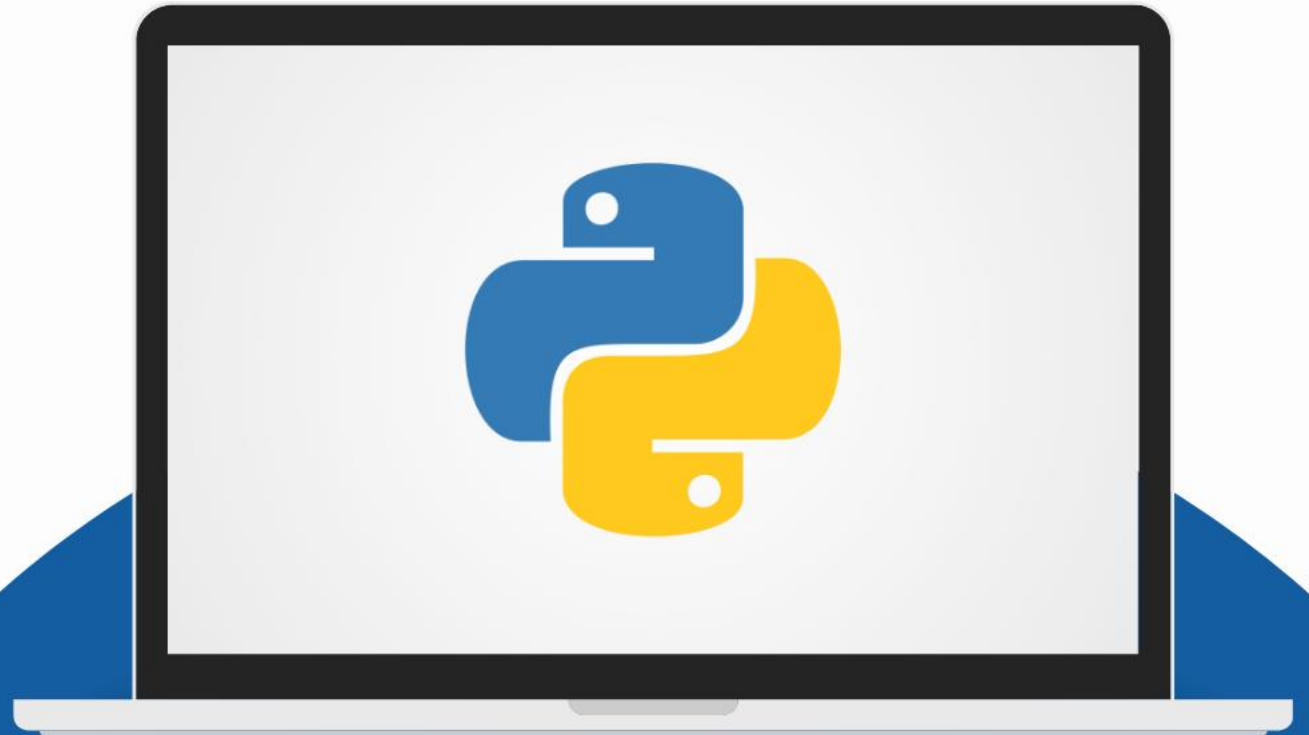


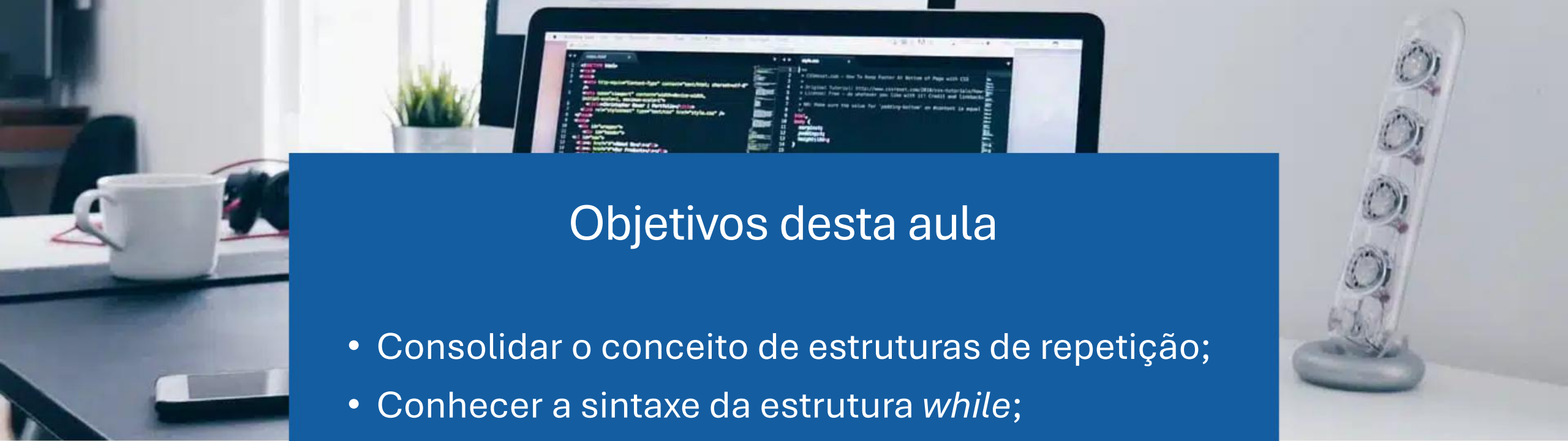


Aula #7:

Estruturas de repetição – o *while*

Prof. Dr. Luiz Álvaro de Oliveira Júnior





Objetivos desta aula

- Consolidar o conceito de estruturas de repetição;
- Conhecer a sintaxe da estrutura *while*;
- Conhecer os modificadores de fluxo *break*, *continue* e *pass*;
- Aplicar a estrutura de repetição *while* e os modificadores de fluxo na resolução de problemas;

Sumário

▶ Estruturas while	04
▶ Exemplos	07
▶ Modificadores de fluxo	11
▶ Exercícios	15



A estrutura **while**

A estrutura **while** é um laço de repetição (ou loop) usado para executar um bloco de código repetidas vezes enquanto uma condição for verdadeira.

A condição é avaliada antes de cada repetição. Se a condição retornar verdadeiro (True), o bloco de código é executado. Depois de executar o bloco, a condição é verificada novamente e, se permanecer verdadeira, o bloco é novamente executado.

O interpretador sai do loop quando a condição retorna falso (False).

A estrutura **while**

A estrutura **while** é normalmente **utilizada quando não se sabe ao certo quantas vezes o bloco deverá ser executado e/ou** existe uma **condição** clara **que precise ser atendida** para que o bloco de código seja executado.

No uso da estrutura **while**, não se pode esquecer de **alterar o valor da variável usada na condição ao final de cada iteração**. Isso evita a criação de **loops infinitos** quando este não é o propósito do programa e evita travamentos indesejados.

A estrutura **while**

Assim como a estrutura **for**, a estrutura **while** também faz uso das indentações para explicitar o conjunto de instruções a serem repetidas a cada nova iteração.

