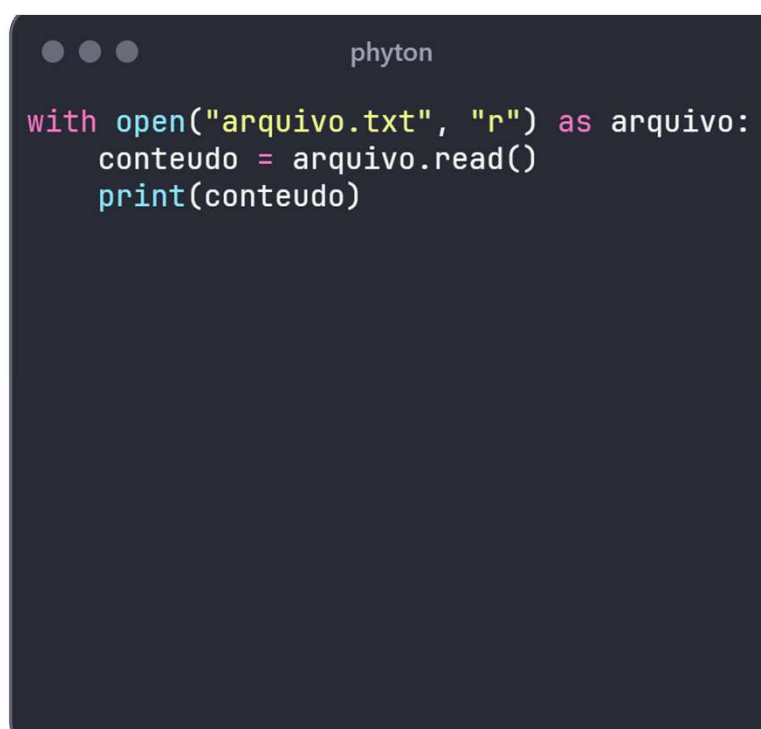
## **Explicação:**

Este código cria e abre um arquivo chamado arquivo.txt no modo de escrita ("w"). Em seguida, escreve a frase "Este é um arquivo de exemplo." no arquivo

# MANIPULAÇÃO DE ARQUIVOS

you can read and write files using Python.

**Read from a file.**

A screenshot of a Python terminal window titled "phyton". The window has three small circles in the top-left corner. The code displayed is: 

```
with open("arquivo.txt", "r") as arquivo:  
    conteudo = arquivo.read()  
    print(conteudo)
```

## **Explicação:**

Este código abre o arquivo 'arquivo.txt' no modo de leitura ("r"), lê seu conteúdo e o imprime na tela.

# 06

## ENTRADA E SAÍDA DE DADOS

---

Neste capítulo, você aprenderá como capturar dados do usuário e como exibir informações de forma interativa. Entrada e saída de dados são essenciais para tornar seus programas mais dinâmicos e interativos.

# ENTRADA DE DADOS

Você pode usar a função input para capturar dados do usuário



```
python
nome = input("Digite seu nome: ")
print(f"Olá, {nome}!")
```

## Explicação:

A função 'input' exibe uma mensagem ao usuário e aguarda a entrada de dados. O valor digitado pelo usuário é armazenado na variável 'nome' e, em seguida, exibido com uma mensagem de saudação.

# CONVERSÃO DE TIPOS

Dados capturados via 'input' são sempre strings. Para realizar operações matemáticas, é necessário converter esses dados para outros tipos, como int ou float.

A screenshot of a terminal window titled 'phyton'. The code inside is: 

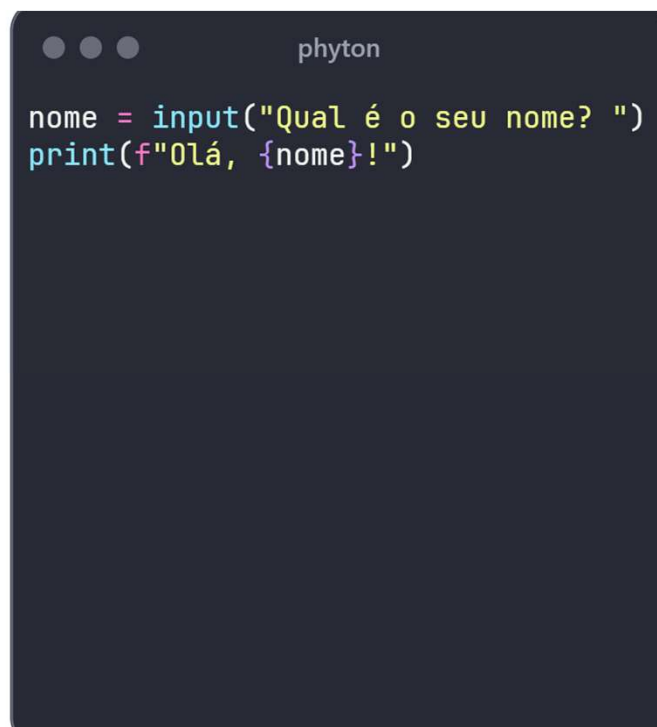
```
idade = int(input("Digite sua idade: "))
ano_nascimento = 2024 - idade
print(f"Você nasceu em {ano_nascimento}.")
```

## Explicação:

O valor digitado pelo usuário é convertido para inteiro usando int. Em seguida, calculamos o ano de nascimento e exibimos o resultado.

