Itaú Unibanco

Itaú

Programa de formação

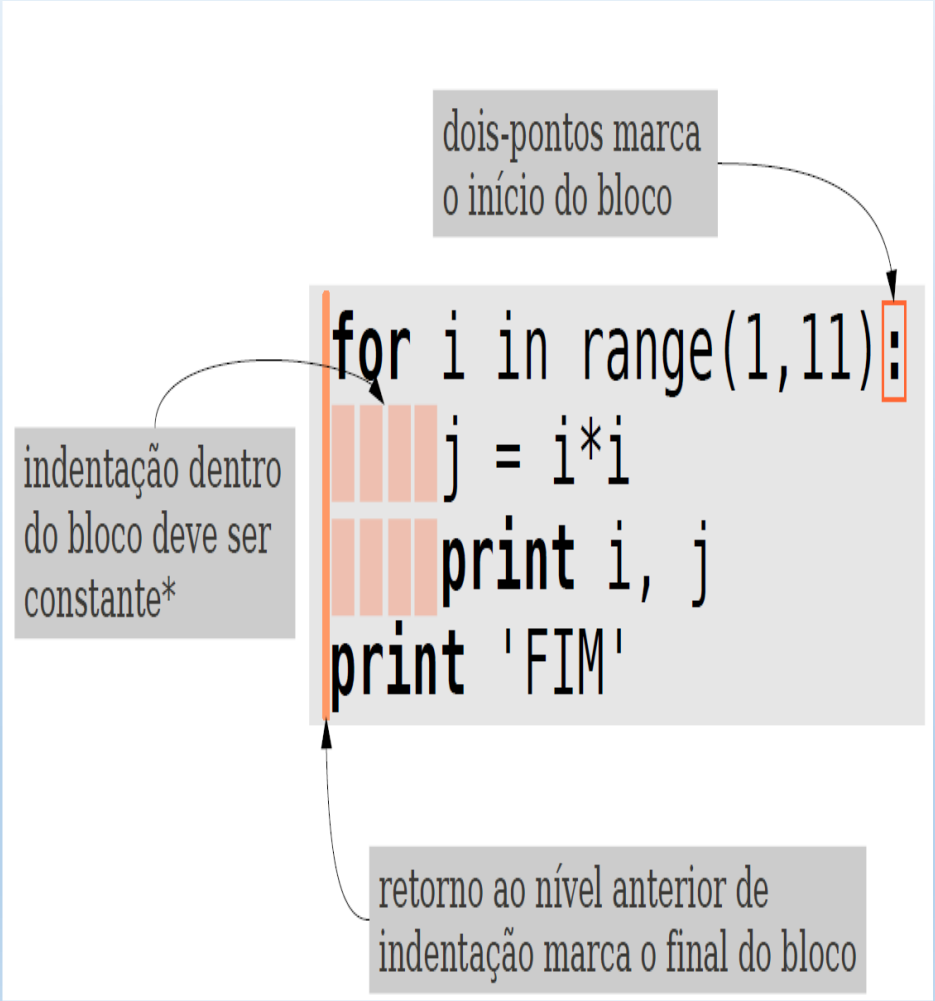
ITAú analytics

Módulo I – Fundamentos Computacionais
Sessão 2 - Aula 3 – Fluxo de Controle

Prof. Dr. Luiz Alberto Vieira Dias
Prof. Dr. Lineu Mialaret

Introdução

- No Python há duas estruturas de controle fundamentais: instruções condicionais e laços
- Comum a estas estruturas é a sintaxe usada para a definição dos blocos de código
  - O caractere “:” é usado para delimitar o começo de um bloco de código;
  - O bloco é indentado na linha seguinte ao caractere “:”
  - A linguagem conta com níveis de indentação para designar a extensão do bloco de código (ou de qualquer bloco de código interno)



Instruções Condicionais

- Construções ou desvios condicionais (instruções if) fornecem um modo de executar um trecho de código com base na avaliação de uma ou mais expressões lógicas (booleanas).
- No Python a forma geral de um desvio condicional é a seguinte:

- Cada condição é uma expressão lógica
- Cada bloco de código contém uma ou mais instruções
- Se a primeira condição procede, então o bloco de código a seguir será executado
- Se ela não procede, então a segunda condição é avaliada e assim por diante
- Podem existir qualquer número de cláusulas elif
- A clausula final else é opcional

```
if first_condition:
    first_body
elif second_condition:
    second_body
elif third_condition:
    third_body
else:
    fourth_body
```

Instruções Condicionais (cont.)

- Tipos de dados não booleanos podem ser usados para avaliações lógicas
- Exemplo:
  - Se a variável response é utilizada para representar uma string que foi fornecida pelo usuário e se deseja avaliar se o usuário forneceu ou não uma string não vazia, pode usar a expressão abaixo para realizar o teste

    if response # is False if response == ‘’
  - Que equivalente a expressão

    if response != ‘ ‘:

Instruções Condicionais (cont.)

