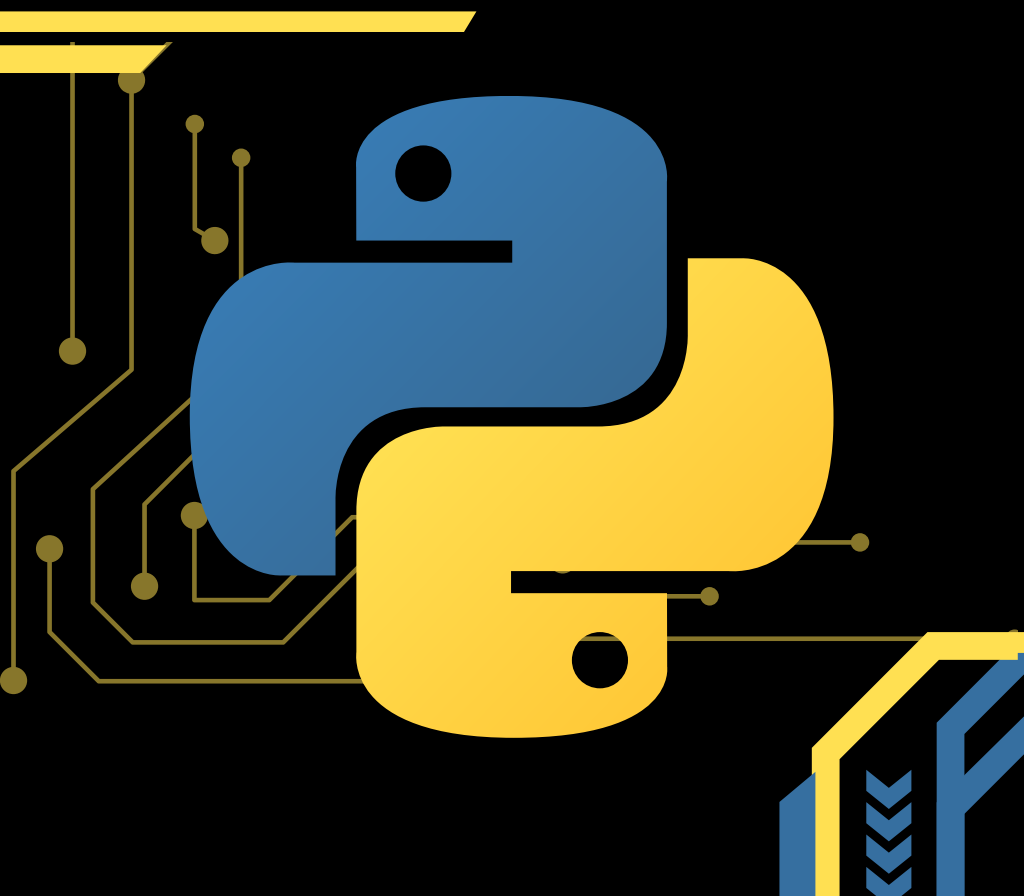


POR RAFAEL PRADO

Python do Zero: **GUIA PRÁTICO** PARA INICIANTES





INTRODUÇÃO

Se você sempre teve vontade de aprender a programar, mas não sabia por onde começar, este eBook foi feito para você. Aqui você aprenderá a programar com Python, uma das linguagens mais populares e acessíveis do mundo.

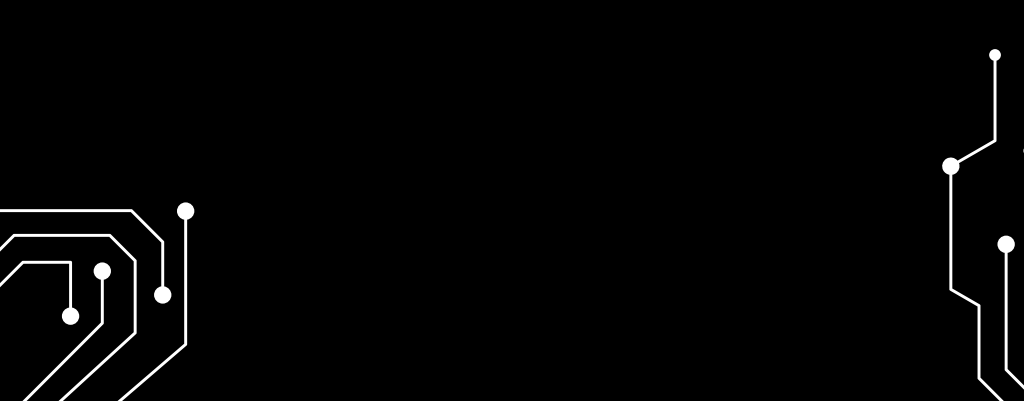
Python é conhecida por sua simplicidade e legibilidade, sendo ideal para iniciantes. Neste guia, vamos explorar os conceitos fundamentais da programação e evoluir até técnicas mais avançadas como a programação orientada a objetos.

