

Tópicos avançados em Informática I

Programação Python



Estruturas de repetição

Profa. Ana Paula Canal
Universidade Franciscana

Sumário

- ▶ Introdução
- ▶ Laços de repetição
 - ▶ while
 - ▶ For
- ▶ Break e continue
- ▶ Laços aninhados

Laços de repetição

- ▶ *Loop* ou *Looping*
- ▶ Executar um bloco de comandos "**n**" vezes
- ▶ Critério de parada para o *loop* : é uma condição
- ▶ O fluxo de execução do programa é desviado para a linha seguinte ao bloco que define o laço de repetição, quando a condição for falsa
- ▶ Sequências de comandos que estão se repetindo são adequadas para o uso de estrutura de repetição
- ▶ Evita a duplicação de código
- ▶ No Python são duas alternativas oferecidas para a construção de laços:
 - ▶ **for**
 - ▶ **while**

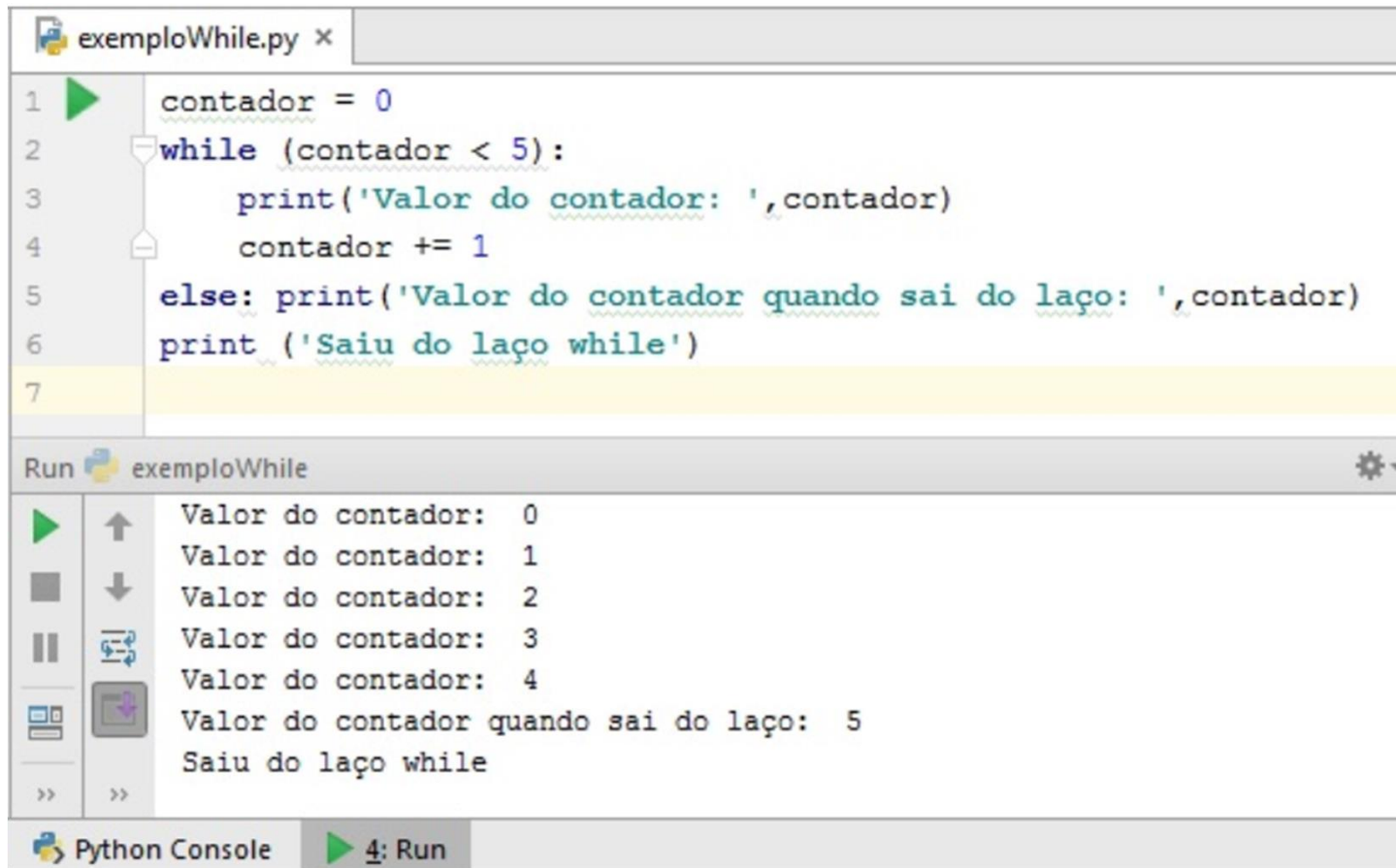
while

- ▶ Testa uma condição e enquanto a condição for verdadeira o bloco de comandos será executado
- ▶ Quando o teste resultar em um valor falso, o programa para de executar as instruções que fazem parte do bloco de repetição e desloca o fluxo para a linha seguinte ao while
- ▶ Se a condição não for bem definida o programa pode entrar em um *looping* infinito
- ▶ Este tipo de repetição é apropriada para as situações em que não se consegue determinar diretamente o número de repetições

while

```
while <condição>:  
    instrução 1  
    instrução 2  
    ...  
    instrução n  
else:  
    instrução m
```

while



The screenshot shows a Python IDE with a file named `exemploWhile.py`. The code defines a `while` loop that prints the value of a counter from 0 to 4, then prints the value when it exits the loop (5), and finally prints a confirmation message. The output window shows the execution results, and the Python Console at the bottom indicates the code was run successfully.