- Exemplo: Instrução if simples

```
if Test Expression:
        Body of if
```

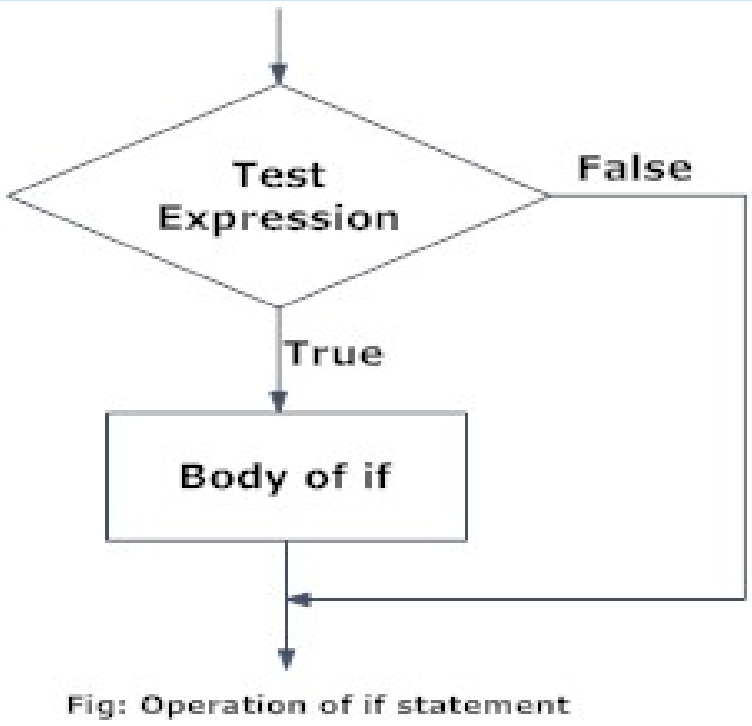


Fig: Operation of if statement

```
# If the number is positive, we print an appropriate
message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

Instruções Condicionais (cont.)

- Exemplo: Instrução if...else

```
if Test Expression:
    Body of if
else:
    Body of else
```

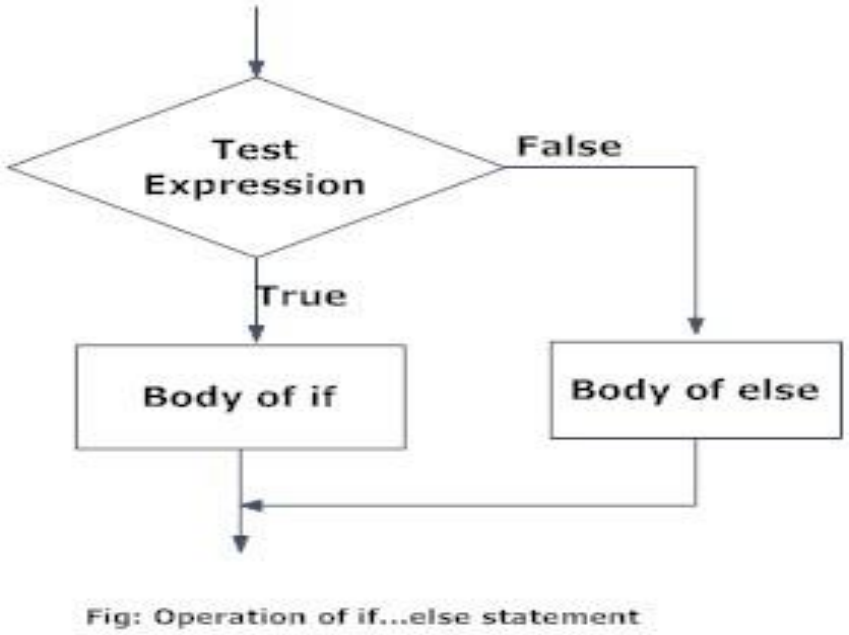


Fig: Operation of if...else statement

```
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Instruções Condicionais (cont.)

- Exemplo: Instrução if...elif...else

```
if Test Expression of if:
   Body of if
elif Test Expression of elif:
    Body of elif
else:
    Body of else
```

