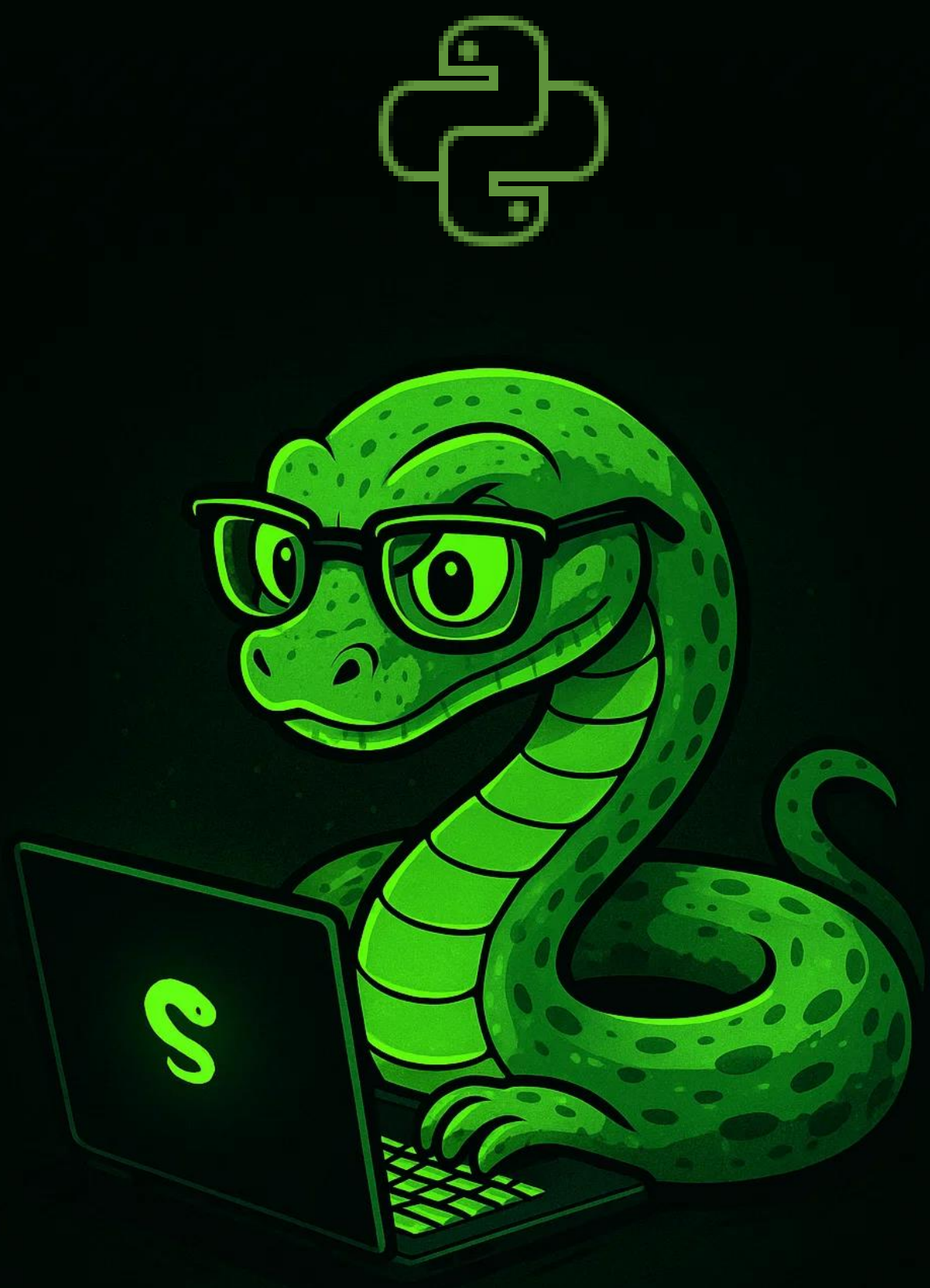
TRUQUES DA SERPENTE



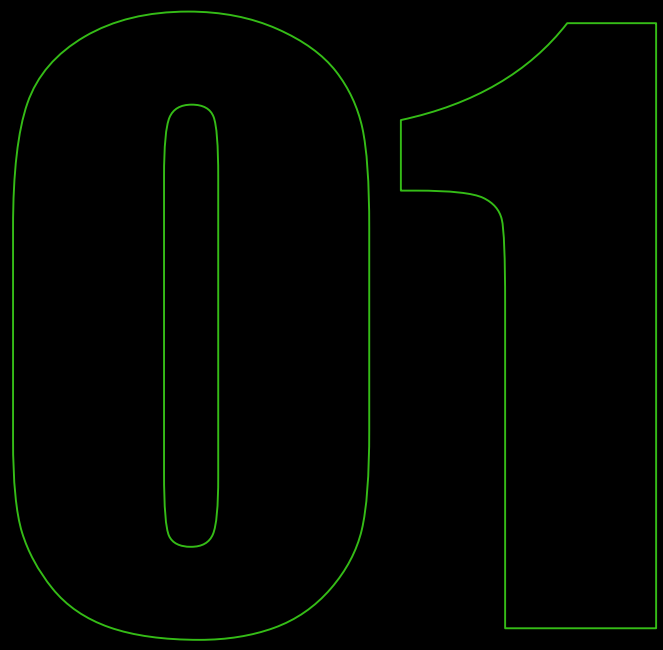
cada dica é como uma mordida precisa da cobra — certeira e eficiente.