```
1 contador = 0
2 while (contador < 5):
3     print('Valor do contador: ', contador)
4     contador += 1
5 else: print('Valor do contador quando sai do laço: ', contador)
6 print ('Saiu do laço while')
7
```

Run exemploWhile

Valor do contador: 0
Valor do contador: 1
Valor do contador: 2
Valor do contador: 3
Valor do contador: 4
Valor do contador quando sai do laço: 5
Saiu do laço while

Python Console 4: Run

Inicialização, incremento e decremento

- ▶ Corresponde ao valor inicial atribuído a uma variável que contará o número de iterações do laço
- ▶ Na linguagem Python as variáveis são inicializadas no momento em que são definidas
- ▶ Incremento (*increment*)
 - ▶ Atualizar o conteúdo de uma variável com uma adição
- ▶ Decremento (*decrement*)
 - ▶ alterar o valor da variável com uma subtração

while

The screenshot shows a Python IDE with a file named `exemploWhile.py`. The code is as follows:

```
1 contador = 0
2 while (contador < 5):
3     print('Valor do contador: ', contador)
4     contador += 1
5 else: print('Valor do contador quando sai do laço: ', contador)
6 print ('Saiu do laço while')
7
```

Annotations with arrows point to specific parts of the code:

- `contador = 0` is labeled "inicialização" (initialization).
- `while (contador < 5):` is labeled "condição" (condition).
- `contador += 1` is labeled "incremento" (increment).

Below the code editor, the output of the program is displayed in the "Run" window:

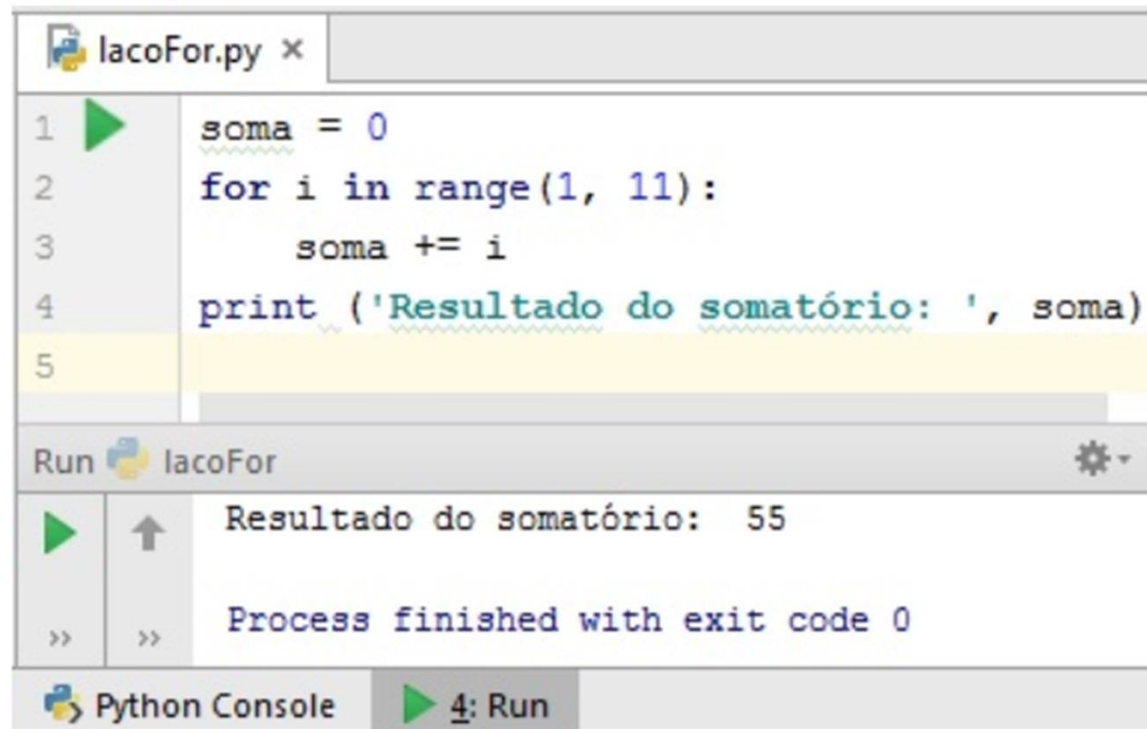
```
Valor do contador: 0
Valor do contador: 1
Valor do contador: 2
Valor do contador: 3
Valor do contador: 4
Valor do contador quando sai do laço: 5
Saiu do laço while
```

The status bar at the bottom indicates "Python Console" and "4: Run".

for

- ▶ É indicada para ser usada quando é conhecida a quantidade de repetições do laço
- ▶ Executa uma iteração sobre os elementos, itens que estão inseridos no bloco de repetição
- ▶ Sempre que ocorrer uma iteração a referência também será atualizada
- ▶ Posteriormente ocorrerá o processamento do bloco de comandos com o elemento referenciado

for



The screenshot shows a Python IDE window titled 'lacoFor.py'. The code defines a variable 'soma' as 0, then uses a 'for' loop with 'range(1, 11)' to iterate from 1 to 10, adding each value to 'soma'. Finally, it prints the result. Below the code editor, the 'Run' button is pressed, and the output console shows the result 'Resultado do somatório: 55' and the message 'Process finished with exit code 0'.

```
1 soma = 0
2 for i in range(1, 11):
3     soma += i
4 print ('Resultado do somatório: ', soma)
5
```