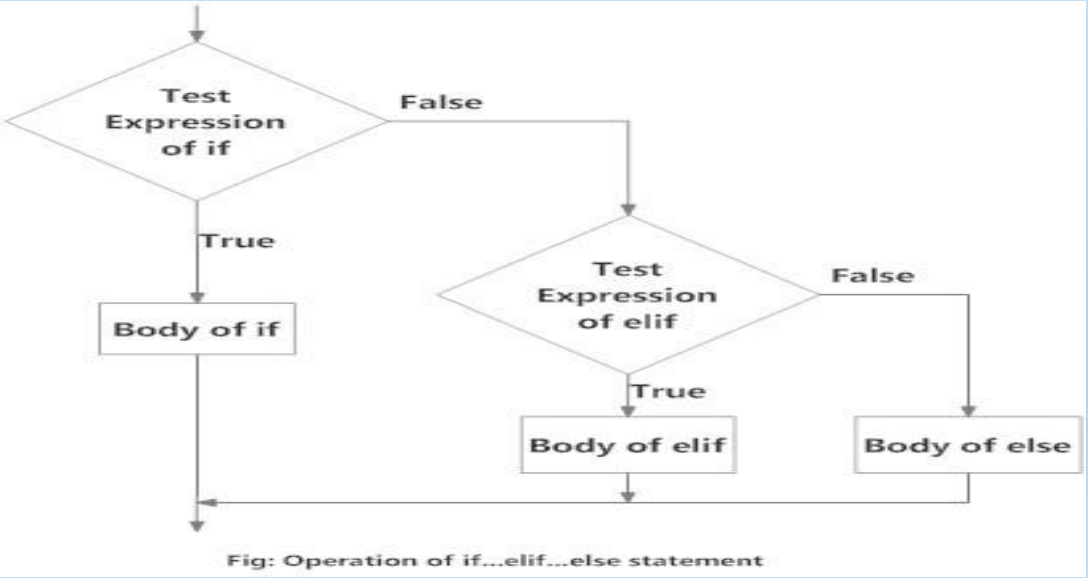


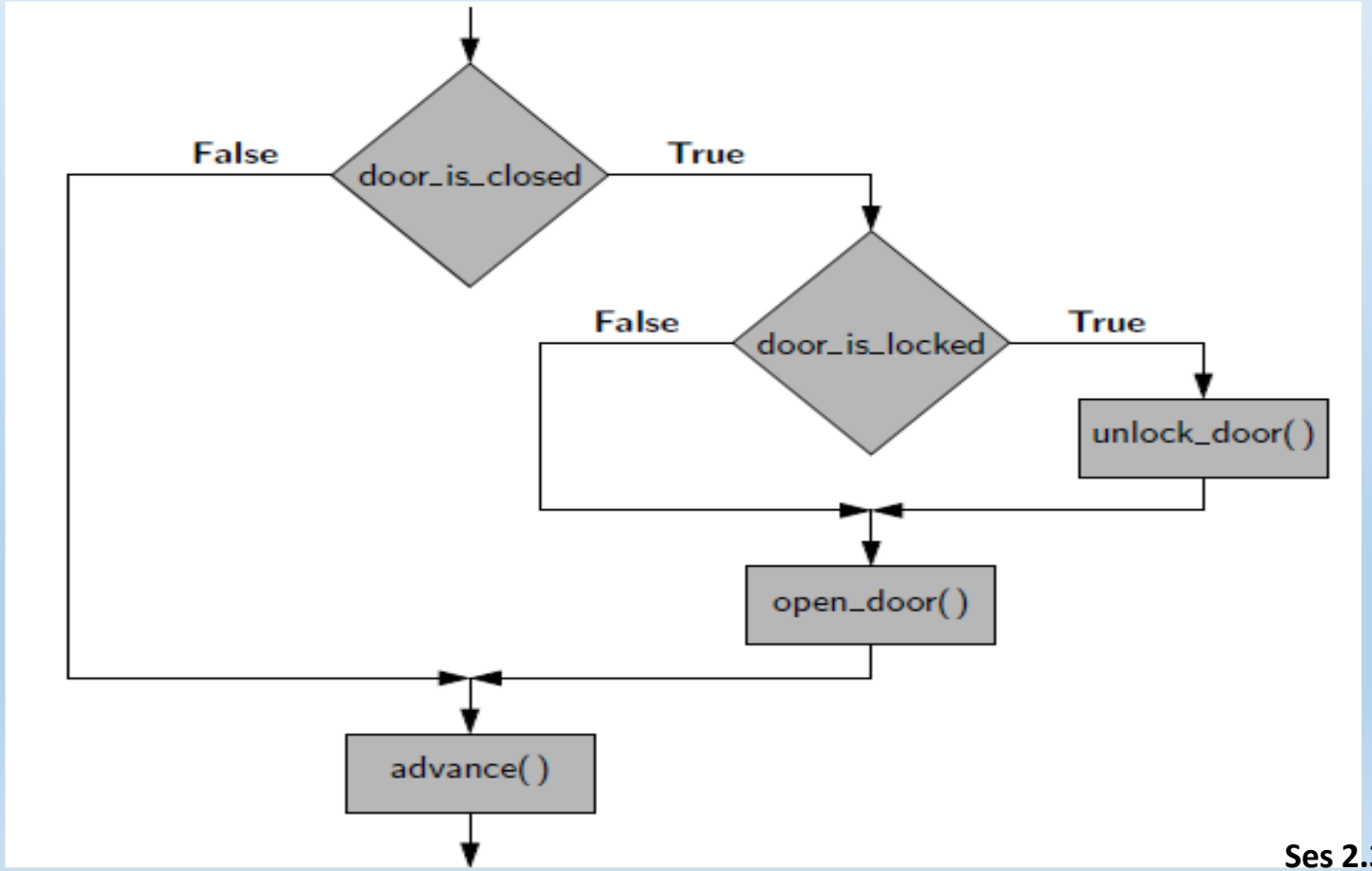
Fig: Operation of if...elif...else statement

```
# In this program,
# we check if the number is positive or
# negative or zero and
# display an appropriate message

num = 3.4

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

Instruções Condicionais (cont.)

- Pode-se embutir uma desvio condicional dentro de outro, contando-se com a indentação para tornar clara a extensão dos blocos de código
- Exemplo:

```
if door_is_closed:
    if door_is_locked:
        unlock_door( )
    open_door( )
advance( )
```



Laços no Python

- A linguagem Python disponibiliza duas construções distintas de laços:
  - O laço while que permite repetições de instruções baseadas num teste de condição lógica; e
  - O laço for que permite uma iteração de valores de uma série definida (tais como caracteres de uma string, elementos de uma lista, números em dado intervalo, etc.)

Laço while

- A sintaxe para um laço while no Python é a seguinte:

