Programação Script

Comandos Condicionais

Aula 03

Prof. Felipe A. Louza

Roteiro

1. Expressões relacionais
2. Expressões lógicas
3. Comandos condicionais
4. Comandos `if-elif-else` encaixados
5. Referências

Roteiro

O Tipo `bool`

Em Python o tipo **bool** especifica os valores booleanos falso (`False`) e verdadeiro (`True`).

```
1 >>> a = True
2 >>> type(a)
3 <class 'bool'>
```

Expressão

Na aula passada:

- Vimos que constantes e variáveis são expressões.

```
1 a = 10
2 a = b
```

- Operações aritméticas também são expressões.

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

Expressões relacionais

Expressões relacionais são aquelas que realizam uma **comparação** entre duas expressões e retornam

1. `False`, se o resultado é falso
2. `True`, se o resultado é verdadeiro.

Operadores Relacionais

Os operadores relacionais da linguagem Python:

Operador	Operação
`==`	`igualdade`
`!=`	`diferente`
`>`	`maior`
`<`	`menor`
`>=`	`maior ou igual`
`<=`	`menor ou igual`

- Cuidado: Não confundir atribuição (=) com igualdade (==).

Expressões relacionais

- *expressão* == *expressão*:

```
1 >>> 9 ==9
2 True
3 >>> 9 == 10
4 False
5 >>>
```

- *expressão* != *expressão*:

```
1 >>> 9 != 9
2 False
3 >>> 9 != 10
4 True
5 >>>
```

Expressões relacionais

- *expressão* $>$ *expressão*:

```
1 >>> 9 >5
2 True
```

- *expressão* $<$ *expressão*:

```
1 >>> 9 < 5
2 False
3 >>>
```

Expressões relacionais

- *expressão* >= *expressão*: da direita.

```
1 >>> 9 >= 5
2 True
3 >>>
```

- *expressão* <= *expressão*:

```
1 >>> 9 <= 5
2 False
3 >>>
```

Expressões relacionais

Quais das seguintes opções é uma expressão booleana?

a. `True`
b. `3 == 4`
c. `3+4`
d. `3+4 == 7`
e. `"False"`

Expressões relacionais

O que será impresso pelo programa?

```
1 a = 3
2 b = 4
3 c = a < b  # c recebe o valor da comparação a < b
4 d = a > b  # d recebe o valor da comparação a > b
5 e = a == b # e recebe o valor da comparação a == b
6
7 print("Valor de c:", c)
8 print("Valor de d:", d)
9 print("Valor de e:", e)
```

```
1 Valor de c: True
2 Valor de d: False
3 Valor de e: False
```

Roteiro

Expressões lógicas

Expressões lógicas são aquelas que realizam uma operação lógica e retornam `True` ou `False`.

- Em **Python** temos os seguintes operadores lógicos:
  - `and`: operador E.
  - `or`: operador OU.
  - `not`: operador NÃO.

Expressões lógicas

- *expressão* `and` *expressão* : Retorna `True` quando ambas as expressões são verdadeiras:

Op_1	Op_2	Op_1 `and` Op_2
V	V	V
V	F	F
F	V	F
F	F	F

Qual o resultado da expressão lógica abaixo?

```
1 a = 0
2 b = 0
3 (a == 0 and b == 0)
```

Expressões lógicas

- *expressão* `or` *expressão* : Retorna `True` quando pelo menos uma das expressões é verdadeira:

Op_1	Op_2	Op_1 `or` Op_2
V	V	V
V	F	V
F	V	V
F	F	F

Qual o resultado da expressão lógica abaixo?

```
1 a = 0
2 b = 1
3 (a == 0 or b == 0)
```

Expressões lógicas

- `not` *expressão* : Retorna `True` quando a expressão é falsa:

Op_1	`not` Op_1
V	F
F	V

Qual o resultado da expressão lógica abaixo?

```
1 a = 0
2 b = 1
3 not(a != b)
```

Expressões relacionais

Qual é a expressão correta em **Python** para verificar se um número armazenado na variável `x` está entre `0` e `5`? (múltiplas respostas)

a. `0<x<5`
b. `x>0 or x<5`
c. `x>0 and x<5`
d. `x> 0 and < 5`
e. `x<5 and x>0`

O que será impresso pelo programa?

```
1 print(8>9 and 10!=2)
2
3 print(14 > 100 or 2>1)
4
5 print(not (14>100)  and not (1>2)  )
```

```
1 False
2 True
3 True
```

Roteiro

Comandos condicionais

Um comando condicional é aquele que permite decidir se um determinado **bloco de comandos** deve ou não ser executado.