Run lacoFor

Resultado do somatório: 55

Process finished with exit code 0

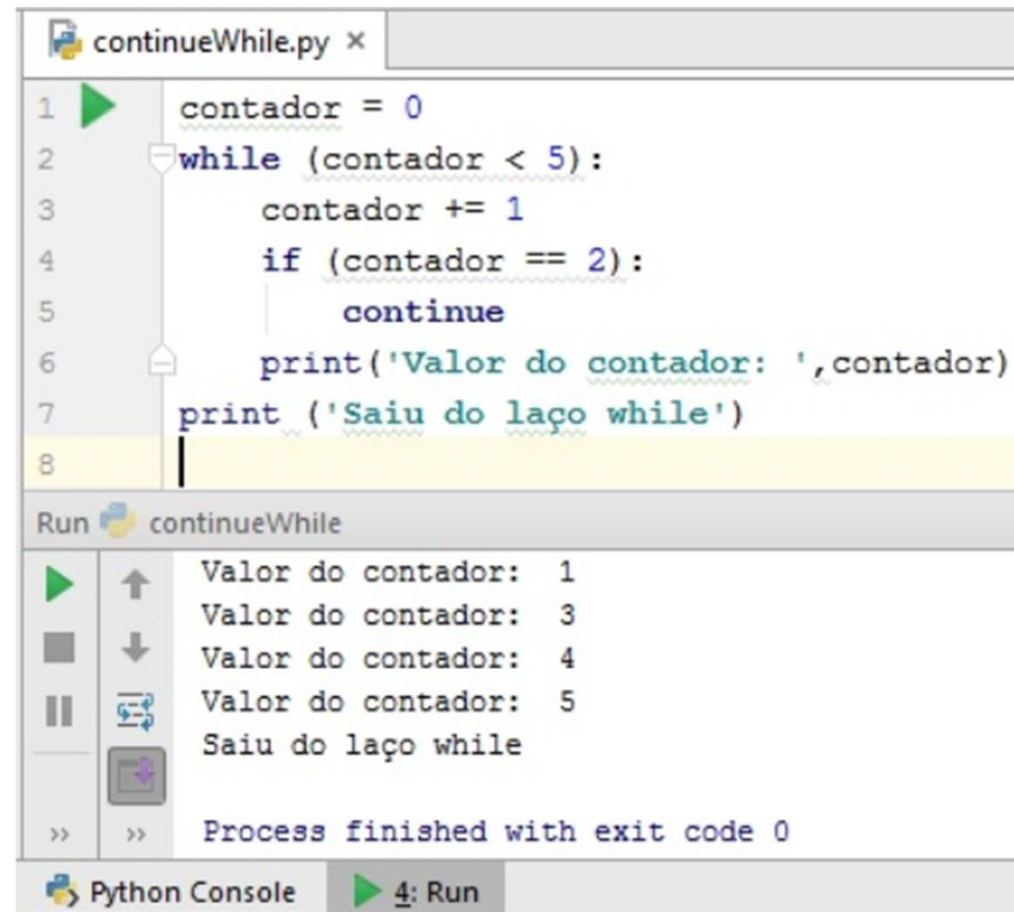
Python Console 4: Run

Break e continue

- ▶ São recursos computacionais utilizados para alterar o fluxo de execução de um laço de repetição
- ▶ Desconsiderar interações
- ▶ Interromper a execução do e deslocar o fluxo para a linha seguinte ao bloco

