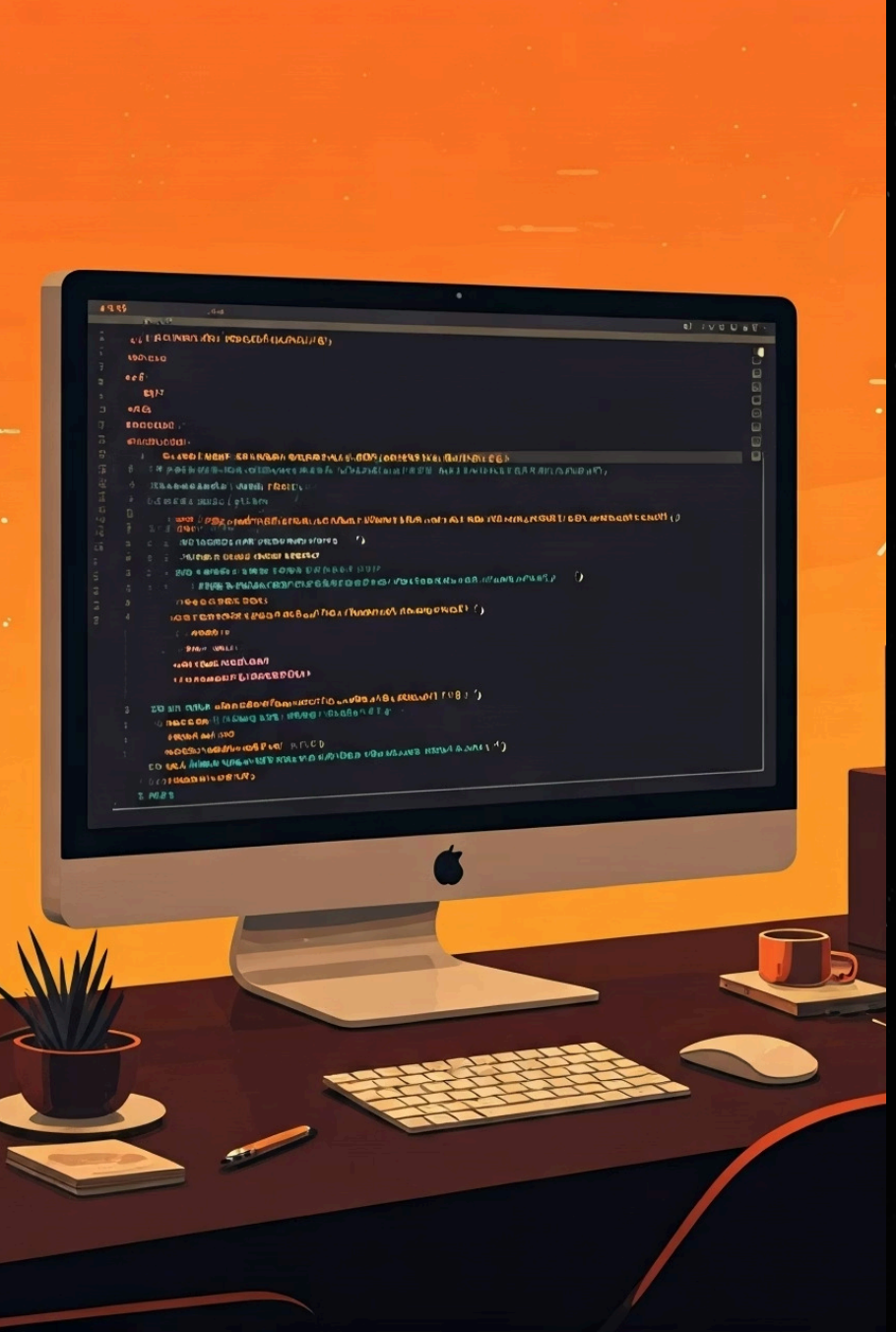


Laço de Repetição while em Python

Uma aula prática completa sobre loops while, condicionais e controle de fluxo para iniciantes