```
while Test Expression:
      Body of while
```

- A expressão *test_expression* pode ser qualquer expressão lógica e *Body of while* pode ser qualquer bloco de código
- A execução do laço começa com o teste lógico
- Se ele for verdadeiro, o bloco de código do laço é executado e o teste novamente repetido, caso contrário, o laço é abandonado

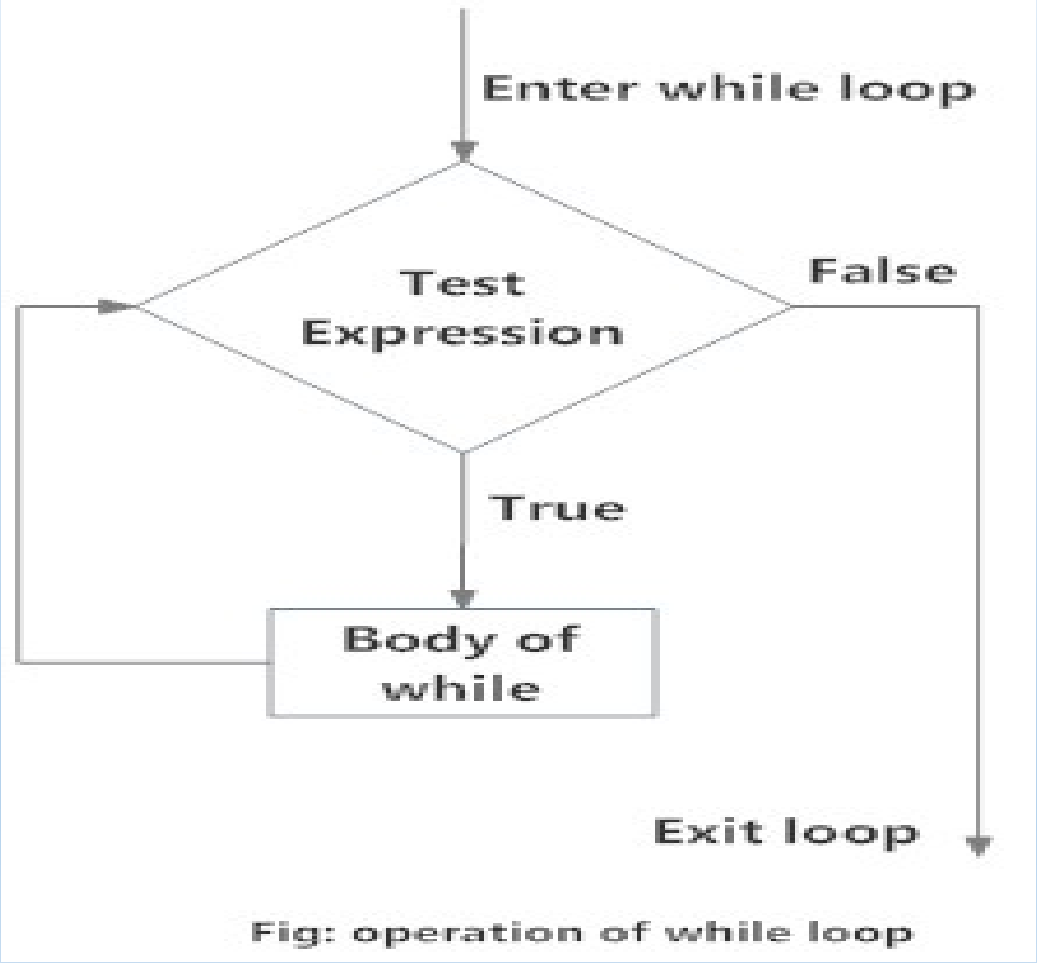


Fig: operation of while loop

Laço while (cont.)

- Exemplo:

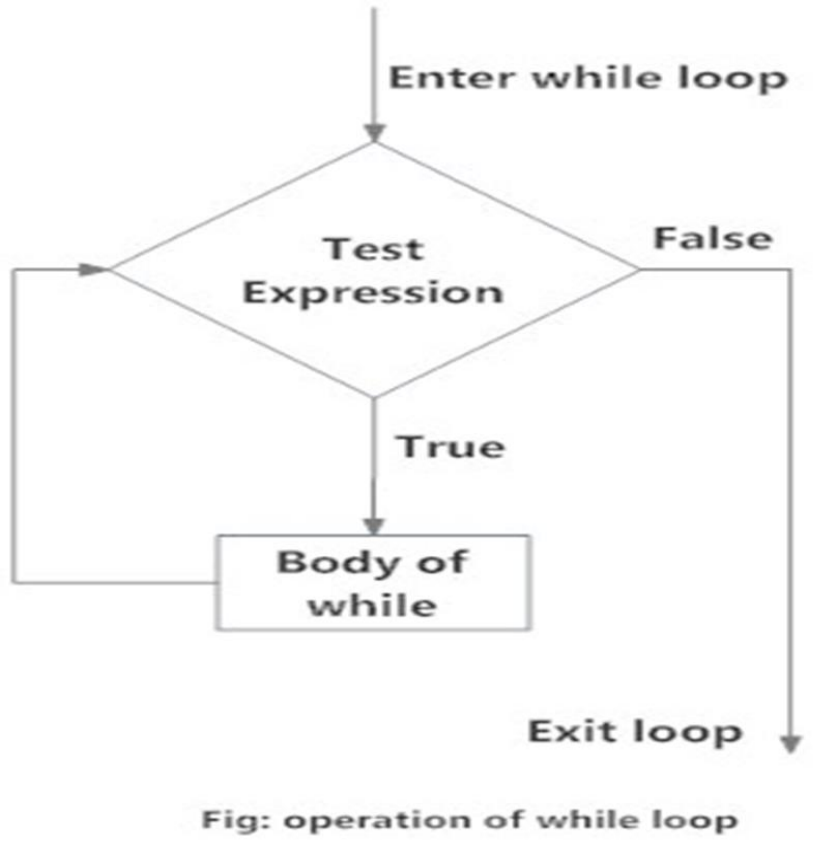


Fig: operation of while loop

```
# Program to add natural
# numbers up to
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))
n = 10

# initialize sum and counter
sum = 0
i = 1
while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

Laço while (cont.)

- Exemplo de um laço while que percorre uma sequencia de caracteres até obter uma caractere com valor = 'X' ou alcançar o final da sequencia
- Qual o problema deste laço while ao lado?

```
>>> data = "xxxxx   "
>>> j = 0
>>> while data[j] != 'X' and j < len(data):
...         j += 1
...
```

Laço while (cont.)