Continue com o while

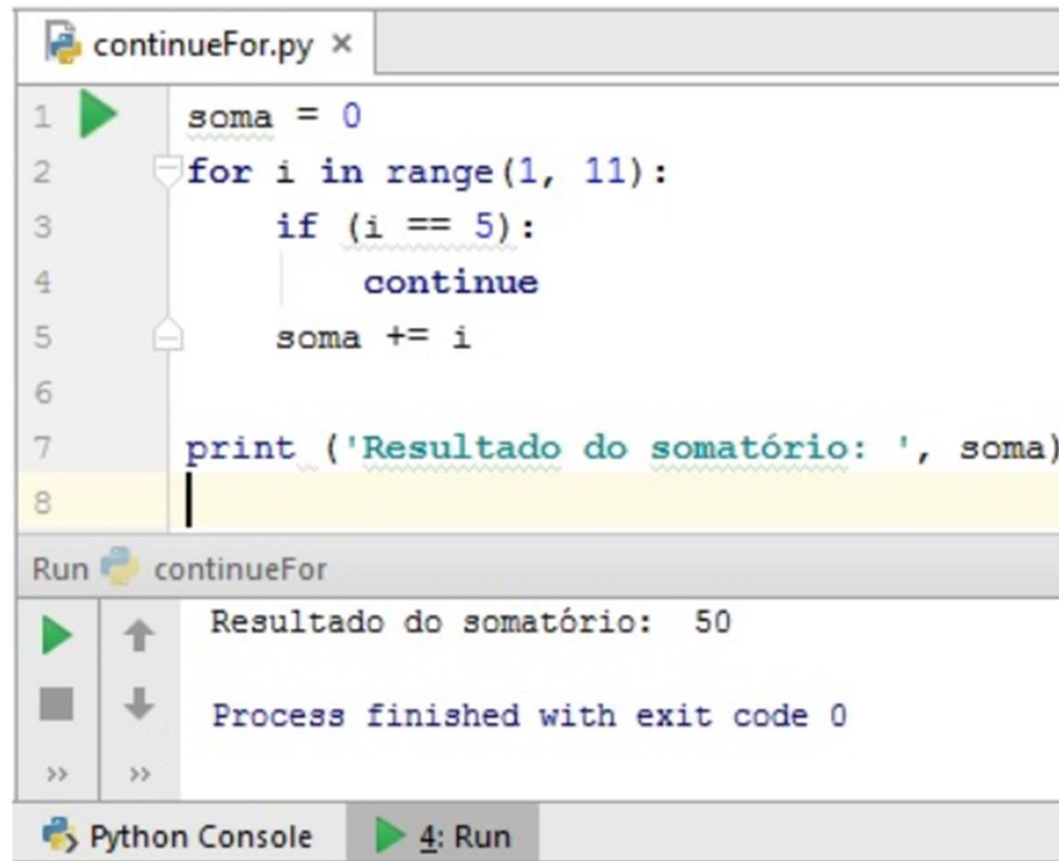
- ▶ Encerra a iteração corrente, porém continua a iterar a repetição do laço
- ▶ O fluxo de execução do programa é deslocado para a linha que define o cabeçalho do laço



```
continueWhile.py x
1  contador = 0
2  while (contador < 5):
3      contador += 1
4      if (contador == 2):
5          continue
6      print('Valor do contador: ', contador)
7  print('Saiu do laço while')
8

Run continueWhile
Valor do contador: 1
Valor do contador: 3
Valor do contador: 4
Valor do contador: 5
Saiu do laço while
Process finished with exit code 0
Python Console 4: Run
```

Continue com for



The screenshot shows a Python IDE window titled 'continueFor.py'. The code is as follows:

```
1 soma = 0
2 for i in range(1, 11):
3     if (i == 5):
4         continue
5     soma += i
6
7 print ('Resultado do somatório: ', soma)
8
```

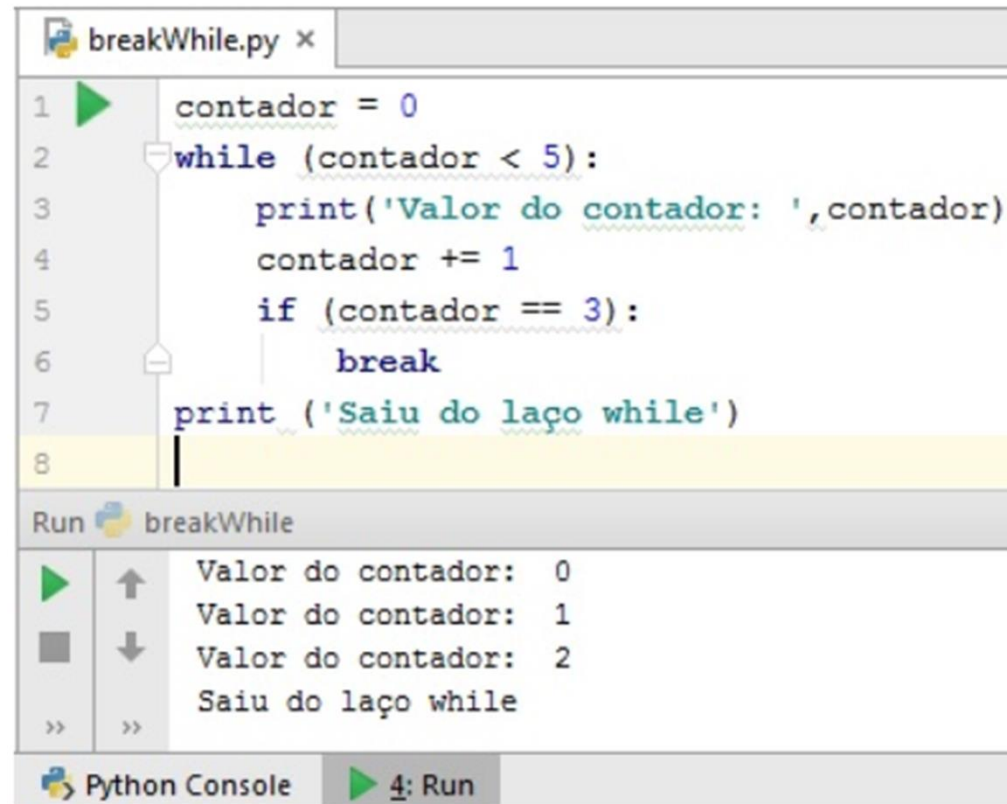
Below the code editor, the 'Run' button is clicked, and the output is displayed in the console:

```
Run continueFor
Resultado do somatório: 50
Process finished with exit code 0
```

