

CC8210 – NCA210

Programação Avançada I

Prof. Reinaldo A. C. Bianchi
Prof. Leandro Alves da Silva
Prof. Isaac Jesus da Silva
Prof. Danilo H. Perico

Operadores Relacionais

Operadores Relacionais

- Operadores relacionais são utilizados para se realizar comparações entre valores;
- Estes valores podem ou não estar armazenados em variáveis.
- O resultado de toda comparação é um tipo lógico: **True** ou **False**

Operadores Relacionais

operador	operação	Símbolo matemático
==	Igualdade	=
>	Maior que	>
<	Menor que	<
!=	diferente	≠
>=	Maior ou igual	≥
<=	Menor ou igual	≤

Exemplos

```
In [26]: 5 > 9
```

```
Out[26]: False
```

```
In [27]: 9 == 9
```

```
Out[27]: True
```

```
In [28]: 7 > 3
```

```
Out[28]: True
```

```
In [29]: 4 != 6
```

```
Out[29]: True
```

```
In [30]: 2 >= 2
```

```
Out[30]: True
```

```
In [32]: a = 10  
b = -5
```

```
b >= a
```

```
Out[32]: False
```

```
In [33]: b <= a
```

```
Out[33]: True
```

```
In [34]: b != a
```

```
Out[34]: True
```

```
In [35]: b == a
```

```
Out[35]: False
```

Estrutura Condicional

Estrutura Condicional

- Nem sempre todas as linhas do código devem ser executadas!
- Normalmente, o programa deve decidir quais partes serão executadas com base em uma ou mais condições
- As condições são construídas com operadores relacionais: são feitas com base no resultado de comparações!

Comando **if**

- Em Python, e em várias outras linguagens de programação, o comando principal para a realização de decisões é o **if**
- Sintaxe do **if**

```
if <condição>:  
    bloco verdadeiro
```
- **If** nada mais é do que nosso **se**
- Em português, podemos entender o comando if da seguinte forma:
 - **Se a condição for verdadeira, faça alguma coisa**

Comando **if** - Exemplo

Ler dois valores e apresentar o maior deles:

```
a = int(input("Primeiro Valor: "))  
b = int(input("Segundo Valor: "))  
  
if a > b:  
    print("O primeiro é o maior!")  
  
if b > a:  
    print("O segundo é o maior!")
```

```
Primeiro Valor: 87  
Segundo Valor: 54  
O primeiro é o maior!
```

Comando **if** - indentação

- O bloco que será executado se a condição do if for verdadeira fica **indentado** com relação ao comando if
- **Indentação** é o **recuo** (deslocamento do texto à direita)
- **Indentação:** neologismo derivado da palavra em inglês *indentation*
- *BLOCOS SÃO DEFINIDOS PELA INDENTAÇÃO!!!*

Comando **if** - indentação

```
if a > b:  
    print("O primeiro é o maior!")  
    print(a)  
    print("fim do if")
```

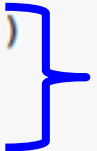
↑
indentação

Comando **if** - indentação - Exemplo

```
a = int(input("Primeiro Valor: "))
b = int(input("Segundo Valor: "))

if a > b:
    print("O primeiro é o maior!")
    print(a)
    print("fim do if")

print("Este print() executa de forma independente com relação à condição a > b")
```



Estes são os comandos que pertencem ao bloco do if

Comando **if** - Exemplo

Voltando ao exemplo do maior entre dois números, o que acontece se os valores digitados para os dois números forem iguais???

```
a = int(input("Primeiro Valor: "))
b = int(input("Segundo Valor: "))

if a > b:
    print("O primeiro é o maior!")

if b > a:
    print("O segundo é o maior!")
```

Comando **if** - Exemplo

Voltando ao exemplo do maior entre dois números, o que acontece se os valores digitados para os dois números forem iguais???

Nenhuma das duas
condições será verdadeira!

Nenhum print() será
chamado!