- Exemplo: Um laço while que percorre uma sequencia de caracteres até obter uma caractere com valor = 'X' ou alcançar o final da sequência

```
>>> data = "xxxxx   "
>>> j = 0
>>> while data[j] != 'X' and j < len(data):
...         j += 1
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

- A função len retorna o comprimento de uma sequencia como uma lista ou string
- A corretividade do laço situa-se na avaliação curto-circuito do operador and (por que?)
- Quando o laço termina, o valor de j será o índice da ocorrência mais a esquerda de 'X' ou o tamanho da string

Laço while (cont.)

- Pode se ter uma cláusula else opcional no laço while

```
while test_expression:
      Body of while
else:
      Body of else
```

- A expressão *test_expression* pode ser qualquer expressão lógica e *Body of while* pode ser qualquer bloco de código
- A execução do laço começa com o teste lógico
- Se ele for verdadeiro, o bloco de código do laço é executado e o teste novamente repetido, caso contrário, a cláusula else é executada com o bloco *Body of else*

```
# Example to illustrate
# the use of else statement
# with the while loop

counter = 0

while counter < 3:
    print("Inside loop")
    counter = counter + 1
else:
    print("Inside else")
```

Laço for

- A sintaxe do laço for no Python é uma alternativa mais conveniente que um laço while quando se está iterando uma sequência de elementos.
  - Iterar sobre uma sequência chama-se caminhamento (*traversal*)
- O laço for pode ser usado em qualquer tipo de estrutura iterável (list, tuple, str, set, dict, etc.).

Laço for (cont.)

- A sintaxe do laço for no Python é a seguinte

```
for val in sequence:
        Body of for
```

- O identificador `val` é a variável que toma o valor do item dentro da sequencia em cada iteração do laço
- O laço continua até alcançar o último elemento da sequencia
- O corpo do laço é separado do resto do código utilizando indentação

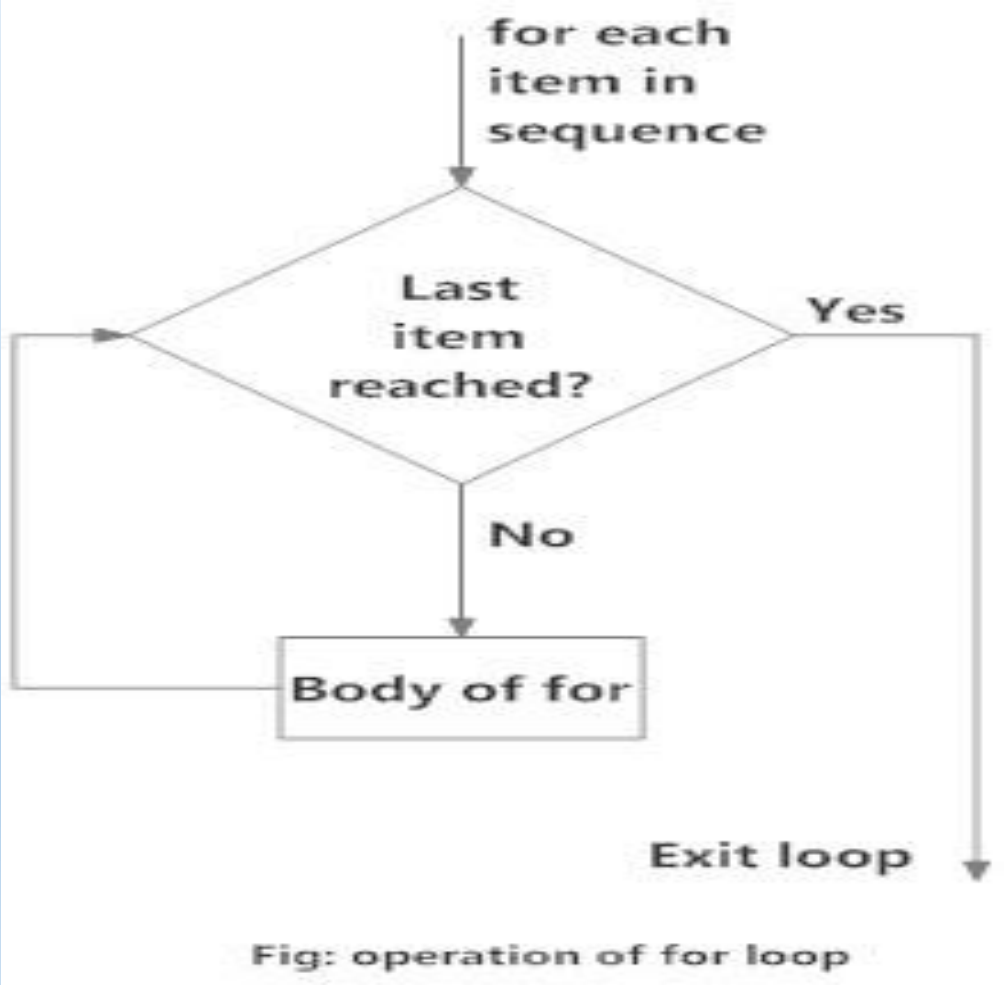


Fig: operation of for loop

Laço for (cont.)

- Exemplo: Seja a tarefa de computar a soma de uma lista de números. O calculo desta soma pode ser feito utilizando-se um laço como mostrado a seguir, admitindo-se que data referencia a lista de números

```
total = 0
for val in data:
  total += val
```

- The loop body executes once for each element of the data sequence, with the identifier, val, from the for-loop syntax assigned at the beginning of each pass to a respective element.
- It is worth noting that val is treated as a standard identifier.
- If the element of the original data happens to be mutable, the val identifier can be used to invoke its methods.
- But a reassignment of identifier val to a new value has no affect on the original data, nor on the next iteration of the loop.

Laço for (cont.)

- Exemplo: Seja a tarefa de obter o valor máximo de uma lista de números. O calculo desta soma pode ser feito utilizando-se um laço como mostrado a seguir, admitindo-se que data referencia a lista de números e que não é vazia.