The bottom status bar indicates 'Python Console' and '4: Run'.

Break com while

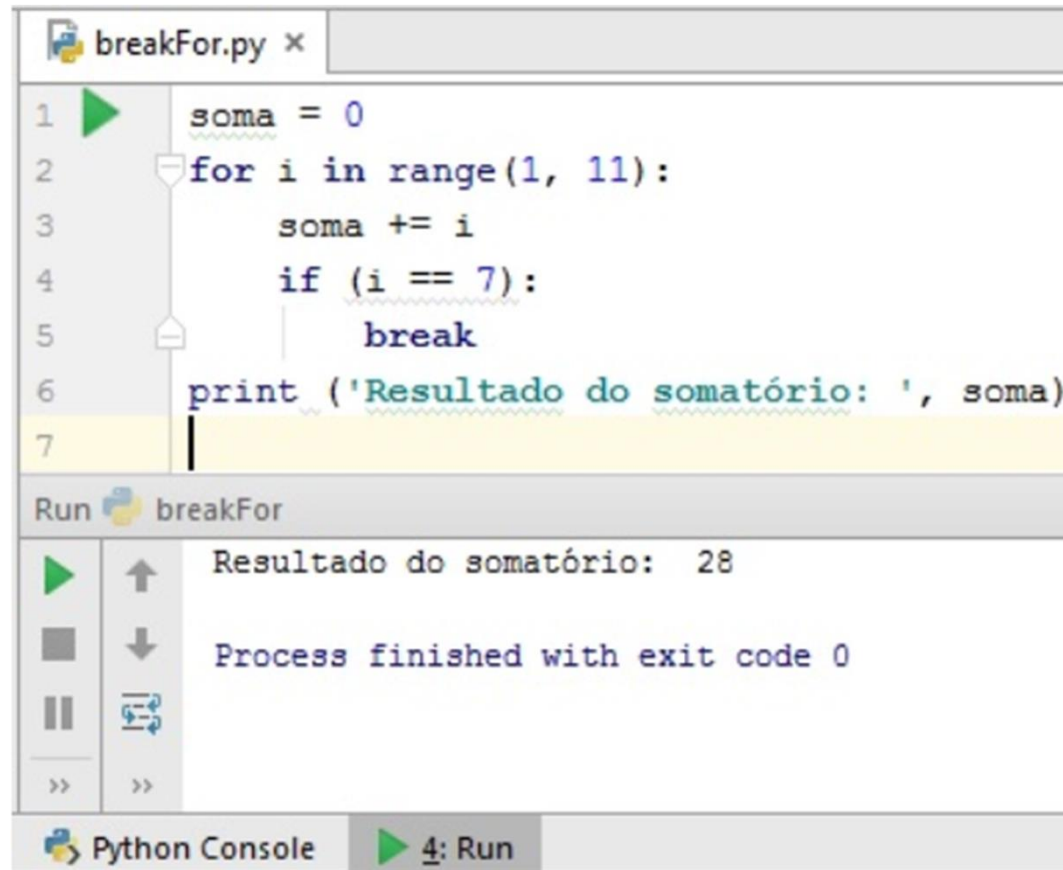
- ▶ Interrompe completamente a execução do laço
- ▶ O fluxo de execução passa para a linha seguinte à definição do bloco de comandos do laço



```
breakWhile.py x
1  contador = 0
2  while (contador < 5):
3      print('Valor do contador: ', contador)
4      contador += 1
5      if (contador == 3):
6          break
7  print ('Saiu do laço while')
8

Run breakWhile
Valor do contador: 0
Valor do contador: 1
Valor do contador: 2
Saiu do laço while
Python Console 4: Run
```