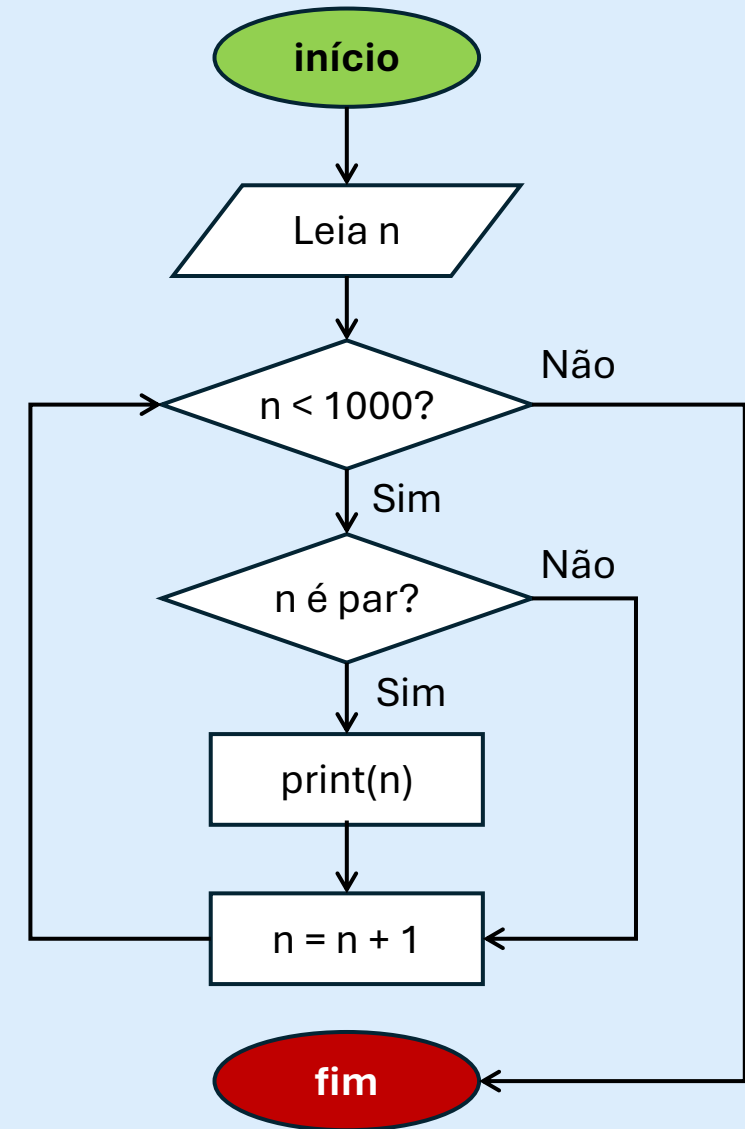
Quando o propósito é criar loops infinitos, é muito comum usar um booleano (True ou False) ou uma variável que não será alterada no código em tempo de execução.

Exemplos

#1: Imprimir na tela os números inteiros pares entre um número escolhido pelo usuário e 1000 (inclusive) usando a estrutura while.



```
n = int(input("Digite o número: "))  
while n <= 1000:  
    if n % 2 == 0:  
        print(n)  
    n += 1
```

