

Departamento de Engenharia de Sistemas e Computação

Iteração

Laços while e for, laços contados e condições complexas

João Araujo

Algoritmos Computacionais I

Aula 08

Roteiro

- Por que repetir? Iteração (Cap. 3.1.3 e 3.5)
- O comando `while`
- Laços contados com `for` e `range` (3.6)
- Padrões: acumulador, contador, sentinela
- Laços encaixados
- Condições complexas e desafios de lógica

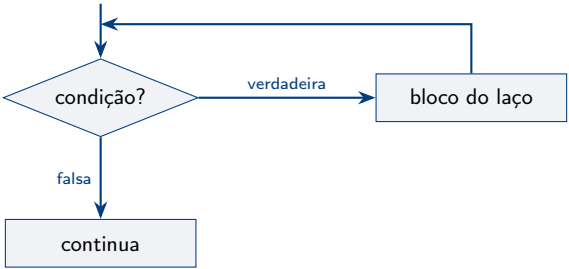
Iteração: repetir um bloco

- Computadores são ótimos em **repetir** tarefas sem se cansar
- Somar $1 + 2 + \cdots + n$, processar mil medições, tabular uma função
- Escrever 1000 `prints` é inviável; um laço resolve em 3 linhas

Dois tipos de laço

- `while`: repete *enquanto* uma condição for verdadeira
- `for`: repete para cada elemento de uma sequência

Fluxograma do `while`



Cuidado com o laço infinito

Algo dentro do bloco precisa *modificar* a condição; caso contrário o programa nunca termina (`Ctrl+C` interrompe).

while: soma de 1 até n

```
n = int(input('Qual o número final? '))
soma = 0
i = 1
while i <= n:
    soma = soma + i
    i = i + 1
print('A soma de 1 até ', n, ' = ', soma)
```

- soma: acumulador (começa em 0)
- i: contador (começa em 1 e cresce até ultrapassar n)
- Teste na mão ($n = 4$): $i = 1$, $soma = 1$; $i = 2$, $soma = 3$; $i = 3$, $soma = 6$; $i = 4$, $soma = 10$

Fonte: code/Cap3/prog.3.11_soma_numeros.py (Exercício 3.11)

Validando a entrada

```
n = int(input('Qual o número final? '))
if n > 0:
    soma = 0
    i = 1
    while i <= n:
        soma = soma + i
        i = i + 1
    print('A soma de 1 até ', n, ' = ', soma)
else:
    print('Use números positivos')
```

- Combina `if` (validação) com `while` (cálculo)
- Aviso: `n` negativo ou zero não deve gerar resultado

Fonte: `code/Cap3/prog.3.12_soma_numeros_v2.py`

Sentinela: repetir até o usuário mandar parar

```
n = int(input('Qual o número final? (0 para terminar) '))
while n != 0:
    if n > 0:
        soma = 0
        i = 1
        while i <= n:
            soma = soma + i
            i = i + 1
        print('A soma de 1 até ', n, ' = ', soma)
    else:
        print('Use números positivos')
    n = int(input('Qual o número final? (0 para terminar) '))
```

O valor 0 é o sentinela. Note os **laços encaixados** e a releitura de n no fim.

Fonte: code/Cap3/prog.3.13_soma_numeros_v3.py

Média de números digitados (Exercício 3.12)

```
soma = 0
quantidade = 0
valor = float(input("Digite um valor (negativo para parar): "))
while valor >= 0:
    soma += valor
    quantidade += 1
    valor = float(input("Digite um valor (negativo para parar): "))
if quantidade > 0:
    print("Média:", soma / quantidade)
else:
    print("Nenhum valor digitado")
```

- `soma += valor` é o mesmo que `soma = soma + valor`
- Proteja a divisão: se `quantidade` for 0, evita `ZeroDivisionError`

Laços contados: `for` e `range`

```
for i in ["Vermelho", "Verde", "Azul"]:
    print(i)
for i in range(1, 10):
    print(i, end=' ')
```

```
Vermelho
Verde
Azul
1 2 3 4 5 6 7 8 9
```

`range(5)`	0, 1, 2, 3, 4
`range(1, 10)`	1, 2, . . . , 9 (o final não entra)
`range(0, 361, 30)`	0, 30, 60, . . . , 360
`range(10, 0, -2)`	10, 8, 6, 4, 2

Fonte: `code/Cap3/prog.3.14_cores.py`, `prog.3.15_contagem.py`

Fatorial com `for`

```
n = int(input('Qual o número para calcular o fatorial? '))
fat = 1
for i in range(1, n+1):
    fat = fat * i
print('O fatorial de', n, 'é', fat)
```

- $n! = 1 \times 2 \times \cdots \times n$; acumulador multiplicativo começa em 1, não em 0
- `range(1, n+1)`: o limite superior é excluído
- Compare com a soma por `while`: quando se sabe quantas vezes repetir, `for` é mais simples (Exercício 3.13)

Fonte: `code/Cap3/prog.3.17_fatorial.py`

Quando usar `for` ou `while`?

`for`

Número de repetições conhecido ou percorrer uma sequência.

Tabela de 0 a 360 graus; notas de uma turma.

`while`

Repetir até que algo aconteça.

Ler até valor válido; aproximar até o erro ficar pequeno.

`break` sai do laço; `continue` pula para a próxima volta:

```
for n in range(1, 100):
    if n % 7 == 0 and n % 5 == 0:
        print("Primeiro múltiplo de 35:", n)
        break
```

Laços encaixados: padrões

```
for linha in range(1, 5):
    for coluna in range(linha):
        print('*', end='')
    print()
```

```
*
**
***
****
```

- O laço interno *completo* executa a cada volta do externo
- Exercício 3.15: gerar outros padrões com laços encaixados

Condições complexas: desafios de lógica

Desafio 1

Mostre todos os anos **bissextos** entre 1900 e 2030. (Revisite a condição da Aula 7.)

Desafio 2

Mostre os números de 1 a 100 que são divisíveis por 3 **ou** 5, mas **não** por ambos.

Desafio 3

Leia números até digitar 0 e informe o maior e o menor.

Estratégia: escreva primeiro a *condição* em português, depois traduza para `and`/`or`/`not`, e teste os extremos.

Desafio 1: bissextos em um intervalo

```python
for ano in range(1900, 2031):
    if (ano % 4 == 0 and ano % 100 != 0) or ano % 400 == 0:
        print(ano, end=' ')
```

```
1904 1908 1912 ... 1996 2000 2004 ... 2028
```

- 1900 não aparece (divisível por 100, não por 400); 2000 aparece
- Dica: conte quantos anos foram impressos usando um contador

Para praticar: exercícios do livro

Site do livro – Capítulo 3

- `while`: **3.11, 3.12**
- `for`: **3.13** (refazer 3.11), **3.14** (tabela Celsius → Fahrenheit), **3.15** (padrões com laços encaixados)

Enunciados e soluções comentadas em
`https://www.araujo.eng.uerj.br/livro/exercicios/cap3/`

Exercícios adicionais (sem solução): `https://www.araujo.eng.uerj.br/livro/extras/`

Resumo da aula

- `while` repete enquanto a condição for verdadeira: cuidado com laços infinitos
- `for` com `range` é ideal para laços contados
- Padrões: contador, acumulador, sentinela
- Laços podem ser encaixados; `break` e `continue` controlam o fluxo
- Próxima aula (prática): laboratório de problemas reais