



Algoritmos e Programação de Computadores

Expressões Relacionais, Lógicas e Comandos Condicionais

Profa. Sandra Avila

Instituto de Computação (IC/Unicamp)

MC102, 13 Março, 2019

Agenda

- Expressões relacionais
- Expressões lógicas
- Comandos condicionais
- Exercícios

Expressões Relacionais

Tipo bool

- Em Python o tipo **bool** especifica os valores booleanos falso (`False`) e verdadeiro (`True`).
- Podemos criar variáveis associadas a booleanos, mas o uso mais comum é na verificação de resultados de expressões relacionais e lógicas.

```
a = True
print(type(a))
<class 'bool'>
```

Expressões

- Já vimos que constantes e variáveis são expressões.

```
a = 10  
b = 50  
a = b
```

- Vimos também que operações aritméticas também são expressões.

```
a = 2 * 2  
a = 10 / 3  
a = a + 1
```

Expressões Relacionais

- Expressões relacionais são aquelas que realizam uma **comparação** entre duas expressões e retornam
 - **False**, se o resultado é falso.
 - **True**, se o resultado é verdadeiro.

Operadores Relacionais

- Os operadores relacionais da linguagem Python são:
 - == : igualdade
 - != : diferente
 - > : maior que
 - < : menor que
 - >= : maior ou igual que
 - <= : menor ou igual que

Expressões Relacionais

- expressão == expressão : Retorna verdadeiro quando as expressões forem iguais.

```
9 == 9  
True  
9 == 10  
False
```

- expressão != expressão : Retorna verdadeiro quando as expressões forem diferentes.

```
9 != 9  
False  
9 != 10  
True
```

Expressões Relacionais

- expressão > expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita.

```
9 > 5  
True
```

- expressão < expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita.

```
9 < 5  
False
```

Expressões Relacionais

- expressão \geq expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.

```
9 >= 5  
True
```

- expressão \leq expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.

```
9 <= 5  
False
```

Expressões Relacionais

- Quais das seguintes opções é uma expressão booleana?

a. True

b. $3 == 4$

c. $3 + 4$

d. $3 + 4 == 7$

e. "False"

```
True           # sim, é uma expressão booleana
True
3 == 4        # sim, é uma expressão booleana
False
3 + 4         # não é uma expressão booleana
7
3 + 4 == 7    # sim, é uma expressão booleana
True
"False"       # não é uma expressão booleana
'False'
```

Expressões Relacionais

```
a = 3
b = 4
c = a < b      # c recebe o valor da comparação a < b
d = a > b      # d recebe o valor da comparação a > b
e = a == b     # e recebe o valor da comparação a == b

print("Valor de c:", c)
print("Valor de d:", d)
print("Valor de e:", e)
```

```
Valor de c: True
Valor de d: False
Valor de e: False
```

Expressões Lógicas

Expressões Lógicas

- Expressões lógicas são aquelas que realizam uma operação lógica (**ou**, **e**, **não**, etc...) e retornam `True` ou `False` (como as expressões relacionais).
- Na linguagem Python temos os seguintes operadores lógicos:
 - **and** : operador E
 - **or**: operador OU
 - **not**: operador NÃO

Expressões Lógicas

- expressão **and** expressão : Retorna verdadeiro quando **ambas** as expressões são verdadeiras. Sua tabela verdade é:

Op1	Op2	Op1 and Op2
V	V	V
V	F	F
F	V	F
F	F	F

Qual o resultado da expressão lógica abaixo?

```
a = 0
b = 0
( a == 0 and b == 0 )
True
```

Expressões Lógicas

- expressão **or** expressão : Retorna verdadeiro quando **pelos menos uma** das expressões é verdadeira. Sua tabela verdade é:

Op1	Op2	Op1 or Op2
V	V	V
V	F	V
F	V	V
F	F	F

Qual o resultado da expressão lógica abaixo?

```
a = 0
b = 0
( a == 1 or b == 0 )
True
```

Expressões Lógicas

- **not** expressão : Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

Op1	not Op1
V	F
F	V

Qual o resultado da expressão lógica abaixo?

```
a = 0
b = 0
not(a != b)
False
```

Expressões Lógicas

- O que será impresso pelo programa?

```
print(8 > 9 and 10 != 2)
print(14 > 100 or 2 > 1)
print(not(14 > 100) and not(1 > 2))
```

```
False
True
True
```

Expressões Lógicas

- Qual é a expressão correta em Python para verificar se um número armazenado na variável x está entre 0 e 5? (múltiplas respostas)
 - $0 < x < 5$
 - $x > 0$ or $x < 5$
 - $x > 0$ and $x < 5$
 - $x > 0$ and < 5

```
x = 6
0 < x < 5
False
x > 0 or x < 5
True
x > 0 and x < 5
False
x > 0 and < 5
File "<stdin>", line 1
    x > 0 and < 5
                ^
```

Precedência de Operadores

Nível	Categoria	Operadores
7 (alto)	exponenciação	* *
6	multiplicação	*, /, //, %
5	adição	+, -
4	relacional	==, !=, <=, >=, >, <
3	lógico	not
2	lógico	and
1 (baixo)	lógico	or

Comandos Condicionais



Comandos Condicionais

- Um comando condicional é aquele que permite decidir se um determinado bloco de comandos deve ou não ser executado, a partir do resultado de uma expressão relacional ou lógica.

