

# Aula 10 — While: Laços de Repetição (Parte 2)

## Tabela de Conteúdos


1. [Objetivos da Aula](#)
  2. [Revisão Rápida: O que é o `while`](#)
  3. [Estrutura Geral e Lógica do Laço](#)
  4. [Exemplo 1 — Contagem simples](#)
  5. [Exemplo 2 — Condição de parada com número](#)
  6. [Exemplo 3 — Controle com resposta do usuário](#)
  7. [Exemplo 4 — Contagem de pares e ímpares](#)
  8. [Conceitos Importantes: `break`, `continue` e laço infinito](#)
  9. [Boas práticas](#)
- 

## Objetivos da Aula

- Entender o funcionamento do **laço `while`** e seu uso em repetições com condição lógica.
  - Aplicar o `while` em situações com **condição de parada controlada**.
  - Utilizar contadores, acumuladores e lógica condicional dentro do laço.
  - Introduzir conceitos de **interrupção (`break`)** e **continuação (`continue`)**.
- 

## Revisão Rápida: O que é o `while`

O **laço `while`** executa um bloco de código **enquanto** uma condição for **verdadeira**. Quando a condição se torna **falsa**, o programa sai do laço e continua a execução normal.

 Em português:

```
enquanto (condição for verdadeira):  
    faça algo
```

 Em Python:

```
while condição:  
    # bloco de código
```

---

## Estrutura Geral e Lógica do Laço

```
while condição:
    instrução_1
    instrução_2
# quando a condição for falsa, o laço termina
```

### ⚠️ Atenção:

Se a condição **nunca mudar para falsa**, o laço se tornará **infinito**, e o programa nunca terminará.

---



## Exemplo 1 — Contagem simples

```
c = 1
while c < 10:
    print(c)
    c += 1
print('Fim!')
```



### Explicação passo a passo:

- Enquanto `c` for menor que 10, imprime o número.
- O comando `c += 1` soma 1 ao contador.
- Quando `c` chegar a 10, a condição `c < 10` fica falsa → o laço termina.



### Saída:

```
1
2
3
4
5
6
7
8
9
Fim!
```

---



## Exemplo 2 — Condição de parada com número

```
n = 1
while n != 0:
    n = int(input('Digite um número (0 para sair): '))
print('Fim!')
```



### Explicação:

- O laço roda **enquanto o número for diferente de 0**.
  - Assim que o usuário digita 0, o `while` termina.
-

## Exemplo 3 — Controle com resposta do usuário

```
resposta = 'S'
while resposta == 'S':
    numero = int(input('Digite um número: '))
    resposta = input('Quer continuar? [S/N]: ').strip().upper()[0]
print('Fim!')
```



### O que está acontecendo aqui:

- O laço continua **enquanto o usuário responder “S”**.
  - `.strip()` → remove espaços.
  - `.upper()` → transforma em maiúscula.
  - `[0]` → pega apenas a primeira letra digitada.
- 



## Exemplo 4 — Contagem de pares e ímpares

```
numero = 1
pares = 0
impares = 0

while numero != 0:
    numero = int(input('Digite um número (0 para sair): '))
    if numero != 0: # evita contar o 0
        if numero % 2 == 0:
            pares += 1
        else:
            impares += 1

print(f'Você digitou {pares} números pares e {impares} números ímpares.')
```



### Explicação:

- O laço roda até o número ser igual a 0.
  - O operador % (módulo) verifica o **resto da divisão** — se for 0, o número é par.
- 



## Conceitos Importantes: break, continue e laço infinito



### break

Interrompe o laço imediatamente, mesmo que a condição ainda seja verdadeira.

```
while True:
    nome = input('Digite seu nome (ou "sair" para encerrar): ')
    if nome.lower() == 'sair':
        break
    print(f'Olá, {nome}!')
```

## **continue**

Pula o restante do bloco e volta ao início do laço.

```
contador = 0
while contador < 10:
    contador += 1
    if contador == 5:
        continue # pula o 5
    print(contador)
```






## **Evitando laço infinito**

Certifique-se de que **a condição muda dentro do laço**, senão o programa nunca terminará.

---



## **Boas práticas**

Dica	Descrição
 Controle bem a condição	Sempre garanta que algo dentro do laço possa torná-la falsa.
 Use contadores claros	Nomes de variáveis como <code>contador</code> , <code>total</code> , <code>soma</code> ajudam a legibilidade.
 Use <code>break</code> com cuidado	Ele deve ser usado quando realmente necessário.
 Normalize entradas	<code>.strip()</code> , <code>.lower()</code> e <code>.upper()</code> ajudam a evitar erros.
 Siga a PEP 8	Utilize 4 espaços por indentação e nomes descritivos.

---



## **Exercícios Práticos**

- Contagem Regressiva:**  
Faça um programa que conte de 10 até 0 e exiba “Fogo!” no final.
  - Soma até zero:**  
Peça números até o usuário digitar 0 e mostre a soma total.
  - Maior e menor número:**  
Solicite números até o usuário decidir parar. Mostre qual foi o maior e o menor digitado.
  - Média de notas:**  
Leia várias notas até o usuário digitar “sair”. Mostre a média final.
- 



## **Desafio de Projeto**

### **Jogo: Adivinhe o número secreto**

```
import random

numero_secreto = random.randint(1, 10)
palpite = 0
```

```
tentativas = 0

while palpite != numero_secreto:
    palpite = int(input('Adivinhe o número (1 a 10): '))
    tentativas += 1

    if palpite < numero_secreto:
        print('Mais... tente um número maior!')
    elif palpite > numero_secreto:
        print('Menos... tente um número menor!')

print(f'Parabéns! Você acertou em {tentativas} tentativas.')
```

---



## Resumo da Aula

Conceito	Descrição
<code>while</code>	Executa um bloco enquanto a condição for verdadeira
<code>break</code>	Interrompe o laço
<code>continue</code>	Pula para a próxima iteração
Condição de parada	Define quando o laço termina
Aplicação prática	Controle de fluxo, jogos, loops de entrada