```
biggest = data[0]
for val in data:
    if val > biggest:
        biggest = val
```

- Although we could accomplish both of the above tasks with a while loop, the for-loop syntax had an advantage of simplicity, as there is no need to manage an explicit index into the list nor to author a Boolean loop condition.
- Furthermore, we can use a for loop in cases for which a while loop does not apply, such as when iterating through a collection, such as a set, that does not support any direct form of indexing.

Laço for (cont.)

- Exemplos de laços

```
>>> # Program to find the sum of all numbers stored in
a list
...
>>> # List of numbers
... numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
>>>
>>> # variable to store the sum
... total = 0
>>>
>>> # iterate over the list
... for val in numbers:
...     total = total+val
...
>>> # Output: The sum is 48
... print("The sum is", total)
The sum is 48
```

```
>>> # Traversing In A Python Tuple
... py_tuple = ('p', 'y', 't', 'h', 'o', 'n')
>>> for item in py_tuple:
...     print("Item:", item)
...
Item: p
Item: y
Item: t
Item: h
Item: o
Item: n
```

Laços com Indexação

- Uma limitação dos laços visto até agora é que não se sabe onde um elemento se localiza dentro de uma sequência
  - Em alguns cenários, é necessário saber o índice de um elemento dentro de uma sequencia, como por exemplo, no caso da localização do maior valor de uma lista de números, quando deseja-se saber o índice deste valor
- Assim, em vez de realizar uma iteração sobre os valores de uma sequencia, pode-se iterar sobre os índices
- No Python há uma classe pré-fabricada chamada `range` que gera sequências inteiras
  - Na sua forma mais simples, `range(n)` gera uma série de *n* valores de 0 a *n*-1, que são precisamente os índices de uma de uma sequencia de tamanho *n*

Laços com Indexação (cont.)

- A sintaxe para se percorrer uma sequencia de dados por meio de seus índices é a seguinte:

```
for j in range (len(data)):
```

- In this case, identifier j is not an element of the data—it is an integer
- But the expression data[j] can be used to retrieve the respective element

Laços com Indexação (cont.)

- Exemplo: Obter o índice do elemento de maior valor de uma lista.

```
big_index = 0
for j in range(len(data)):
  if data[j] > data[big_index]:
    big_index = j
```

Laços com Indexação (cont.)

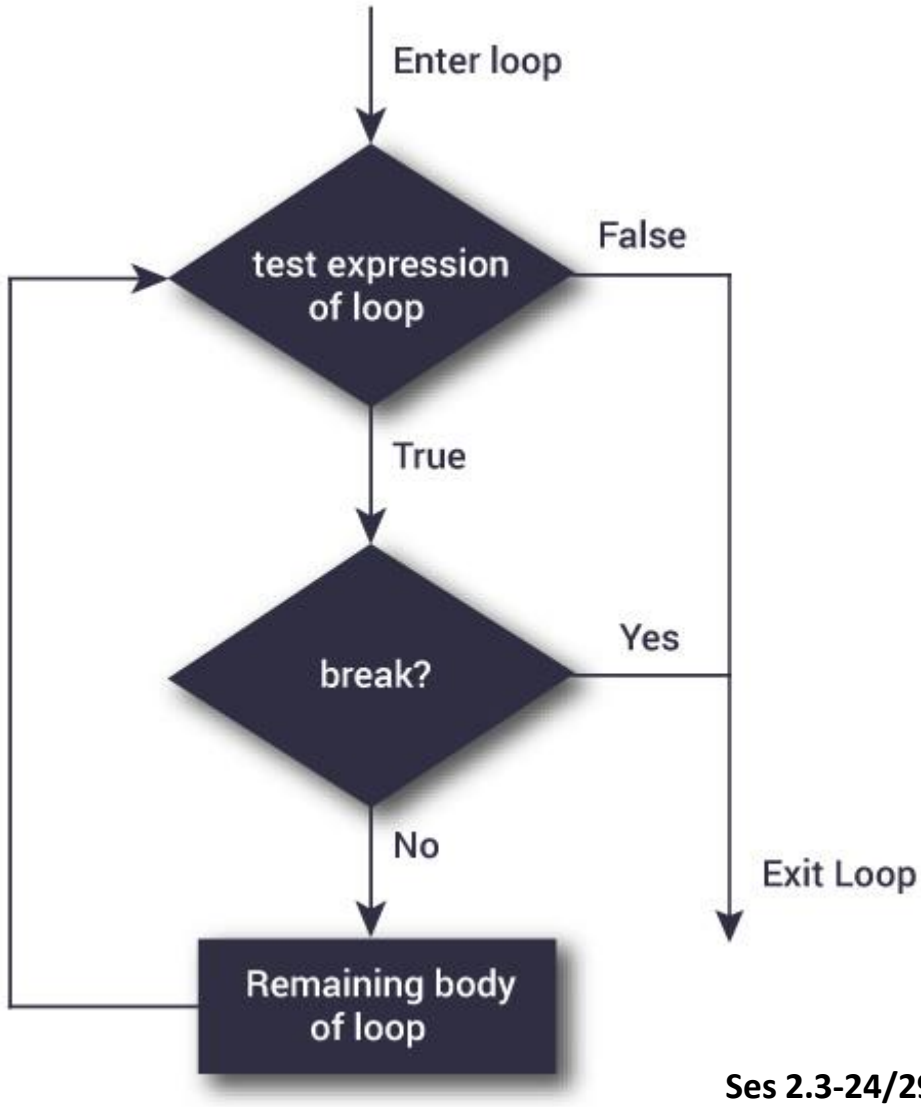
- Exemplo: Imprimir uma lista de gêneros de músicas utilizando-se de indexação

```
# Program to iterate through a list using indexing

genre = ['pop', 'rock', 'jazz']