Tudo de forma clara, objetiva e com muitos exemplos práticos. Prepare seu computador, sua curiosidade e vamos começar essa jornada!



CAPÍTULO 1: FUNDAMENTOS DA LINGUAGEM

Neste capítulo você aprenderá os blocos mais básicos da programação em Python: variáveis, tipos de dados, entrada e saída de informações e operadores.



VARIÁVEIS E TIPOS DE DADOS

Variáveis são nomes que armazenam dados em memória. Cada dado tem um tipo, como texto (str), números inteiros (int), reais (float) e lógicos (bool).

```
● ● ● Python  
  
nome = "Maria"  
idade = 25  
altura = 1.68  
estudando = True
```

TYPE

Use `type()` para verificar o tipo:

```
● ● ● Python  
  
print(type(nome))
```

ENTRADA E SAÍDA DE DADOS

A função `input()` coleta informações do usuário e `print()` exibe resultados.

```
Python

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

OPERADORES MATEMÁTICOS E OPERADORES LÓGICOS

Servem para realizar cálculos e comparações:

```
Python

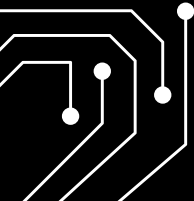
# operadores matemático
soma = 10 + 5
subtracao = 5 - 3
multiplicacao = 8 * 6
divisao = 8 // 2
Resto da divisao = 81 % 9

# operador lógico
igual = 10 == 5
```



CAPÍTULO 2: CONTROLE DE FLUXO

Controle de fluxo permite que seu programa tome decisões e repita ações com base em condições.



CONDICIONAIS

Permitem executar blocos diferentes de código dependendo de condições:

```
Python

idade = int(input("Digite sua idade: "))
if idade > 18:
    print("Maior de idade")
elif idade > 0:
    print("Menor de idade")
else:
    print("Idade inválida")
```

LAÇOS DE REPETIÇÃO

Repetem instruções várias vezes:

```
Python

for i in range(5):
    print(i)

contador = 0
while contador < 5:
    print(contador)
    contador += 1
```

COMPREENSÕES DE LISTA

Forma concisa de criar listas:

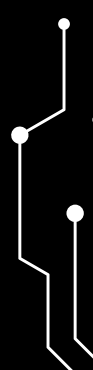
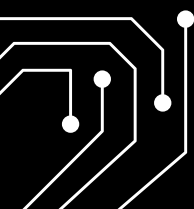
```
Python

quadrados = [x**2 for x in range(10)]
```



CAPÍTULO 3: TRABALHANDO COM DADOS

Aqui aprendemos sobre estruturas que armazenam coleções de dados e como manipulá-las.



LISTAS, TUPLAS, CONJUNTOS E DICIONÁRIOS

Permitem executar blocos diferentes de código dependendo de condições:

- Lista: coleção ordenada e mutável
- Tupla: coleção ordenada e imutável
- Conjunto: coleção não ordenada e sem repetição
- Dicionário: pares chave-valor

Python

```
lista = [1, 2, 3]
tupla = (1, 2, 3)
conjunto = {1, 2, 3}
dicionario = {"nome": "João", "idade": 30}
```

MÉTODOS COMUNS

Manipulam ou acessam dados dessas estruturas:

```
Python  
lista.append(4)  
print(dicionario["nome"])
```

FATIAMENTO

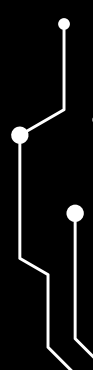
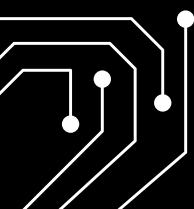
Permite acessar partes de strings e listas:

```
Python  
texto = "Python"  
print(texto[0:3])
```



CAPÍTULO 4: FUNÇÕES

Funções são blocos reutilizáveis de código
que realizam tarefas específicas.



FUNÇÕES

Funções são blocos reutilizáveis de código que realizam tarefas específicas.