Comando **else**

- O comando **else** (senão) é utilizado nos casos em que a segunda condição é simplesmente o **contrário** da primeira.
- Sempre utilizado como uma sequência de um **if**
- Sintaxe:

```
if <condição>:  
    bloco verdadeiro  
else:  
    bloco contrário
```

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Sun Aug  9 16:26:51 2020
5
6  @author: rbianchi
7  """
8
9  a = int(input("Digite o primeiro valor: "))
10 b = int(input("Digite o segund valor: "))
11
12 if a > b:
13     print ("0 primeiro numero é maior")
14
15 if b > a:
16     print ("0 segundo numero é maior")
17
18 else:
19     print ("0s numeros são iguais!")
20
21
```

Var	Type	Size	Value
a	int	1	10
b	int	1	10

Variable explorer Help Plots Files

Console 1/A

```
In [21]: runfile('/Users/rbianchi/Documents/AULAS/CC8210 - PROGRAMAÇÃO
AVANÇADA I/Semana 01/exemplo02-if-else.py', wdir='/Users/rbianchi/
Documents/AULAS/CC8210 - PROGRAMAÇÃO AVANÇADA I/Semana 01')
```

Digite o primeiro valor: 10

Digite o segund valor: 11
0 segundo numero é maior

```
In [22]: runfile('/Users/rbianchi/Documents/AULAS/CC8210 - PROGRAMAÇÃO
AVANÇADA I/Semana 01/exemplo02-if-else.py', wdir='/Users/rbianchi/
Documents/AULAS/CC8210 - PROGRAMAÇÃO AVANÇADA I/Semana 01')
```

Digite o primeiro valor: 10

Digite o segund valor: 10
0s numeros são iguais!

In [23]:

IPython console History

Condições aninhadas

- Nem sempre nossos programas são tão simples!
- Precisamos, muitas vezes, aninhar vários **ifs** para obter o comportamento desejado no programa
- Aninhar significa colocar um **if** dentro do outro (ou um **if** dentro do **else**).

Condições aninhadas - Exemplo

- Calcular o preço de uma conta de telefone que é formada por preços diferenciados: acima de 400 min, R\$ 0,15 por min; abaixo de 400 min, R\$ 0,18 por min; e abaixo de 200 min, R\$ 0,20 por min.

Condições aninhadas - Exemplo

```
minutos = int(input("Quantos minutos foram utilizados este mês: "))

if minutos > 400:
    preco = 0.15
else:
    if minutos < 200:
        preco = 0.2
    else:
        preco = 0.18

print("O valor da sua conta é R$ %.2f" % (minutos*preco))
```

```
Quantos minutos foram utilizados este mês: 150
O valor da sua conta é R$ 30.00
```

Comando **elif**

- O comando **elif** (*else if* - senão se) substitui, em muitos casos, a necessidade do aninhamento
- É sempre utilizado como uma sequência de um **if**
- Sintaxe:

```
if <condição 1>:  
    #bloco se a condição 1 for verdadeira  
elif <condição 2>:  
    #bloco se a condição 1 for falsa e a condição 2 verdadeira  
else:  
    #bloco contrário a todas outras condições
```

Comando **elif** - Exemplo

- Exemplo da conta de telefone alterado para usar **elif**

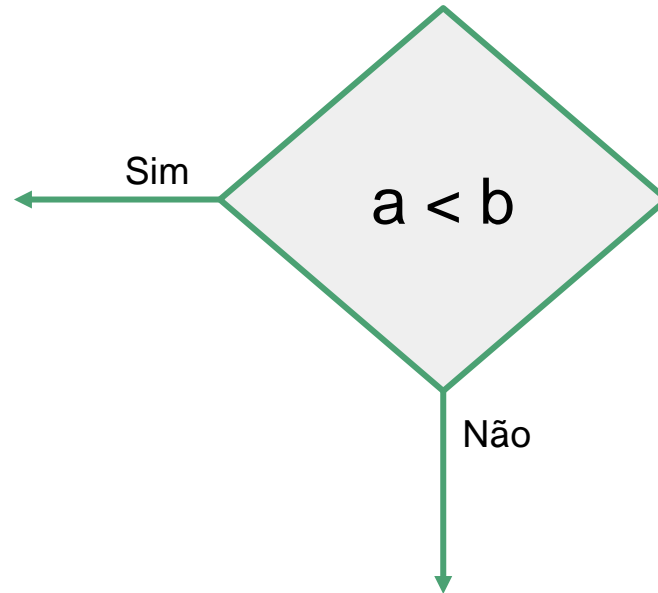
```
minutos = int(input("Quantos minutos foram utilizados este mês: "))

if minutos < 200:
    preco = 0.20
elif minutos < 400:
    preco = 0.18
else:
    preco = 0.15

print("O valor da sua conta é R$ %.2f" % (minutos*preco))
```

```
Quantos minutos foram utilizados este mês: 485
O valor da sua conta é R$ 72.75
```

Símbolo no fluxograma - condição



Pseudocódigo

SE ($a < b$) **ENTÃO**

Comandos para condição verdadeira

SENÃO

Comandos caso contrário

ou

IF ($a < b$) **THEN**

Comandos para condição verdadeira

ELSE

Comandos caso contrário

Comentários

Comentários

- São textos que não são interpretados como código de programa
- Servem para documentar o programa
- No Python, uma linha de comentário começa com o símbolo # (padrão mais comum)
- Exemplo:

```
a = 8
#isso é um comentário
print(a)
```

8

Operadores Lógicos

Operadores Lógicos

- Podemos combinar condições para determinar como continuar o fluxo de um programa!
- O Python fornece operadores lógicos para permitir a construção de condições mais complexas.
- Os operadores lógicos mais utilizados são:
 - and (E condicional)
 - or (OU condicional)
 - not (NÃO lógico)

Operadores Lógicos

Lembre-se sempre:

- **Operadores relacionais retornam sempre um valor Booleano:**
 - **True** ou **False**

Operadores Lógicos - *and*

- Operador *and*:
 - Também retorna **True** ou **False** na comparação das condições
 - Todas as condições devem ser verdadeiras para o *and* retornar **True**

Operadores Lógicos - *or*

- Operador *or*:
 - Também retorna **True** ou **False** na comparação das condições
 - Basta que uma condição seja verdadeira para o *or* retornar **True**

Operadores Lógicos

AND:

S = A and B

S	A	B
False	False	False
False	False	True
False	True	False
True	True	True

OR:

S = A or B

S	A	B
False	False	False
True	False	True
True	True	False
True	True	True

Exemplo

- Faça um programa que lê um ano como entrada e verifica se esse ano é bissexto.
- Regras para definição de ano bissexto:
 - Se o ano for divisível por 400 ele é bissexto! Acaba aqui!
 - Se o ano não for divisível por 400, para ser bissexto ele deve:
 - Ser divisível por 4
 - Não ser divisível por 100
- Faça o programa com somente 1 if, 1 else, nenhum elif
- Alguns anos bissextos para verificação: 1904, 1920, 1932, 2016

Exemplo

```
ano = int(input('Digite o ano: '))  
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):  
    print('É um ano bissexto')  
else:  
    print('Não é bissexto')
```

```
Digite o ano: 1906  
Não é bissexto
```

Operadores Lógicos - Exemplo Ano Bissexto

ano = 2000

```
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):
    print('É um ano bissexto')
else:
    print('Não é bissexto')
```

Operadores Lógicos - Exemplo Ano Bissexto

ano = 2000

```
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):  
    print('É um ano bissexto')  
else:  
    print('Não é bissexto')
```

Operadores Lógicos - Exemplo Ano Bissexto

ano = 2000

True True and False

```
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):
    print('É um ano bissexto')
else:
    print('Não é bissexto')
```

Operadores Lógicos - Exemplo Ano Bissexto

ano = 2000

True

True

and

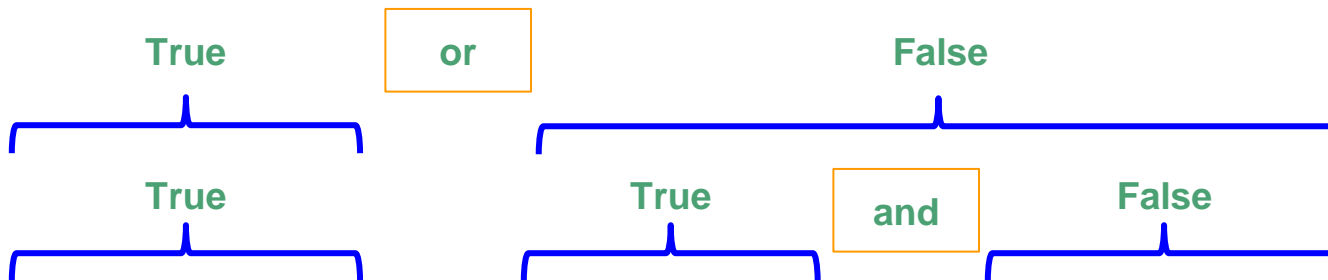
False

False