# iterate over the list using index
for i in range(len(genre)):
        print("I like", genre[i])
```

Instrução break

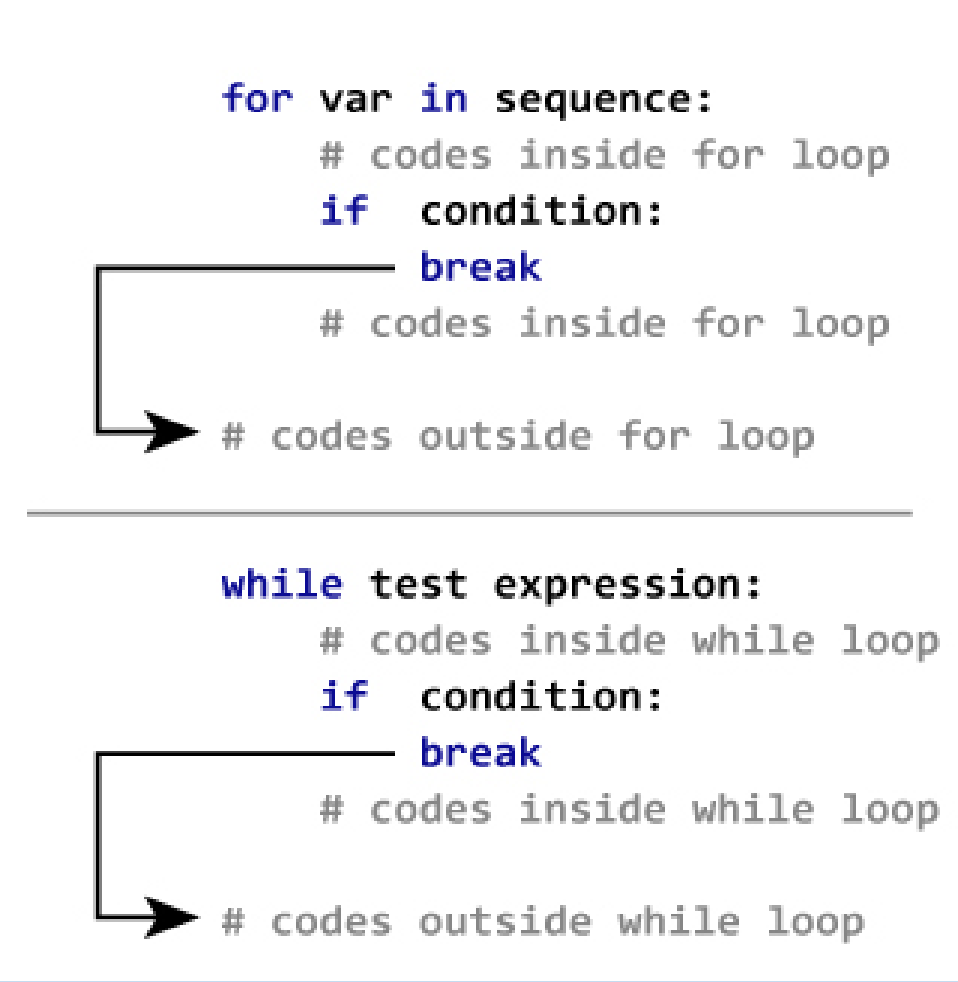
- A linguagem Python disponibiliza uma instrução chamada break que termina de forma imediata a execução de laços while ou for, quando executada dentro de seu bloco de código
  - Quando ela é aplicada dentro de estruturas de controle internas, ela termina o bloco de controle mais interno
- Sintaxe:

```
break
```



Instrução break (cont.)

- Exemplos:



```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

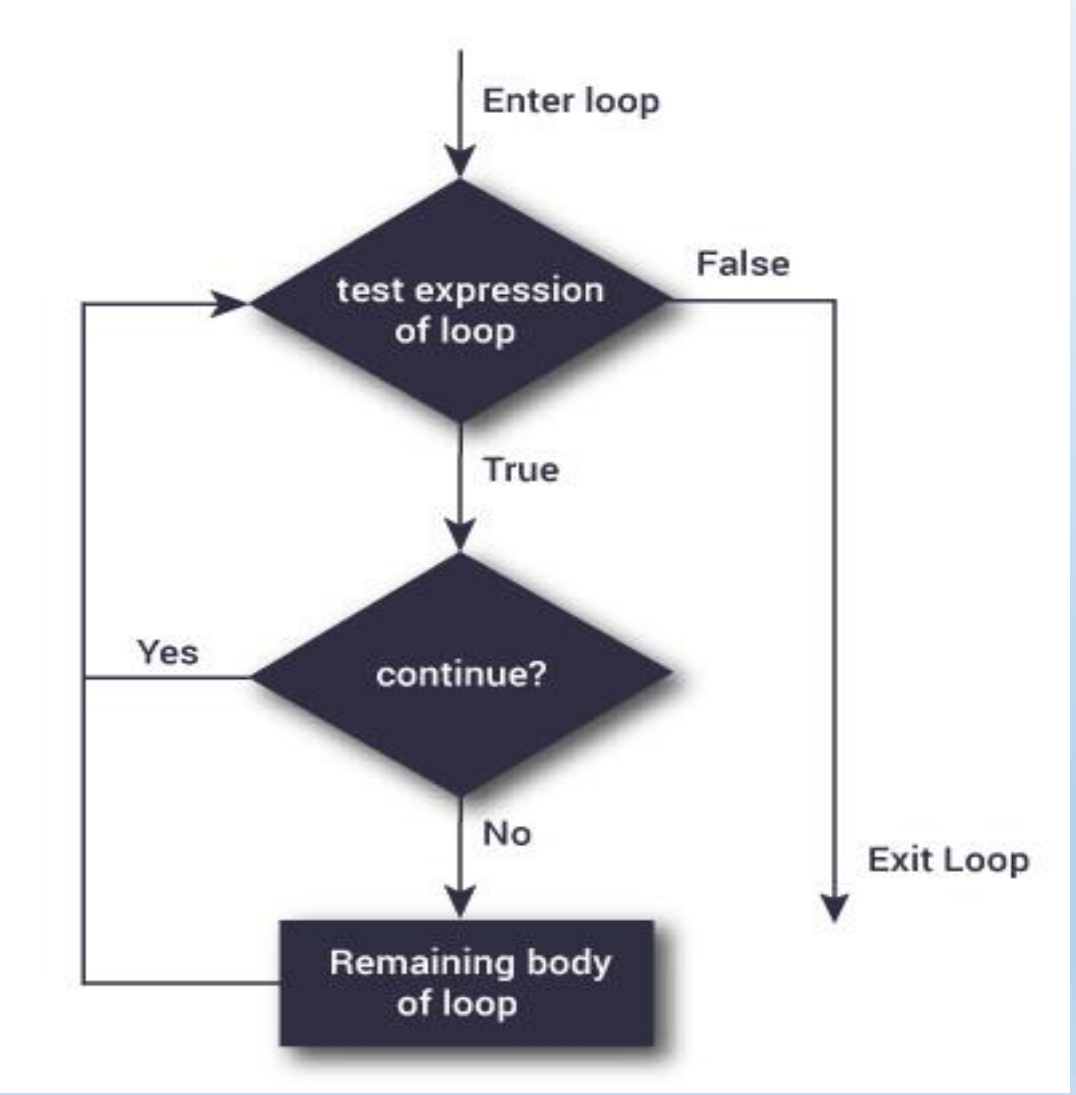
```
# Use of break statement inside loop
# First Example
for val in "string":
   if val == "i":
      break
   print(val)
print("The end")
```

```
# Use of break statement inside loop
# Second Example
var = 10
while var > 0:
  print ('Current variable value :', var)
  var = var -1
  if var == 5:
     break
print ("Good bye!“)
```