# SAIDA DE DADOS

A função print pode ser usada para exibir informações de diferentes maneiras, incluindo formatação



```
python
nome = input("Qual é o seu nome? ")
print(f"Olá, {nome}!")
```

## Explicação

O programa solicita o nome do usuário e imprime uma saudação personalizada.



# 07

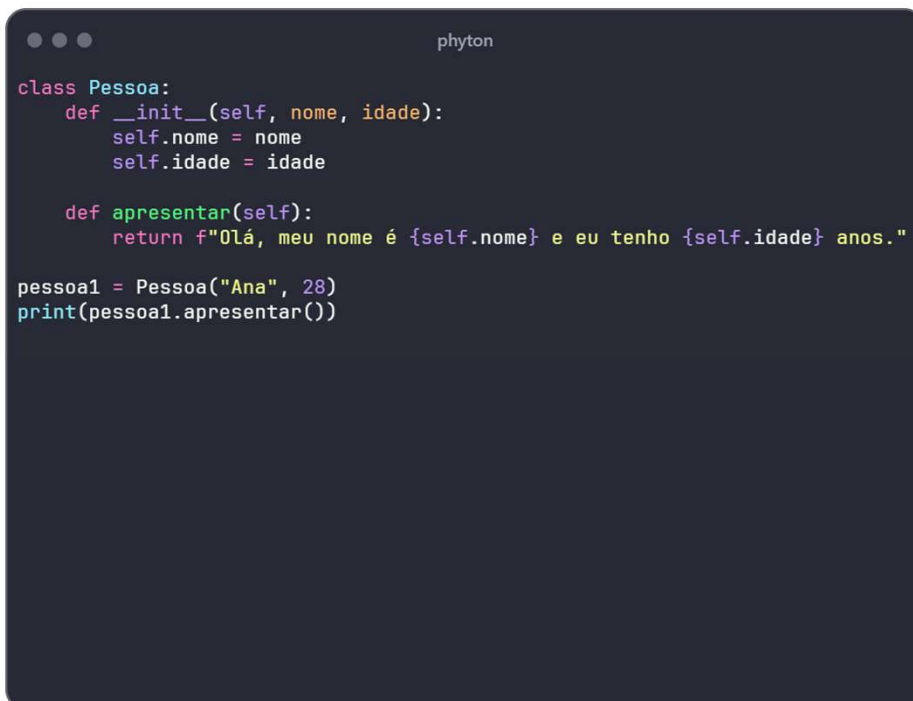
## INTRODUÇÃO À PROGRAMAÇÃO ORIENTADA A OBJETOS (POO)

---

A Programação Orientada a Objetos (POO) é um paradigma de programação que utiliza "objetos" para modelar dados e comportamentos. Neste capítulo, vamos aprender os conceitos básicos de POO em Python.

# CLASSES E OBJETOS

Uma classe é uma estrutura que define um objeto. Um objeto é uma instância de uma classe.



```
python
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def apresentar(self):
        return f"Olá, meu nome é {self.nome} e eu tenho {self.idade} anos."

pessoa1 = Pessoa("Ana", 28)
print(pessoa1.apresentar())
```

## Explicação:

A classe '**Pessoa**' define um objeto com atributos '**nome**' e '**idade**'. O método '**apresentar**' retorna uma mensagem com essas informações. Criamos uma instância da classe '**Pessoa**' e chamamos o método '**apresentar**' para exibir a mensagem.

# HERANÇA

Herança permite que uma classe herde atributos e métodos de outra classe

```
python
class Animal:
    def __init__(self, nome):
        self.nome = nome

    def emitir_som(self):
        pass

class Cachorro(Animal):
    def emitir_som(self):
        return "Au Au"

dog = Cachorro("Rex")
print(dog.emitir_som())
```

## Explicação:

A classe '**Cachorro**' herda da classe **Animal**. O método '**emitir\_som**' é definido na classe '**Cachorro**' para retornar "**Au Au**". Criamos uma instância de '**Cachorro**' e chamamos '**emitir\_som**' para exibir o som do cachorro.

08

# AGRADECIMENTO

---

# OBRIGADO POR LER ATÉ AQUI

Esse book foi gerado por **IA** e diagramado por humano

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizada uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma **IA**



<https://github.com/Andreacbsilva/prompts-recipe-to-create-a-ebook>

**Autor: Andrea Bezerra**