Bloco de comandos

Bloco de comandos em **Python**:

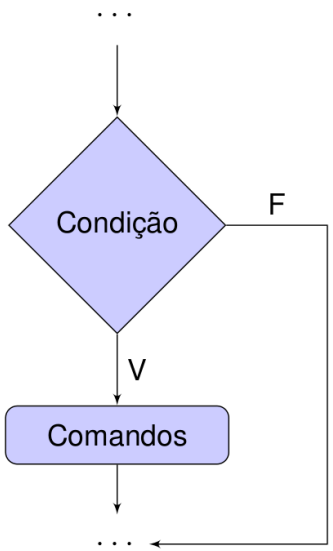
- É um conjunto de instruções agrupadas.
- Os comandos do bloco devem estar **indentados**.
- A indentação é feita em geral com 2 espaços em branco (ou quantos você quiser, ou TAB) antes de cada comando dentro do bloco.

```
1 print("Bloco 0")
2   print("Bloco 1")
3   print("Bloco 1")
4     print("Bloco 2")
5     print("Bloco 2")
6   print("Bloco 1")
```

Comandos condicionais

O principal comando condicional é o `if`, cuja sintaxe é:

```
1 if (expressão relacional ou lógica ):
2   comando1
3   comando2
4   ...
5   comandoN
```



- Os comandos são executados somente se a expressão relacional/lógica for `True`[1].

[1] Os parentêses são opcionais (boa prática).

Exemplo

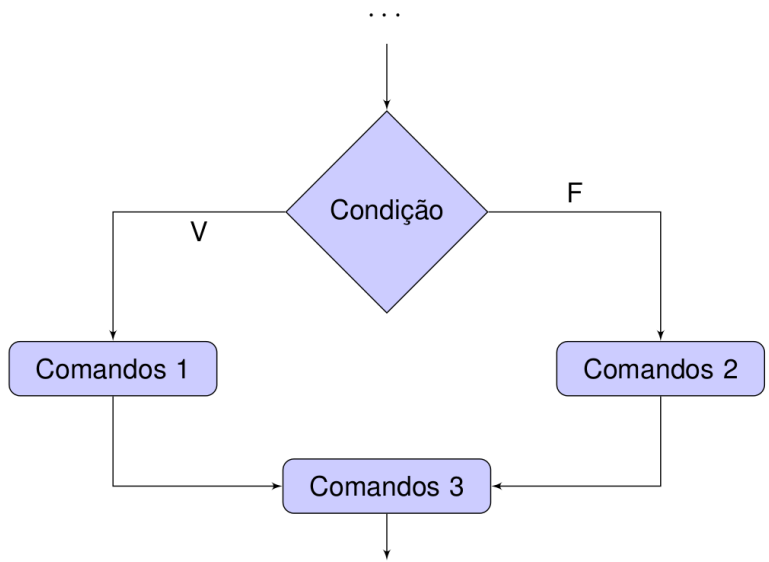
Escreva um programa que determina se o valor lido é par.

```
1 # Informa se o número é par.
2 numero = int(input())
3
4 if (numero % 2 == 0):
5   print("O número digitado é par.")
```

Comandos condicionais

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

```
1  if (expressão relacional ou lógica ):
2    comando1
3    comando2
4    ...
5    comandoN
6  else:
7    comando1
8    comando2
9    ...
10   comandoM
```



Exemplo

Escreva um programa que determina se o valor lido é par ou ímpar.

```
1 # Informa se o número é par.
2 numero = int(input("Digite um número: "))
3
4 if numero % 2 == 0:
5   print("O número digitado é par.")
6 else:
7   print("O número digitado é ímpar.")
```

Exemplo

Escreva um programa que determina o menor de dois números.

```
1 # Determina o menor de dois números.
2 a = int(input("Digite um número: "))
3 b = int(input("Digite um número: "))
4
5 if a < b:
6   print("O menor número é:", a )
7 else:
8   print("O menor número é:", b )
```