Instrução continue

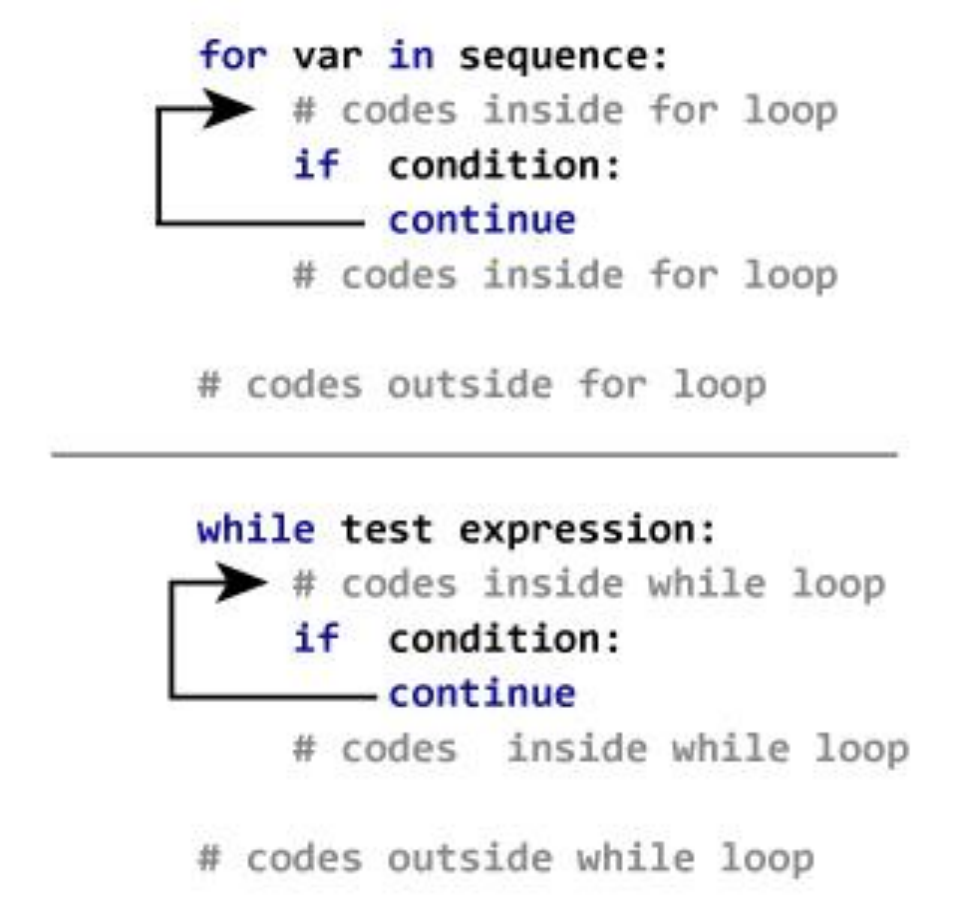
- O Python também disponibiliza a instrução continue, que faz a iteração corrente do laço parar e voltar para o início do laço
- Ou seja, as instruções do laço não são executadas e ele continua na próxima iteração
- Sintaxe:

```
continue
```



Instrução continue (cont.)

- Exemplos:




```
# Program to show the use of continue statement inside loops
# First Example
for val in "string":
    if val == "i":
        continue
        print(val)
print("The end")
```

```
# Second Example
var = 10
while var > 0:
   var = var-1
   if var == 5:
      continue
      print ('Current variable value :', var)
print ("Good bye!")
```

Instrução for loop with else

- Um laço for pode uma cláusula else opcional, que será executada se quando acaba-se a iteração sobre os elementos da sequencia
  - Quando a instrução break é usada para terminar o laço, a cláusula else é ignorada.
  - Assim, a clausula else só será executada se nenhuma instrução break for executada

```
>>> for i in digits:
...     print(i)
... else:
...     print("No items left.")
...
0
1
5
No items left.
```

Instrução pass

- A instrução pass é usada quando uma instrução é necessária sintaticamente, mas não há nenhum comando a ser executado
- A instrução funciona como um marcador para futuras instruções
- Sintaxe

```
pass
```

```
for letter in 'Python':
   if letter == 'h':
      pass
      print ('This is pass block')
   print ('Current Letter :', letter)
print ("Good bye!")
```

```
============ RESTART: C:/Python-36/Codigo-Python-
PEDS/pass1.py ============
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
```