```
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):
    print('É um ano bissexto')
else:
    print('Não é bissexto')
```

Operadores Lógicos - Exemplo Ano Bissexto

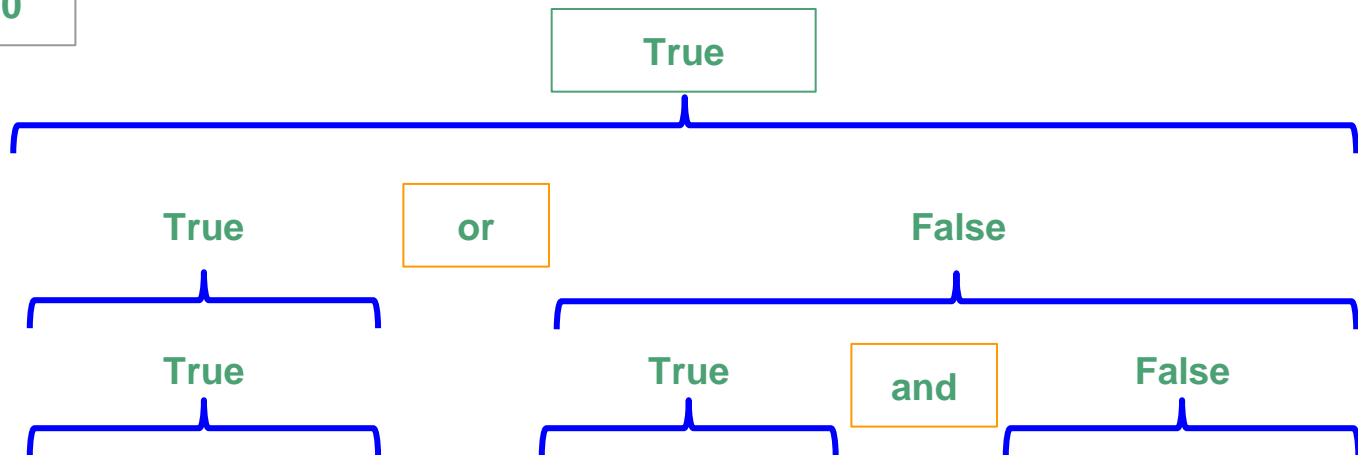
ano = 2000



```
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):
    print('É um ano bissexto')
else:
    print('Não é bissexto')
```

Operadores Lógicos - Exemplo Ano Bissexto

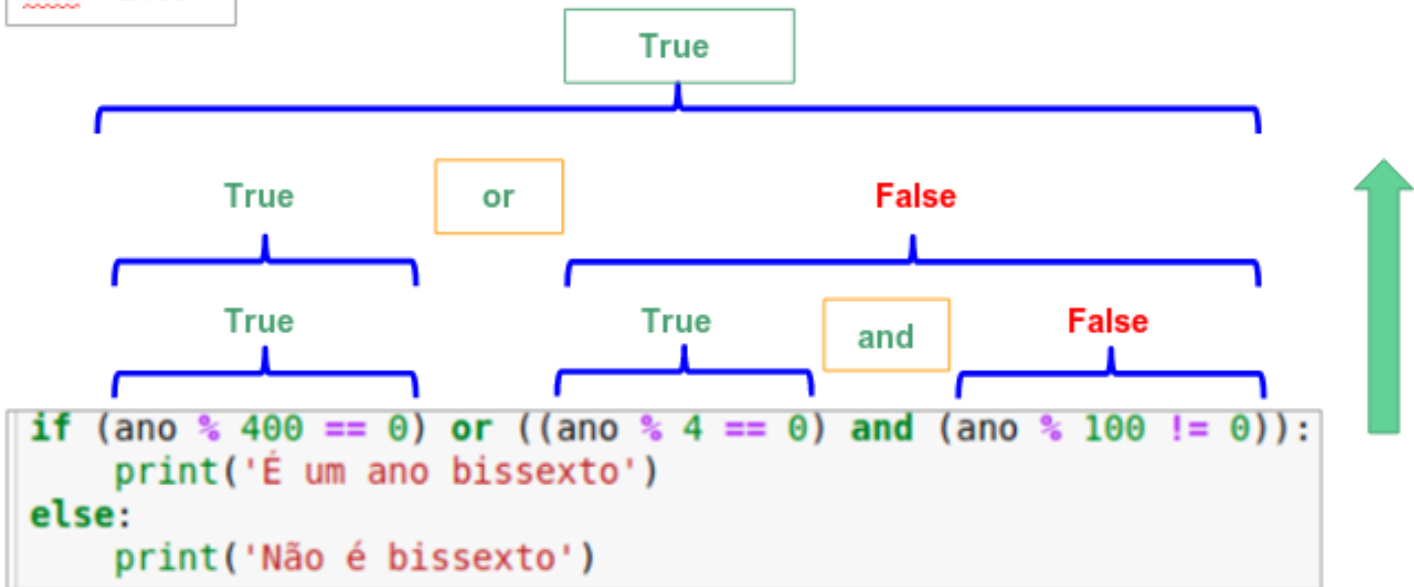
ano = 2000