ADILAN COSTA

HACKS E MACETES DE PYTHON

O Que Todo Dev Deveria Saber

Python é uma linguagem poderosa e simples. Mas existe um conjunto de atalhos, truques e padrões inteligentes que tornam seu código mais rápido, limpo e profissional. Vamos explorar os principais com exemplos claros e aplicáveis no mundo real.



INVERTENDO UMA LISTA COM SLICING

Imagine que você tem uma lista de nomes de usuários que chegaram em ordem cronológica e deseja exibi-los do último para o primeiro:

INVERTENDO UMA LISTA COM SLICING

Usar slicing (**[::-1]**) é muito mais rápido e limpo do que escrever um loop para inverter uma lista. Ideal para históricos, logs e reversão de dados:

truques.py

```python
usuarios = ['Ana', 'Bruno', 'Carlos', 'Diana']
usuarios_recentes = usuarios[::-1]
print(usuarios_recentes)
# Saída: ['Diana', 'Carlos', 'Bruno', 'Ana']
```

02

TROCA DE VARIAVEIS SEM VARIÁVEL TEMPORÁRIA

Você está ajustando a lógica de um jogo e precisa inverter as posições de dois jogadores:

TROCA DE VARIÁVEIS SEM VARIÁVEL TEMPORÁRIA

É um macete simples, mas muito útil em lógica de jogos, algoritmos e trocas rápidas.
No Python, você pode trocar o valor de duas variáveis sem precisar de uma variável temporária. Exemplo:

truques.py

```python
jogador1 = "Alice"
jogador2 = "Bob"
jogador1, jogador2 = jogador2, jogador1

print(jogador1, jogador2)  # Bob Alice
```

03

VERIFICAÇÃO SIMPLES COM OPERADOR IN

Você está desenvolvendo uma verificação em um formulário de cadastro. Quer saber se a cidade do usuário está na lista de cidades atendidas:

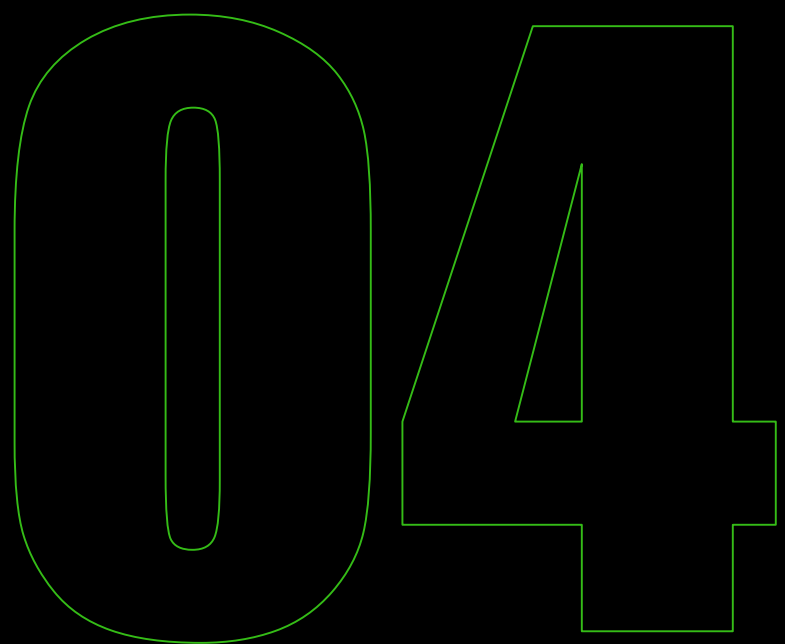
VERIFICAÇÃO SIMPLES COM OPERADOR IN

Em vez de usar um for para percorrer toda a lista, você pode usar o operador in, que faz isso de forma automática e limpa.
in verifica se um valor está em uma lista e elimina a necessidade de loops.
Perfeito para validações, filtros e buscas diretas:

truques.py

```python
cidades_atendidas = ['São Paulo', 'Rio de Janeiro', 'Curitiba']
cidade_usuario = 'Curitiba'

if cidade_usuario in cidades_atendidas:
    print("Entrega disponível para sua cidade!")
```