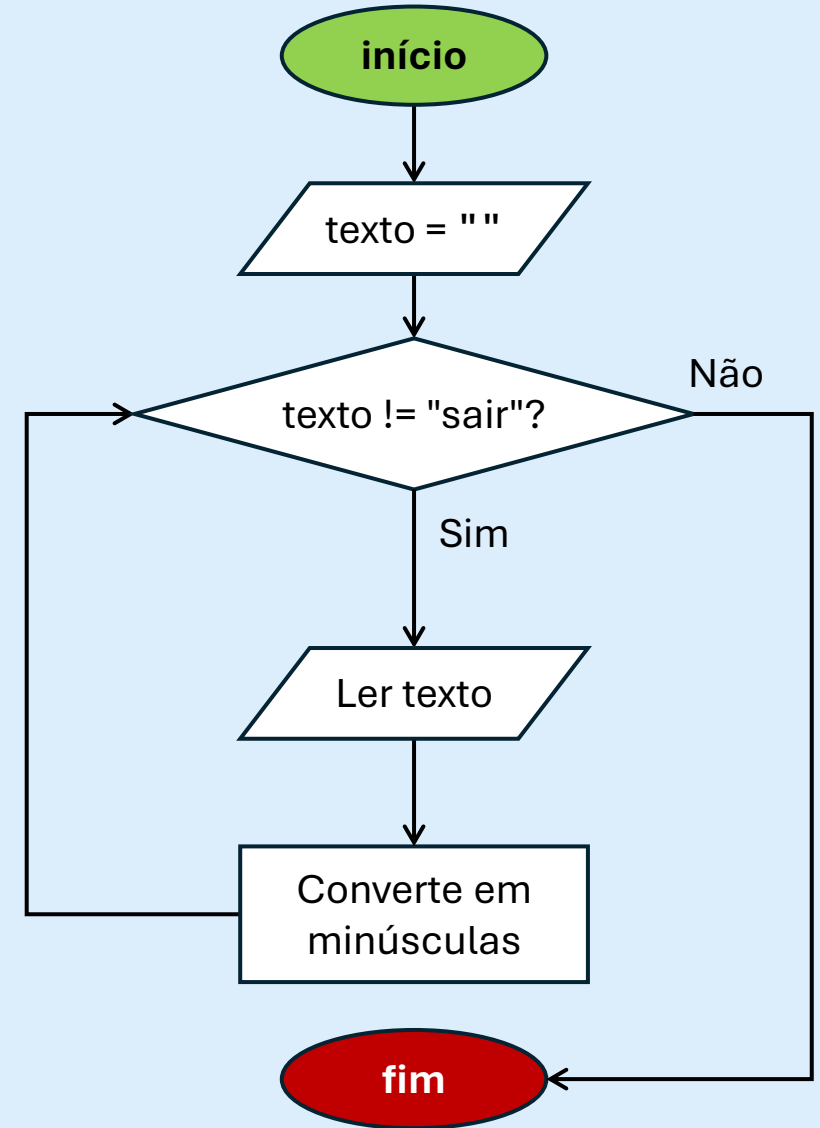


Exemplos

#2: Escrever um programa que peça ao usuário para digitar algo até que o usuário digite sair.

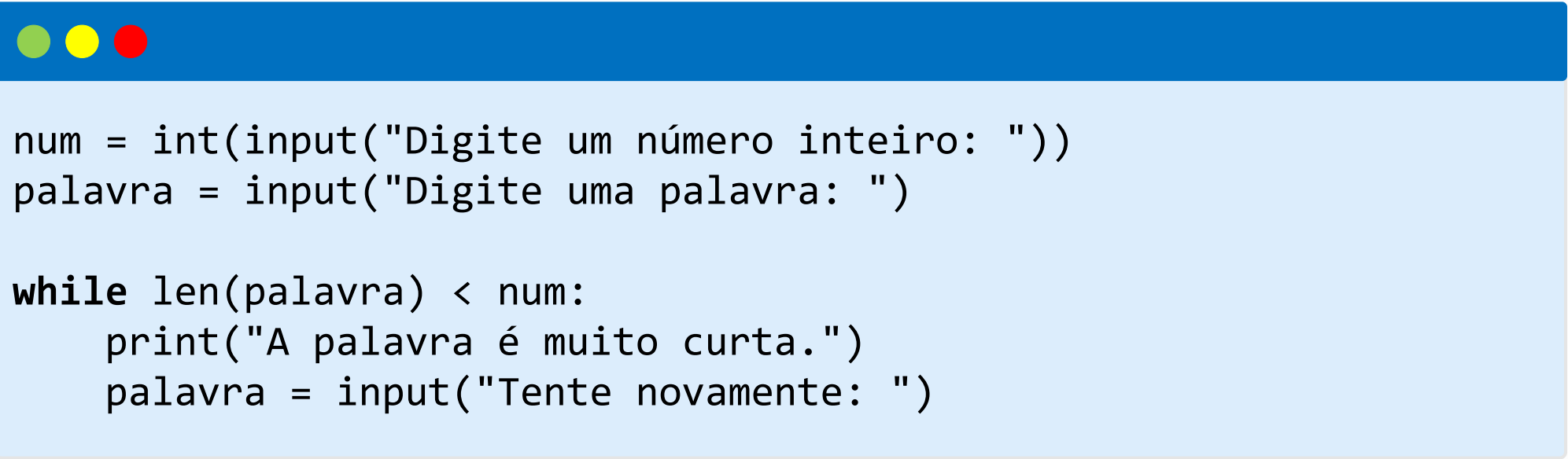


```
texto = ""    #vazio
while texto != "sair":
    texto = input("Digite algo: ")
    texto = texto.lower()
```



Exemplos

#3: Escreva um programa peça um número ao usuário e, em seguida, solicite que ele digite uma palavra enquanto a palavra digitada não tiver mais letras do que o número informado.