```
if (ano % 400 == 0) or ((ano % 4 == 0) and (ano % 100 != 0)):
    print('É um ano bissexto')
else:
    print('Não é bissexto')
```

Operadores Lógicos - No Exemplo

ano = 2000



Precedência de Operadores

primeiro



último

**	Exponencial
*, @, /, //, %	Multiplicação, multiplicação de matrizes, divisão, resto
+, -	Adição e Subtração
in, is, is not, <, <=, >, >=, !=, ==	Comparações
not x	NÃO booleano
and	E booleano
or	OU booleano
if - elif - else	Expressões condicionais

Exemplos de Utilização de Operadores

- `x >= y`
- `num1 == num2`
- `valor1 < valor2 and valor1 < valor3`
- `nota >= 4 and nota < 7`
- `valor != x or valor <= y and not valor == z`
- `(nota1 + nota2)/2 >= 7`
- `not(x >= y and y >= z) or x < z * 2`
- `valor2 != valor3 or valor1 == num1/2`

Exercícios

Coloque **True** ou **False** para cada comparação da tabela ao lado dados os valores das variáveis

- $a = 4$
- $b = 10$
- $c = 5.0$
- $d = 1$
- $f = 5$

$b \geq 4 \text{ and } b < 7$	
$a < b \text{ and } a < c$	
$a < b \text{ or } a < c$	
$a < b \text{ and } a < d$	
$a < b \text{ or } a < d$	
$(b + f)/2 \geq 7$	
$c \neq f \text{ or } f == b/2$	
$\text{not } a \neq d$	
$a \neq b \text{ or } a \leq c \text{ and not } a == d$	
$\text{not}(a \geq b \text{ and } b \geq f) \text{ or } a < d * 2$	

Estruturas de Repetição

Repetições

- São utilizadas para executar **várias vezes** a mesma parte do programa
- Normalmente dependem de uma condição
- Repetições são a base de vários programas!

Exemplo

- Fazer um programa para imprimir 10 números sequenciais na tela, começando do número 1.

Exemplo

- Esta é uma solução para o problema do exemplo:
- É uma solução boa?!
- Não: **Muitos comandos repetidos!**

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

```
1
2
3
4
5
6
7
8
9
10
```

Estruturas de Repetição

Comando *while*

Comando **while**

- O comando **while** (enquanto) serve para executarmos alguma repetição **enquanto** uma condição for verdadeira (True)

```
• while <condição>:  
    #bloco que será repetido enquanto a condição for verdadeira
```

Comando **while**

```
x = 1  
while x <= 10:  
    print(x)  
    x = x + 1
```

1
2
3
4
5
6
7
8
9
10

Exemplo

- Impressão do número 1 até o número digitado pelo usuário:

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 1  
  
while i <= ultimo:  
    print(i)  
    i += 1
```

Equivalente a $i = i + 1$

Digite o último dígito da contagem: 5

1
2
3
4
5

- O lado direito do sinal de igual (=) é executado primeiro!
- O resultado é atribuído para a variável que estiver do lado esquerdo do sinal de igual (=)

Exemplo

- Impressão do número 1 até o número digitado pelo usuário:

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 1  
  
while i <= ultimo:  
    print(i)  
    i += 1
```

i é um contador

um contador é uma
variável que conta o
número de ocorrências de
um evento: neste caso, o
número de repetições!