ENUMERATE() PARA LISTAS COM ÍNDICES

Você está criando um sistema de ranking em um campeonato. Tem uma lista com os nomes dos jogadores classificados, e quer mostrar o nome de cada um com sua posição no pódio:

ENUMERATE() PARA LISTAS COM ÍNDICES

enumerate(jogadores) retorna cada item da lista junto com seu **índice (posição).**
start=1 faz com que a contagem comece no 1, e não no zero (útil para rankings, já que ninguém fica em "0º lugar")
O loop pega: **posicao**: o índice (posição do jogador) e **nome**: o nome do jogador na lista:

truques.py

```python
jogadores = ['Lucas', 'Paula', 'Marcos']
for posicao, nome in enumerate(jogadores, start=1):
    print(f"{posicao}º lugar: {nome}")
```

05

CONDICIONAL EM UMA LINHA (IF TERNÁRIO)

Você está criando um sistema de compras. A regra de negócio é: Se o valor da compra for maior ou igual a R$100, o frete é grátis. Caso contrário, o frete custa R$10:

CONDICIONAL EM UMA LINHA (IF TERNÁRIO)

Essa **Única linha (If ternário)**: frete = "Grátis" if valor >= 100 else "R$ 10,00“ significa literalmente: Se valor >= 100, então frete recebe "Grátis".
Senão, frete recebe "R$ 10,00"

truques.py

```python
valor = 120
frete = "Grátis" if valor >= 100 else "R$ 10,00"
print(f"Frete: {frete}")
```

06

ZIP()
PARA JUNTAR LISTAS

Você tem uma lista de produtos e outra com seus respectivos preços. Precisa exibir isso de forma organizada:

ZIP()
PARA JUNTAR LISTAS

O zip() combina as duas listas item por item:
1º de produtos com 1º de preços
2º com 2º, e assim por diante.
É ideal quando duas listas têm relação entre si (como nome e valor, aluno e nota, etc).

truques.py

```python
produtos = ['Mouse', 'Teclado', 'Monitor']
precos = [50, 120, 900]

for produto, preco in zip(produtos, precos):
    print(f"{produto} - R${preco}")
```

07

SET()
PARA ELIMINAR DUPLICATAS

Você recebe uma lista de e-mails, mas alguns se repetem.
Precisa limpá-la:

SET()
PARA ELIMINAR DUPLICATAS

set() converte a lista em um conjunto, que não permite elementos duplicados
Depois, usamos **list()** para voltar a ter uma lista comum
Atenção: o set pode mudar a ordem dos elementos

truques.py

```python
emails = ['a@email.com', 'b@email.com', 'a@email.com']
emails_unicos = list(set(emails))
print(emails_unicos)
```

08

PROTEGENDO DICIONARIOS COM GET()

Você acessa um dicionário com dados de usuário, mas nem todos os usuários têm e-mail registrado:

PROTEGENDO DICIONÁRIOS COM GET()

get() tenta buscar a chave 'email’
Se não existir, não dá erro: retorna o valor padrão ('Email não informado’)
Vantagem: Evita erros como KeyError e deixa o código mais seguro e previsível.

truques.py

```python
usuario = {'nome': 'João', 'idade': 27}
email = usuario.get('email', 'Email não informado')
print(email)
```

09

F-STRINGS PARA FORMATAÇÃO MODERNA

Você quer exibir dados com valores embutidos em frases de forma clara:

F-STRINGS
PARA FORMATAÇÃO MODERNA

O f antes das aspas indica que dentro da string **você pode usar variáveis diretamente entre chaves** {}
{nome} será substituído por "**Mariana**“
{vendas} será substituído por **3500**

truques.py

```python
nome = "Mariana"
vendas = 3500
print(f"{nome} vendeu R${vendas} este mês.")
```

10

DEBUG SIMPLES COM BREAKPOINT()

Você está tendo erro numa função e quer pausar a execução exatamente onde o erro pode estar, para inspecionar variáveis:

DEBUG SIMPLES COM BREAKPOINT()

Quando o Python encontra breakpoint(),ele pausa a execução e abre o terminal para você inspecionar o que está acontecendo (como um raio-X do código em tempo real).
Ideal para: Testar lógica passo a passo, investigar bugs sem precisar de ferramentas externas e substitui o uso de vários print()

truques.py

```python
def calcular_total(produtos):
    total = sum(produtos)
    breakpoint()  # Abre um modo interativo de debug
    return total

print(calcular_total([10, 20, 30]))
```

AGRADECIMENTOS

OBRIGADO POR LER 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 com base da aula e ebook do professor Felipão, modificado, atualizado e revisado por mim para área de python.

https://github.com/adilanlf/desafio-criando-ebook-CHATGPT

https://www.linkedin.com/in/adilan-costa-62173434b/

https://www.duolingo.com/profile/AdilanCost

AUTOR: *ADILAN COSTA*