Comandos aninhados

Exemplo: Usando apenas **operadores relacionais e aritméticos** ($<, <=, ==, >, >=, \%, \ldots$), vamos escrever um programa que lê um número e verifica se ele é:

- 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 aninhados

```
1  numero = int(input("Digite um número: "))
2
3  if (numero % 2 == 0): # se o número for par
4    if (numero < 100):
5      print("O número é par e menor que 100")
6    else:
7      print("O número é par e maior ou igual que 100")
8  else: # se o número for ímpar
9    if (numero < 100):
10     print("O número é ímpar e menor que 100")
11   else:
12     print("O número é ímpar e maior ou igual que 100"
```

Vamos reescrever esse programa usando operadores lógicos.

Comandos aninhados

```
1  a = int(input("Digite um número: "))
2
3  if (a % 2 == 0 and a<100):
4    print("O número é par e menor do que 100")
5  if (a % 2 == 0 and a>=100):
6    print("O número é par e maior ou igual que 100")
7  if (a % 2 != 0 and a<100):
8    print("O número é ímpar e menor do que 100")
9  if (a % 2 != 0 and a>=100):
10   print("O número é ímpar e maior ou igual que 100")
```

Comandos aninhados

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

```
1 if cond1:
2   if cond2:
3     comando1
4 else:
5   comando2
```

```
1 if cond1:
2   if cond2:
3     comando1
4   else:
5     comando2
```

- Quando o `comando2` é executado?

Comandos aninhados

Outros exemplos:

```
1  if cond1:
2    if cond2:
3      comando1
4    else:
5      comando2
6  else:
7    if cond3:
8      comando3
9    else:
10     comando4
```

```
1  numero = 5
2  if (numero > 3):
3    if (numero < 7):
4      print("a")
5  else:
6    if (numero > -10):
7      print("b")
8    else:
9      print("c")
```

```
1  numero = -12
2  if (numero > 3):
3    if (numero < 7):
4      print("a")
5  else:
6    if (numero > -10):
7      print("b")
8    else:
9      print("c")
```

1. Quando o `comando4` é executado?
2. O que será impresso nos programas 2 e 3?

Exemplo

Escreva um programa que lê três números e imprime o maior deles.

```
1  # Encontra o maior
2  a = int(input("Digite um número: "))
3  b = int(input("Digite um número: "))
4  c = int(input("Digite um número: "))
5
6  if a > b and a > c:
7    print("O maior número é:", a)
8  else:
9    if b > a and b > c:
10     print("O maior número é:", b)
11   else:
12     print("O maior número é:", c)
```

Roteiro

Comandos `if-elif-else`

Quando apenas uma de várias alternativas é `True` podemos usar o `if-elif-else` do **Python**:

```
1  if condicao_1:
2    comando1
3    comando2
4    ...
5    comandoN
6  elif condicao_2:
7    comando1
8    comando2
9    ...
10   comandoM
11 else:
12   comando1
13   comando2
14   ...
15   comandok
```

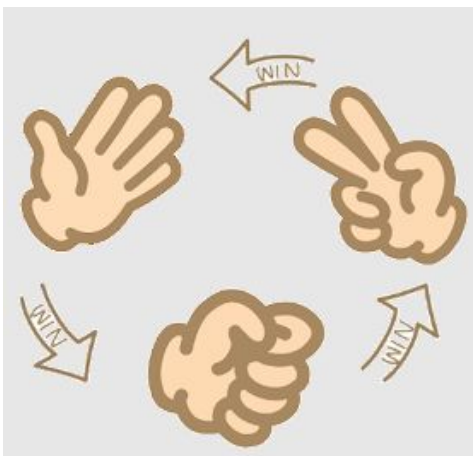
Exemplo

Reescreva o programa que lê três números e imprime o maior deles com a construção `if-elif-else`.

```
1  # Encontra o maior
2  a = int(input("Digite um número: "))
3  b = int(input("Digite um número: "))
4  c = int(input("Digite um número: "))
5
6  if (a > b) and (a > c):
7    print("O maior número é:", a) # a é o maior
8  elif (b > c):
9    print("O maior número é:", b) # b é o maior
10 else:
11   print("O maior número é:", c) # c é o maior
```

Exemplo

Escreva um programa que simula o jogo conhecido como "Pedra, Papel e Tesoura" de um jogador contra outro.