```
Digite o último dígito da contagem: 5  
1  
2  
3  
4  
5
```

Exemplo - combinando repetição com **if**

- Programa que imprime todos os números pares de 0 até um número digitado pelo usuário.

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 0  
  
while i <= ultimo:  
    if i % 2 == 0:  
        print(i)  
    i += 1
```

Exemplo - combinando repetição com **if**

- Programa que imprime todos os números pares de 0 até um número digitado pelo usuário.

```
ultimo = int(input("Digite o último dígito da contagem: "))  
  
i = 0  
  
while i <= ultimo:  
    if i % 2 == 0:  
        print(i)  
    i += 1
```

```
Digite o último dígito da contagem: 6  
0  
2  
4  
6
```

while infinito

- Muitas vezes, queremos que nossos programas sejam executados infinitamente
- Nesses casos, podemos utilizar uma condição que nunca deixe de ser verdadeira (True)

while infinito

```
while True:  
    #bloco que sempre será executado,  
    #nunca sai do loop de repetição
```


Comando **break**

- Porém, mesmo quando utilizamos um **while** infinito, é possível que em determinadas situações o programa precise sair do loop de repetição.
- Esta interrupção pode ser alcançada com o comando **break**
- O comando **break** pode ser utilizado para interromper o while, independentemente da condição

Comando **break** - Exemplo

- Somatória de valores digitados pelo usuário até que o número 0 (zero) seja digitado; quando 0 for digitado o resultado da somatória é exibido:

```
somatoria = 0

while True:
    entrada = int(input("Digite um número a somar ou 0 para sair:"))
    if entrada == 0:
        break
    else:
        somatoria = somatoria + entrada

print("Somatória", somatoria)
```

```
Digite um número a somar ou 0 para sair:5
Digite um número a somar ou 0 para sair:6
Digite um número a somar ou 0 para sair:4
Digite um número a somar ou 0 para sair:0
Somatória 15
```

Repetições aninhadas

- Podemos combinar vários **while**, um dentro do outro!
- Com isso, conseguimos alterar automaticamente o valor de mais do que somente uma variável

Exemplo

- Programa para calcular as tabuadas do número 1 até o número 10.

```
tabuada = 1
while tabuada <= 10:
    print()
    print("Tabuada do", tabuada)
    multiplicador = 0
    while multiplicador <= 10:
        print(tabuada, "x", multiplicador, "=", (tabuada*multiplicador))
        multiplicador += 1
    tabuada += 1
```

```
Tabuada do 1
1 x 0 = 0
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 0 = 0
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
```

Estruturas de Repetição

Comando *for*

Comando for

- for é a estrutura de repetição mais utilizada

- Sintaxe:

```
for <referência> in <sequência>:  
    #bloco de código que será repetido  
    #a cada iteração
```

- Durante a execução, a cada iteração, a referência aponta para um elemento da sequência.
- Uma vantagem do for com relação ao while é que o contador não precisa ser explícito!

Comando **for** - Exemplo

- Calcular a somatória dos números de 0 a 99

```
somatoria = 0  
  
for x in range(0,100):  
    somatoria = somatoria + x  
print(somatoria)
```

4950

A função **range(i, f, p)** é bastante utilizada nos laços com **for**

Ela gera um conjunto de valores inteiros:

- Começando de **i**
- Até valores menores que **f**
- Com passo **p**

Se o passo **p** não for definido, o padrão de 1 será utilizado.

Exercício

- Faça um programa que gera 100 vezes um número aleatório entre 1 e 100 e, então, exiba qual foi o maior número gerado e quantas vezes o maior número foi atualizado no seu código.

Para isso, você deve comparar o número gerado na iteração presente com o maior número armazenado até o momento.

A função `randrange(i, f)` gera números inteiros de `i` até valores menores que `f`

```
from random import randrange  
numero = randrange(1, 101)
```


Comando **else** na repetição

- É possível a utilização do comando **else** nas estruturas de repetição
- Tanto no **while** quanto no **for**
- A cláusula **else** só é executada quando a condição do *loop* se torna falsa.
- Se você sair do *loop* com o comando **break**, por exemplo, ela não será executada.