A screenshot of a terminal window with a blue title bar and three colored window control buttons (green, yellow, red) on the left. The terminal has a light blue background and contains Python code for a word length validation program.

```
num = int(input("Digite um número inteiro: "))
palavra = input("Digite uma palavra: ")

while len(palavra) < num:
    print("A palavra é muito curta.")
    palavra = input("Tente novamente: ")
```

Exemplos

#4: Usando a estrutura while, escreva um programa que calcule a média das notas de um aluno. Interrompa digitando -1.

```
soma = 0, n = 1
nota = float(input("Digite a nota ou -1 para sair: "))
while nota != -1:
    soma += nota
    n += 1
    nota = float(input("Digite a nota ou -1 para sair: "))
if n > 0:
    media = soma / n
else:
    print("Nenhuma nota foi digitada.")
print(f"Média: {media:.1f}")
```

Modificadores de fluxo

Em algumas situações, é necessário modificar o fluxo padrão de execução, especialmente em estruturas de repetição. Isso pode ser feito por meio dos modificadores de fluxo **break**, **continue** e **pass**.

Vejamos alguns exemplos.

Modificadores de fluxo

O **break** encerra o laço imediatamente. É usado **quando se deseja encerrar o loop quando alguma condição específica** for satisfeita.



```
contador = 1
while True:
    print("Contador:", contador)
    if contador == 3:
        break
    contador += 1
```



```
Contador: 1
Contador: 2
Contador: 3
```

Modificadores de fluxo

O **continue** interrompe a iteração atual e volta ao início do loop. É usado **quando se deseja ignorar um caso específico mas continuar o loop** normalmente.

```
numero = 0
while numero < 5:
    numero += 1
    if numero == 3:
        continue
    print("Número:", numero)
```

Saída

```
Número: 1
Número: 2
Número: 4
Número: 5
```

Modificadores de fluxo

O **pass** não executa nenhuma ação (literalmente) e serve como preenchimento de bloco de código **quando nada precisa ser feito naquele momento**. É útil, por exemplo, quando algo ainda será implementado, ajudando a evitar erros de sintaxe.

```
texto = "entrar"
while texto != "sair":
    texto = input("Digite algo: ")
    if texto == "":
        pass
    else:
        print("Você digitou:", texto)
```

Se o usuário pressionar a tecla enter sem digitar nada, o input retorna uma string vazia, atendendo a condição do if e executando o pass. Veremos na tela a string do input, somente.

Exercícios

Exercício #1

Escreva um programa em Python que conte os números **múltiplos de 2 ou 7 menores que 1000**. O programa deverá imprimir na tela a contagem e a soma deles. Use condicional simples.

```
soma, cont, num = 0, 0, 0
while num < 1000:
    if num % 2 == 0 or num % 7 == 0:
        soma += num
        cont += 1
    num += 1
print(f"Contagem: {cont}, Soma: {soma}.")
```


Exercício #2

Escreva um programa que peça repetidamente ao usuário para digitar uma senha. Quando a senha correta ("python123") for digitada, o programa para e exibe a mensagem: "Acesso permitido.". Use **break**.



```
while True:
    senha = input("Digite a senha: ").lower()
    if senha == "python123":
        print("Acesso permitido.")
        break
```

