

# Iteração

**Programação I**  
**2017.2018**

*Teresa Gonçalves*  
[tcg@uevora.pt](mailto:tcg@uevora.pt)

Departamento de Informática, ECT-UÉ

# Sumário

**Revisão**

**Iteração**

**Terminação antecipada**

# Revisão

# Execução condicional

## Instrução if

```
if <condição>:  
    <instruções quando a condição é verdadeira>
```

## Instrução if-else

```
if <condição>:  
    <instruções quando a condição é verdadeira>  
else:  
    <instruções quando a condição é falsa>
```

# Execução condicional

## Instrução if-elif

```
if <condição1>:
```

```
    <instruções quando condição1 é verdadeira>
```

```
elif <condição2>:
```

```
    <instruções quando condição1 é falsa e condição2  
é verdadeira>
```

```
...
```

# Características

**É obrigatório existir instruções no corpo do if**

**Pode não existir um else**

**Podem existir inúmeros elif**

**Apenas é executado um dos ramos**

**Apenas as instruções referentes à 1ª condição verdadeira são executadas**

# Condicionais encadeados vs. encaixados

## Encadeado

Utilização da instrução `elif`

## Encaixado

Instrução `if` como corpo do `else`

## Condicionais encaixados

Podem tornar a leitura/compreensão mais difícil

Podem ser utilizados indevidamente

**Iteração**



# Problema - *countdown*

**Mostrar uma contagem decrescente: de 5 a 0**

## **Solução 1**

```
print(5)
```

```
print(4)
```

```
print(3)
```

```
print(2)
```

```
print(1)
```

```
print(0)
```

# Problema - *countdown*

**E se a contagem começar em 20?**

**E se a contagem começar em N?**

**O problema é idêntico!**

Varia o **número de vezes** que a função print é invocada e o **valor** que é mostrado

**Iteração**

Capacidade de executar um bloco de instruções repetidamente!

# Instrução while

```
while <condicao>:  
    <bloco de instruções>
```

## Fluxo de execução

1. Avaliar a condição, obtendo True ou False
2. Se False, sai da instrução while e continua com próxima instrução
3. Se True, executa o corpo do while e volta ao passo 1

# Problema *countdown* (2)

```
n=5
while n>=0:
    print(n)
    n = n-1
```

# Condição while

## Para o ciclo terminar...

... o valor das variáveis da condição devem ser alteradas no corpo do while!

## Nem sempre é fácil verificar a convergência

```
while n>1:
    print(n)
    if n%2 == 0:
        n = n//2
    else:
        n = n*3+1
```

# Exemplos

**Calcular a média de 5 números**

**Calcular a média até à introdução de um valor negativo**

**Calcular a raiz quadrada de um número**

# Calcular a média de 5 números

```
soma=0
n=1
while n<=5:
    num=int(input('Introduza o num ' +str(n)+' : '))
    soma=soma+num
    n=n+1
media=soma/5
print('A media dos numeros introduzidos é ', media)
```

# Calcular a média até à introdução de um valor negativo

```
soma=0
n=0
num=int(input('Introduza um num (<0 para terminar): '))
while num>=0:
    soma=soma+num
    n=n+1
    num=int(input('Introduza um num (<0 para terminar): '))
media=soma/n
print('A media dos numeros introduzidos é ', media)
```



# Raíz quadrada: método de Newton

**Começa com uma estimativa da raiz quadrada,  $x$**

**Calcula uma nova estimativa,  $y$**

**Termina quando a diferença entre 2 estimativas consecutivas é desprezível**

**Estimativa**

$$y = \frac{x + a/x}{2}$$

# Calcular a raiz quadrada de um número

## Variáveis

Número: a

Estimativa inicial: x

Valor de paragem: epsilon

$$y = \frac{x + a/x}{2}$$

```
while True:
```

```
    y = (x+a/x)/2
```

```
    print(y)
```

```
    if abs(y-x)<epsilon:
```

```
        break
```

```
    x = y
```

```
print( 'A raiz quadrada de',  
a, 'é', x )
```

# Instrução break

## Termina o ciclo antecipadamente

### Exemplo

```
while True:
    line = input('> ')
    if line == 'done':
        break
    print(line)
print('Done!')
```

# Instrução continue

## Passa de imediato para nova iteração (fazendo o teste)

Não executa as instruções restantes do bloco

### Exemplo

```
while True:
    line = input('> ')
    if line == 'jump':
        continue
    if line == 'done':
        break
    print(line)
print('Done!')
```

# Atenção à saída do ciclo!

## Verificar a condição de paragem

É comum existir uma iteração a mais ou a menos

## Verificar efeitos da interrupção antecipada

As instruções `break` e `continue` podem produzir efeitos não desejados no comportamento do ciclo

# Exemplo - qual o valor de n no final?

```
n = 0
while True:
    line = input('> ')
    if line == 'jump':
        continue
    if line == 'done':
        break
    print(line)
    n = n+1
print('Done!')
```

# Exemplo - qual o valor de n no final?

```
n = 0
while True:
    line = input('> ')
    n = n+1
    if line == 'jump':
        continue
    if line == 'done':
        break
    print(line)
print('Done!')
```