Comando **else** na repetição

```
i = 0
while i < 11:
    print(i)
    i+=2
else:
    print("Os números pares de 0 a 10 foram exibidos")
```

```
0
2
4
6
8
10
Os números pares de 0 a 10 foram exibidos
```

```
for i in range(0,11,2):
    print(i)
else:
    print("Os números pares de 0 a 10 foram exibidos")
```

```
0
2
4
6
8
10
Os números pares de 0 a 10 foram exibidos
```

Comando **else** na repetição

- Exemplo com **break**: o **else** não é executado

```
1 i = 0
2
3 while i < 11:
4     print(i)
5     i+=2
6     if i == 8:
7         break
8 else:
9     print("Os números pares de 0 a 10 foram exibidos")
10
```

0
2
4
6


Comando **continue** na repetição

- O comando **continue** funciona de maneira parecida com o **break**, porém o break interrompe e sai do *loop*;
- Já o **continue** faz com que a próxima iteração comece a ser executada, não importando se existem mais comandos depois dele ou não
- O **continue** não sai do *loop*
- O **continue** faz com que a próxima iteração seja executada imediatamente

Comando **continue** na repetição

- Exemplo com **continue**

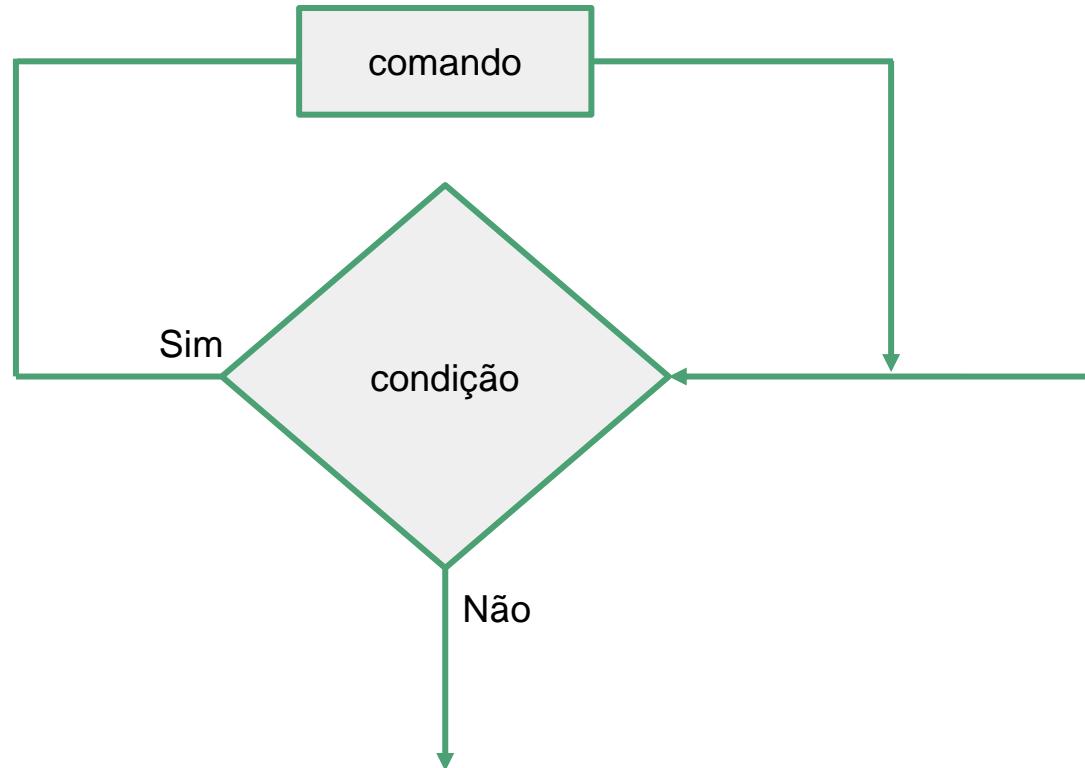
```
1 i = 0
2
3 while i < 12:
4     i+=2
5     if i == 8:
6         continue
7     print(i)
8 else:
9     print("Os números pares de 2 a 12 foram exibidos, com exceção do 8")
```



- Chama a próxima iteração
- Não executa os comandos da própria iteração: neste caso pula o *print()* com *i = 8*

```
2
4
6
10
12
Os números pares de 2 a 12 foram exibidos, com exceção do 8
```

Símbolo no fluxograma - repetição



Pseudocódigo

ENQUANTO ($a < b$) **FAÇA**

Comandos para condição verdadeira

FIM-ENQUANTO

ou

WHILE ($a < b$) **DO**

Comandos para condição verdadeira

END-WHILE

Conclusão

- Vimos na aula de hoje todos os comandos básicos para programar em Python:
- Comandos de Decisão:
 - if, else, elif
- Comandos de Repetição:
 - while, for
- Já podemos começar a fazer exercícios!