Break com for



```
breakFor.py x
1 soma = 0
2 for i in range(1, 11):
3     soma += i
4     if (i == 7):
5         break
6 print ('Resultado do somatório: ', soma)
7

Run breakFor
Resultado do somatório: 28
Process finished with exit code 0

Python Console 4: Run
```

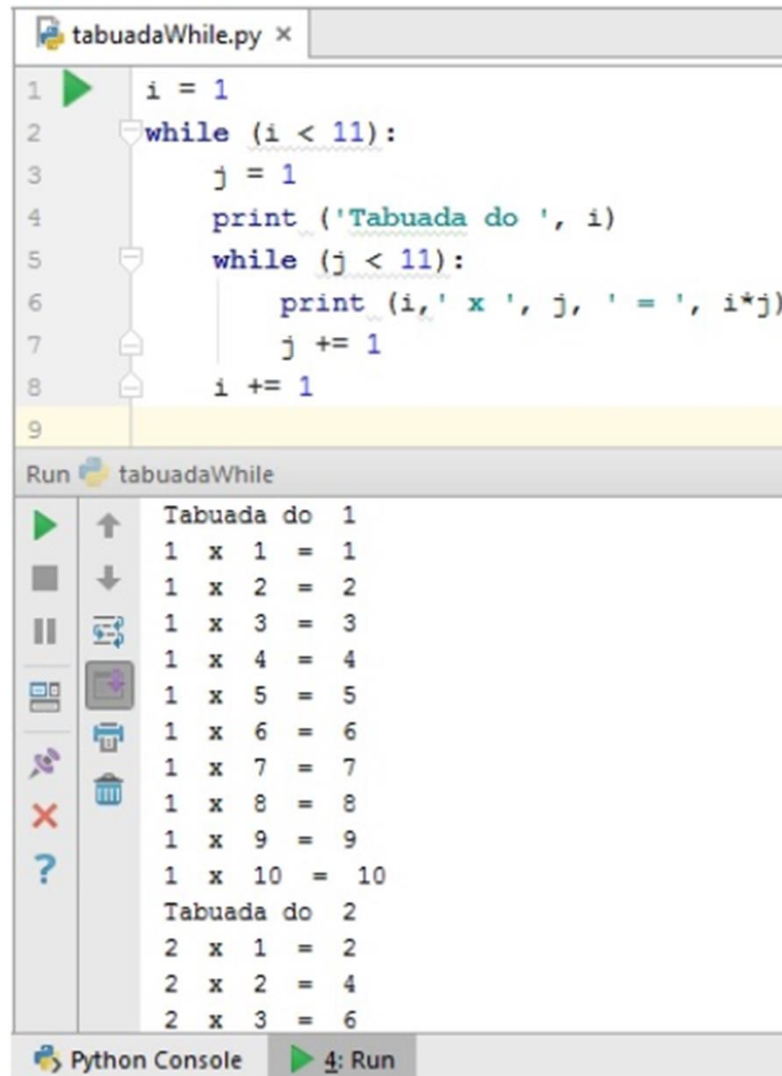
Laços aninhados

- ▶ Laços de repetição encadeados
 - ▶ Um laço dentro de outro
- ▶ Pode-se aninhar tantos laços quantos forem necessários

```
while condição 1:  
    instrução 1.1  
    instrução 1.2  
  
    ...  
    instrução 1.n  
    while condição 2:  
        instrução 2.1  
        instrução 2.2  
  
        ...  
        instrução 2.n
```

```
for <referência 1> in <sequencia>:  
    instrução 1.1  
    instrução 1.2  
  
    ...  
    instrução 1.n  
    for <referência 2> in <sequencia>:  
        instrução 2.1  
        instrução 2.2  
  
        ...  
        instrução 2.n
```


Exemplo: while



The screenshot shows a Python IDE with a file named `tabuadaWhile.py`. The code defines a `while` loop that iterates over `i` from 1 to 10. For each `i`, it prints a header `Tabuada do {i}` and then enters an inner `while` loop that iterates over `j` from 1 to 10, printing the multiplication `{i} x {j} = {i*j}`. The output window shows the execution results, displaying the multiplication tables for `i=1` and `i=2`.