O que vamos aprender hoje

Laços while básicos

Executar código repetidamente até uma condição ser falsa

While True infinito

Criar loops que só param com break

Condicionais no loop

Usar if, elif e else para tomar decisões

Interação dinâmica

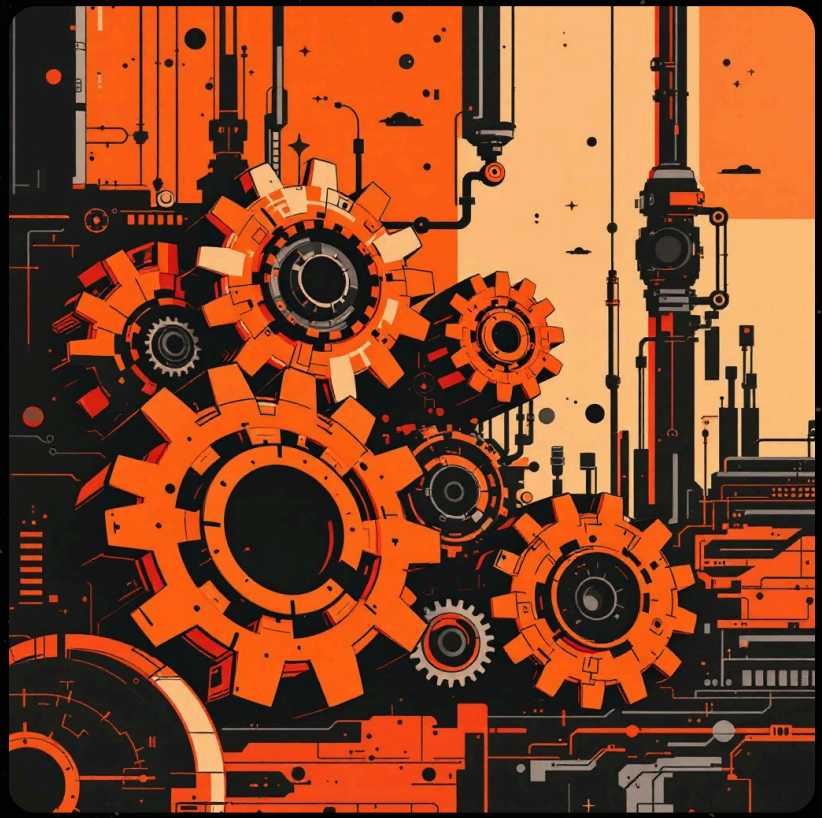
Construir programas que respondem ao usuário

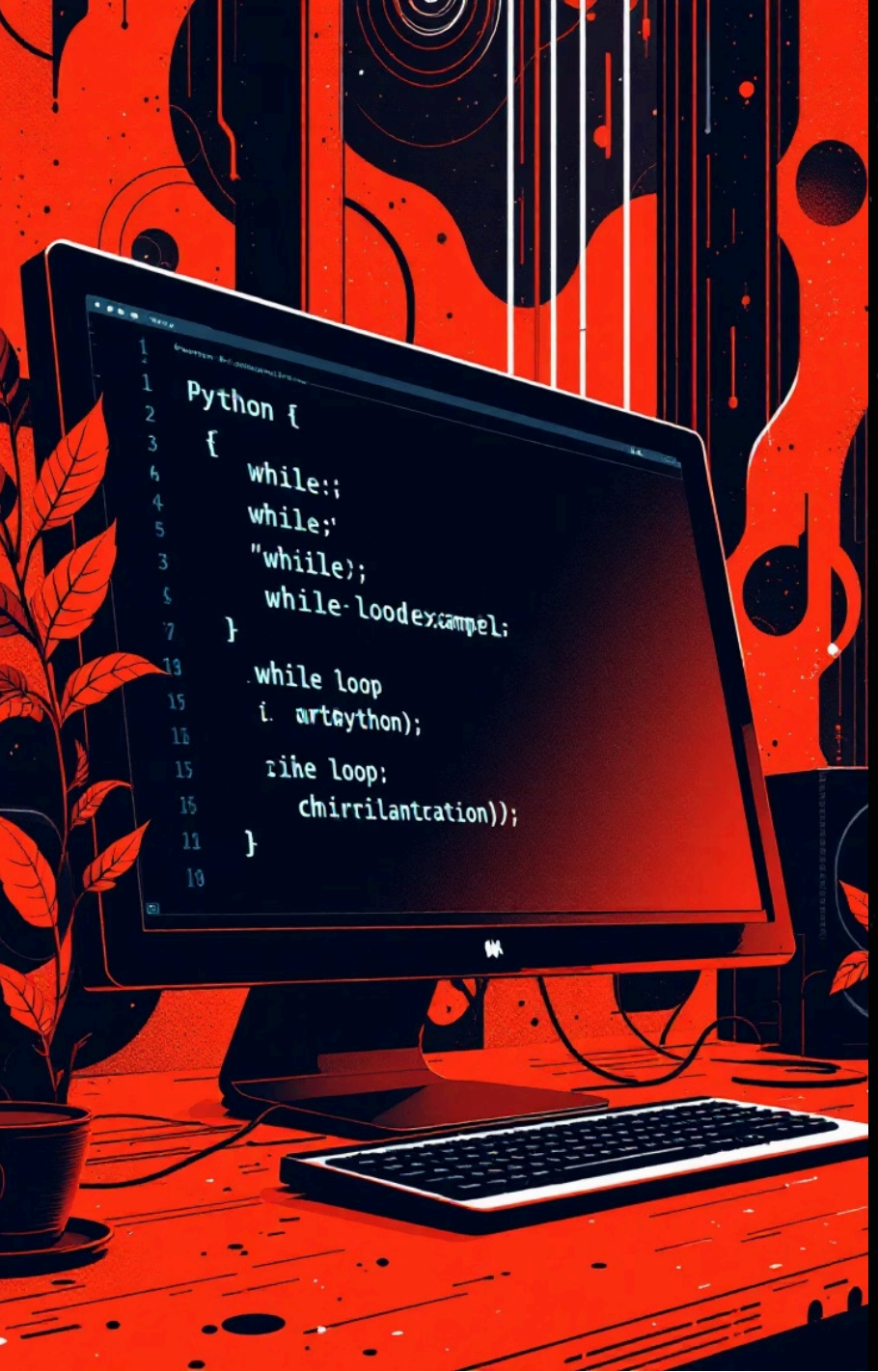
Por que usar laços while?