Ganhador	Perdedor
Pedra	Tesoura
Tesoura	Papel
Papel	Pedra

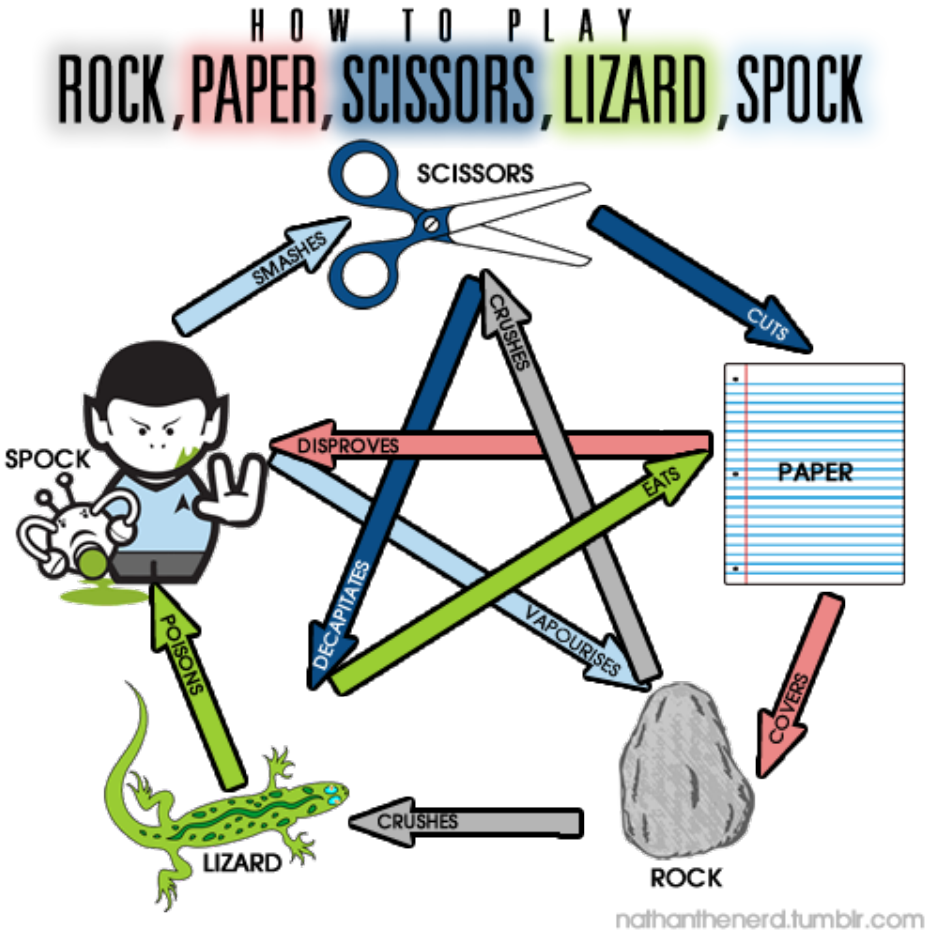
Exemplo

"Pedra, Papel e Tesoura":

```
1  # Entrada dos dados
2  jogador1 = input("Jogador1, digite pedra, papel ou tesoura: ")
3  jogador2 = input("Jogador2, digite pedra, papel ou tesoura: ")
4
5  if (jogador1 == jogador2):
6    print("Empate! Ninguém ganhou.") # empate
7  elif (jogador1 == "pedra" and jogador2 == "tesoura") or \
8       (jogador1 == "tesoura" and jogador2 == "papel") or \
9       (jogador1 == "papel" and jogador2 == "pedra"):
10   print("Jogador 1 ganhou.")
11 else:
12   print("Jogador 2 ganhou.")
```

- A \ no fim de linha indica que o comando continua na próxima linha.

Desafio


HOW TO PLAY
ROCK, PAPER, SCISSORS, LIZARD, SPOCK
SCISSORS
SMASHES
CUTS
CRUSHES
SPOCK
DISPROVES
PAPER
EATS
DECAPITATES
VAPOURISES
POISONS
COVERS
CRUSHES
LIZARD
ROCK
nathanthenerd.tumblr.com

Fim

Dúvidas?

Roteiro

Referências

1. Materiais adaptados dos slides do Prof. Eduardo C. Xavier, da Universidade Estadual de Campinas.