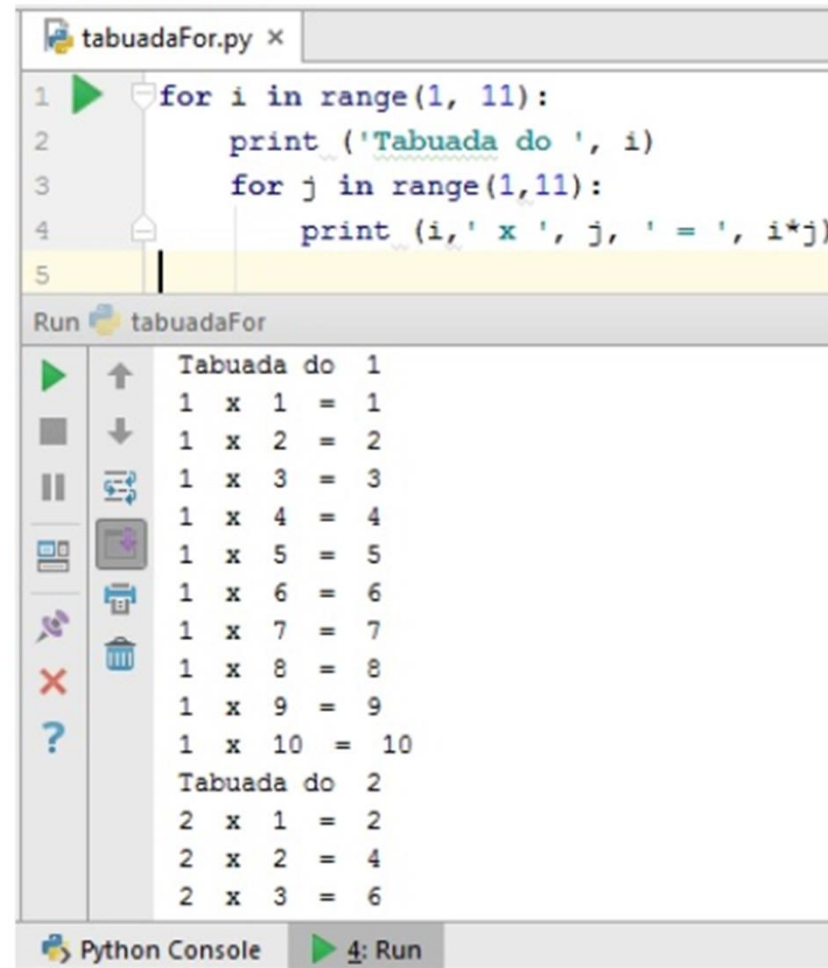
```
1 i = 1
2 while (i < 11):
3     j = 1
4     print ('Tabuada do ', i)
5     while (j < 11):
6         print (i, ' x ', j, ' = ', i*j)
7         j += 1
8     i += 1
9
```

Run tabuadaWhile

```
Tabuada do 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
Tabuada do 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
```

Python Console 4: Run

Exemplo: for



The screenshot shows a Python IDE with a file named `tabuadaFor.py`. The code defines a nested loop that prints multiplication tables for `i` from 1 to 10, and for each `i`, `j` from 1 to 10. The output in the console shows the first two tables for `i=1` and `i=2`.

```
1 for i in range(1, 11):
2     print ('Tabuada do ', i)
3     for j in range(1,11):
4         print (i, ' x ', j, ' = ', i*j)
5
```

Run tabuadaFor

```
Tabuada do 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
Tabuada do 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
```

Python Console 4: Run