Os laços while são fundamentais na programação porque permitem **automatizar tarefas repetitivas** de forma inteligente.

Diferente do for, o while continua executando enquanto uma condição for verdadeira, sendo perfeito para situações onde não sabemos quantas repetições serão necessárias.

É como dizer: "continue fazendo isso até que algo específico aconteça".





Estrutura básica do while

Sintaxe simples

```
while condição:
    # código a repetir
    # atualizar variável
```

Exemplo prático

```
contador = 1
while contador <= 5:
    print(f"Contagem: {contador}")
    contador += 1
```

Importante: Sempre lembre de atualizar a variável de controle para evitar loops infinitos acidentais!

While True: O loop infinito controlado

Como funciona

O **while True** cria um loop que roda para sempre, até encontrar um comando **break**.

É muito útil para menus de programa, jogos ou quando queremos que o usuário escolha quando parar.

```
while True:
    resposta = input("Continue? (s/n): ")
    if resposta == 'n':
        break
    print("Continuando...")
```

O **break** é nossa "porta de saída" do loop infinito!

Condicionais dentro do while

if - primeira verificação

Testa a primeira condição e executa o código se for verdadeira

elif - alternativas

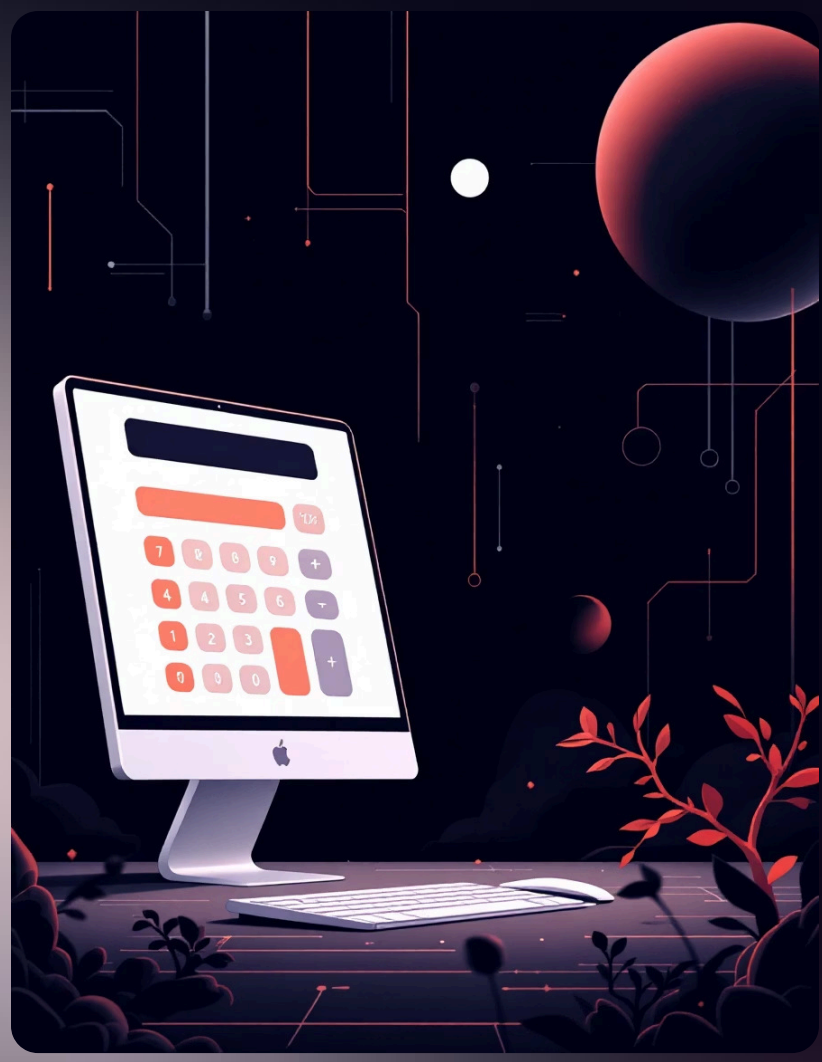
Testa outras condições se a primeira for falsa

else - caso padrão

Executa quando nenhuma condição anterior foi verdadeira

Exemplo prático: Calculadora simples

```
while True:

operacao= input("Digite a operação (+,-,*,/) ou 'sair' para encerrar)

if operacao.lower() == "sair":
print("Encerrando a calculadora")
break

try:
num1 = float(input("Digite o primeiro numero:")
num2= float(input("Digite o segundo numero:")


if operacao ==  "+":
    resultado = num1 + num2

elif operacao == "-":
     resultado == num1 - num2
elif operacao == "*":
    resultado == num1 * num2
elif operacao == "/"
    resultado == num1 / num2
elif operacao == "/":
   if num2 != 0:
    resultado = num1 / num2
else:
    print("Erro: Divisão por zero!")
   continue
else:
    print("Operação invalida. Tente novamente.")
   continue
print(f"resultado: {resultado}")
```



Vantagens dos loops while + condicionais

Repetição automática

Executa tarefas sem precisar reescrever código

Tomada de decisões

O programa reage de forma diferente a cada situação

Interação dinâmica

Responde às escolhas do usuário em tempo real

Controle total

Você decide quando e como o programa deve parar

Exercícios para praticar

1

Menu de Opçôes

Crie um programa que peça dois números e uma operação (+, -, *, /).

- Use `while True` para permitir que o usuário faça várias contas.
- Digitar `"sair"` deve encerrar o programa.

2

Soma até Zero

Peça números ao usuário em um `while True`.

- Vá somando todos os números digitados.
- Quando o usuário digitar `0`, mostre a **soma final** e encerre o programa.

3

Jogo de Adivinhação

O computador "pensa" em um número secreto entre 1 e 10.

- O usuário deve adivinhar usando `while True`.
- Se errar, mostre `"Tente novamente"`.
- Se acertar, mostre `"Parabéns!"` e finalize.

4

Contador de Vogais

Solicite uma frase ao usuário.

- Percorra cada caractere usando `while`.
- Conte e exiba quantas **vogais** existem na frase.

5

Tabuada Interativa

Crie um programa que peça um número ao usuário.

- Exiba a **tabuada desse número** de 1 a 10 usando `while`.

6

Controle de Caixa

Faça um programa que simule o caixa de uma loja.

- O usuário digita valores de produtos infinitamente.
- Se digitar `0`, o programa mostra o **total da compra**.
- Se digitar um valor **negativo**, o programa deve avisar `"Valor inválido"` e pedir outro número.

Próximos passos

Agora você domina os conceitos fundamentais de **loops while** e **condicionais** em Python!

Com essas ferramentas, você pode criar programas que:

- Respondem de forma inteligente às ações do usuário
- Executam tarefas repetitivas automaticamente
- Tomam decisões baseadas em condições
- Controlam o fluxo de execução do código

Continue praticando com os exercícios propostos e logo você estará criando programas cada vez mais complexos e úteis!