```
Python

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

saudacao("Lucas")
```

RETORNO DE VALORES

Funções podem devolver valores para serem usados depois:

```
Python

def soma(a, b):
    return a + b

resultado = soma(3, 4)
```

LAMBDA

Funções pequenas e sem nome:

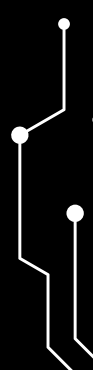
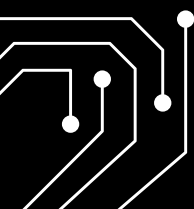
```
Python

dobro = lambda x: x * 2
print(dobro(5))
```



CAPÍTULO 5: MANIPULAÇÃO DE ARQUIVOS

Manipular arquivos é essencial
para salvar e ler dados.



ABRIR E LER ARQUIVOS

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

ESCREVER EM ARQUIVOS

```
Python  
  
with open("arquivo.txt", "w") as arquivo:  
    arquivo.write("Escrevendo no arquivo")
```

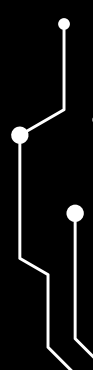
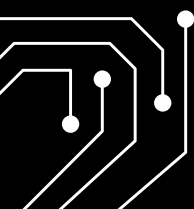
TRATAMENTO DE ERROS

```
Python  
  
try:  
    numero = int(input("Digite um número: "))  
except ValueError:  
    print("Valor inválido!")
```



CAPÍTULO 6: MÓDULOS E PACOTES

Módulos ajudam a organizar e reutilizar código.
Você pode importar funcionalidades prontas
ou criar as suas.



REUTILIZAÇÃO DE CÓDIGO

```
Python

import math
print(math.sqrt(16))

import random
print(random.randint(1, 10))
```

CRIANDO MÓDULOS

```
Python

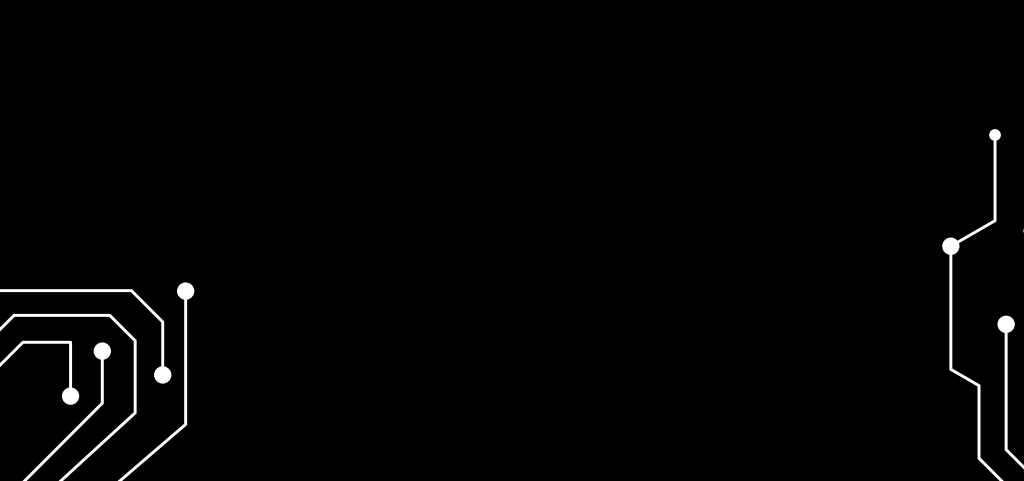
# arquivo: saudacao.py
def dizer_ola():
    print("Olá!")

# outro arquivo
import saudacao
saudacao.dizer_ola()
```




CAPÍTULO 7: INTRODUÇÃO À POO

Programação Orientada a Objetos (POO) é um paradigma que organiza o código com base em objetos do mundo real.



CLASSES E OBJETOS

Classe é um molde, objeto é uma instância dessa classe.

```
Python

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

    def apresentar(self):
        print(f"Meu nome é {self.nome} e tenho {self.idade} anos")

p = Pessoa("Ana", 22)
p.apresentar()
```

HERANÇA

Permite que uma classe herde atributos e métodos de outra:

```
Python

class Aluno(Pessoa):
    def estudar(self):
        print(f"{self.nome} está estudando")

aluno = Aluno("Carlos", 20)
aluno.estudar()
```




CONCLUSÃO

Parabéns! Você concluiu sua introdução à programação com Python. Agora você tem uma base sólida para explorar novas áreas como desenvolvimento web, análise de dados, automação e muito mais.

Continue praticando, criando projetos pessoais e participando de comunidades. A melhor forma de aprender é praticando.

Boa jornada!





RAFAEL PRADO

.....



[raffaprado](#)



[raffaprado](#)