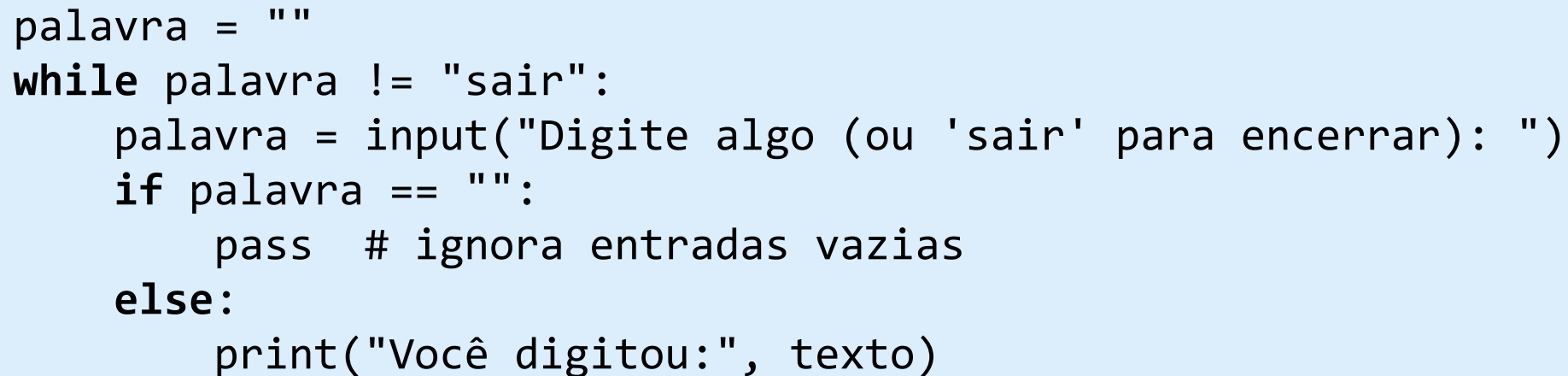
Exercício #3

Escreva um programa que peça números ao usuário, ignore os negativos e exiba o dobro dos positivos. Pare se o número for 0. Use **break** e **continue**.

```
while True:
    numero = int(input("Digite um número (0 para sair): "))
    if numero == 0:
        print("Encerrando o programa.")
        break # Finaliza o laço
    elif numero < 0:
        continue # Pula para a próxima iteração sem fazer nada
    else:
        print("O dobro de", numero, "é", numero * 2)
```

Exercício #4

Escreva um programa em Python que peça palavras ao usuário até que ele digite fim. Ignore entradas vazias.

A code snippet is displayed within a light blue rectangular box that mimics a terminal window. The box has a dark blue header bar at the top containing three small circles in green, yellow, and red, representing window control buttons. The code is written in a monospaced font and implements a loop that prompts the user for input. If the input is empty, it skips the loop body. If the input is 'sair', it exits the loop. Otherwise, it prints the input and continues the loop.

```
palavra = ""  
while palavra != "sair":  
    palavra = input("Digite algo (ou 'sair' para encerrar): ")  
    if palavra == "":  
        pass # ignora entradas vazias  
    else:  
        print("Você digitou:", texto)
```

Exercício #5

Crie um programa que simule um sistema de mensagens. O programa deve:

- Continuar pedindo ao usuário para digitar uma mensagem até ele digitar "sair". Use **break** para encerrar.
- Se o usuário pressionar apenas Enter (mensagem vazia), o programa deve ignorar a entrada com o **pass**.
- Se a mensagem contiver a palavra "ignorar" (em qualquer lugar do texto), não exiba nada e vá para a próxima iteração com **continue**.
- Para qualquer outra entrada, o programa deve exibir a mensagem digitada com um prefixo: Mensagem recebida:..

Exercício #5

```
while True:
    mensagem = input("Digite a mensagem (ou 'sair' para encerrar): ")

    if mensagem == "sair":
        print("Encerrando o programa.")
        break
    elif mensagem == "":
        pass # Não faz nada, apenas continua
    elif "ignorar" in mensagem:
        continue # Pula para o próximo ciclo sem imprimir
    else:
        print("Mensagem recebida:", mensagem)
```