Bloco de
Comandos 1

Bloco de
Comandos 2

Falso

Verdadeiro

Condição

Blocos de Comandos

- É um conjunto de instruções agrupadas.
- Os comandos agrupados do bloco devem estar **indentados** dentro de um comando anterior **seguido de dois pontos**.
- A indentação é feita em geral com 3 espaços em branco (ou quantos você quiser, ou TAB) antes de cada comando que deve estar dentro do bloco.

Comandos Condicionais

- O principal comando condicional é o **if**, cuja sintaxe é:

if expressão relacional ou lógica:

comandos executados se a expressão é verdadeira

- Os comandos são executados somente se a expressão relacional/lógica for verdadeira.

Comandos Condicionais

- O programa determina se um valor é par.

```
# Informa se o número é par.  
numero = int(input())  
if numero % 2 == 0:  
    print("O número digitado é par.")
```

Comandos Condicionais

- Uma variação do comando **if** é o **if/else**, cuja sintaxe é:

if expressão relacional ou lógica:

comandos executados se a expressão é verdadeira

else:

comandos executados se a expressão é falsa

Comandos Condicionais

- O programa determina se um valor é par.

```
# Informa se o número é par.
numero = int(input())
if numero % 2 == 0:
    print("O número digitado é par.")
else:
    print("O número digitado é ímpar.")
```

Comandos Condicionais

- O programa determina o menor de dois números.

```
# Determina o menor de dois números.
numero1 = int(input("Digite um número:"))
numero2 = int(input("Digite um número:"))

if numero1 < numero2:
    print("O menor número é:", numero1)
else:
    print("O menor número é:", numero2)
```

Comandos Condicionais

- Note que o **if** é um comando, e como tal pode aparecer dentro do bloco de comandos de outro **if**.
- Exemplo: Usando apenas **operadores relacionais** e **aritméticos**, vamos escrever um programa que lê um número e verifica em qual dos seguintes casos o número se enquadra:
 - Par e menor que 100
 - Par e maior ou igual a 100
 - Ímpar e menor que 100
 - Ímpar e maior ou igual a 100

Comandos Condicionais

```
numero = int(input("Digite um número:"))

if (numero % 2 == 0): # se o número for par
    if (numero < 100):
        print("O número é par e menor que 100")
    else:
        print("O número é par e maior ou igual que 100")
else: # se o número for ímpar
    if (numero < 100):
        print("O número é ímpar e menor que 100")
    else:
        print("O número é ímpar e maior ou igual que 100")
```

Comandos Condicionais

```
numero = int(input("Digite um número:"))

if (numero % 2 == 0): # se o número for par
    if (numero < 100):
        print("O número é par e menor que 100")
    else:
        print("O número é par e maior ou igual que 100")
else: # se o número for ímpar
    if (numero < 100):
        print("O número é ímpar e menor que 100")
    else:
        print("O número é ímpar e maior ou igual que 100")
```

Se você pudesse usar operadores lógicos, como você poderia refazer este programa?

Comandos Condicionais

```
# Determina o menor de dois números.
numero = int(input("Digite um número:"))

if (numero % 2 == 0) and (numero < 100):
    print("O número é par e menor que 100")
if (numero % 2 == 0) and (numero >= 100):
    print("O número é par e maior ou igual que 100")
if (numero % 2 != 0) and (numero < 100):
    print("O número é ímpar e menor que 100")
if (numero % 2 != 0) and (numero >= 100):
    print("O número é ímpar e maior ou igual que 100")
```

Comandos Condicionais

- Lembre-se que o que define a qual bloco de comandos um comando pertence é a sua indentação!

```
if (cond1):  
    if (cond2):  
        comando1  
else:  
    comando2
```

- Quando o **comando2** é executado?
 - Resposta: quando cond1 for falsa.
 - Resposta: quando a cond1 for verdadeira e cond2 for falsa.

Comandos Condicionais

```
if (cond1):  
    if (cond2):  
        comando1  
    else:  
        comando2  
else:  
    if (cond3):  
        comando3  
    else:  
        comando4
```

- Quando o **comando4** é executado?
 - Resposta: quando a **cond1** for falsa e **cond3** for falsa.

Comandos Condicionais

```
numero = 5
if (numero > 3):
    if (numero < 7):
        print("a")
    else:
        if (numero > -10):
            print("b")
        else:
            print("c")
```

- O que será impresso?
 - Resposta: a

Comandos Condicionais

```
numero = -12
if (numero > 3):
    if (numero < 7):
        print("a")
    else:
        if (numero > -10):
            print("b")
        else:
            print("c")
```

- O que será impresso?
 - Resposta: c

Comandos Condicionais

```
numero = 9
if (numero > 3):
    if (numero < 7):
        print("a")
    else:
        if (numero > -10):
            print("b")
        else:
            print("c")
```

- O que será impresso?
 - Resposta: b

Exercício

- Escreva um programa que lê três números e imprime o maior deles.
- Escreva um programa que lê três números distintos e os imprime em ordem decrescente.

Referências

- O slides dessa aula foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp)
- Decisões e Seleção
 - <https://panda.ime.usp.br/pensepy/static/pensepy/06-Selecao/selecao.html>
 - <https://runestone.academy/runestone/static/thinkcspy/Selection/toctree.html>