Introdução à Programação para Sensoriamento Remoto

Aula 05 – Introdução à Programação com a Linguagem Python

Gilberto Ribeiro de Queiroz
Thales Sehn Körting
Fabiano Morelli

25 de Março de 2019

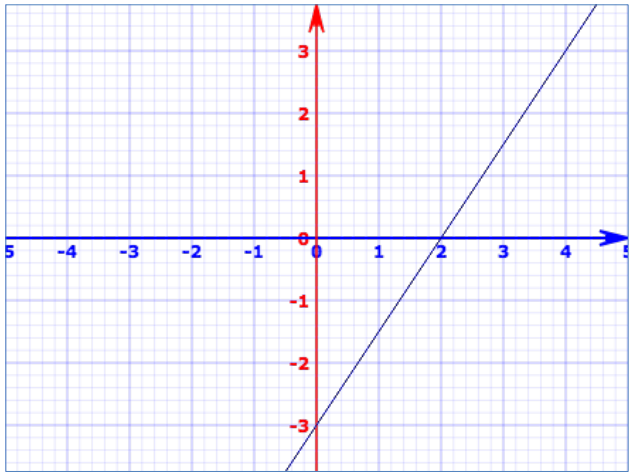
Tópicos

- Tipo Lógico e Operadores Lógicos.
- Operadores Relacionais.
- Expressões Lógicas.
- Controlando o fluxo de um programa com estruturas condicionais.

Aquecimento...

Hands-on

- Uma reta pode ser representada por uma equação geral da seguinte forma: $ax+by+c=0$.
- Considere a reta com os seguintes coeficientes:
  $a=3,\ b=-2$ e $c=-6$
- Faça um programa que pergunte ao usuário os valores de um par de coordenadas (x,y), aplique a equação e verifique o resultado (>0.0, =0.0, <0.0)



$$3x-2y-6=0$$

Tipo Lógico e Operadores Lógicos

Tipo `bool`: Definição

- O tipo `bool` é usado para representar valores **booleanos** ou **lógicos**.

- Este tipo possui apenas dois valores possíveis:
  `True` ou `False`

- Podemos obter o tipo de um valor através do operador **`type`**:

```
>>> type(True)
<type 'bool'>
```

Tipo `bool`: Operadores Lógicos

- As operações usuais sobre os tipos booleanos são conhecidas como **operações lógicas**.

Operador: `and`		
p	**q**	**p ∧ q**
False	False	False
False	True	False
True	False	False
True	True	True

Operador: `or`		
p	**q**	**p ∨ q**
False	False	False
False	True	True
True	False	True
True	True	True

Operador: `not`	
p	**!p**
False	True
True	False

Obs.: Os valores e operadores lógicos são essenciais nos comandos condicionais e nos testes condicionais dos laços de repetição.

Tipo `bool`: Operadores Lógicos

```
>>> False and True
False

>>> False or True
True

>>> not True
False

>>> not False
True
```

Operadores Relacionais ou de Comparação

Operadores Relacionais

- Os **operadores relacionais** ou **operadores de comparação** permitem comparar dois valores e produzir um valor booleano como resultado.

- Esse tipo de operador, juntamente com os operadores lógicos, são essenciais **nos comandos condicionais** e nos **testes condicionais** dos **laços de repetição**.

Operadores Relacionais

Operador	Nome	Expressão	Valor
==	igual	`5 == 4`	`False`
		`5 == 5`	`True`
!=	diferente	`5 != 4`	`True`
		`5 != 5`	`False`
<	menor que	`5 < 5`	`False`
		`4 < 5`	`True`
<=	menor ou igual a	`5 <= 5`	`True`
		`6 <= 5`	`False`
>	maior que	`5 > 4`	`True`
		`4 > 5`	`False`
>=	maior ou igual a	`5 >= 5`	`True`
		`5 >= 6`	`False`

Operadores Relacionais

```
>>> 5 > 4.1
True

>>> "Roger" < "Rogerio"
True

>>> "Roger" < "Rodrigo"
False

>>> 5 != 4.1
True
```

Expressões Lógicas

Exemplo: Ano bissexto?

- Anos bissextos ocorrem a cada quatro anos, exceto anos múltiplos de 100 que não são múltiplos de 400.

```
01 ano = int( input( "Ano: " ) )

02 bissexto = (ano % 4 == 0 and \
               ano % 100 != 0) or \
               ano % 400 == 0

03 print("É bissexto: ", bissexto)
```

Exemplo: Ano bissexto?

- Podemos utilizar outra expressão:

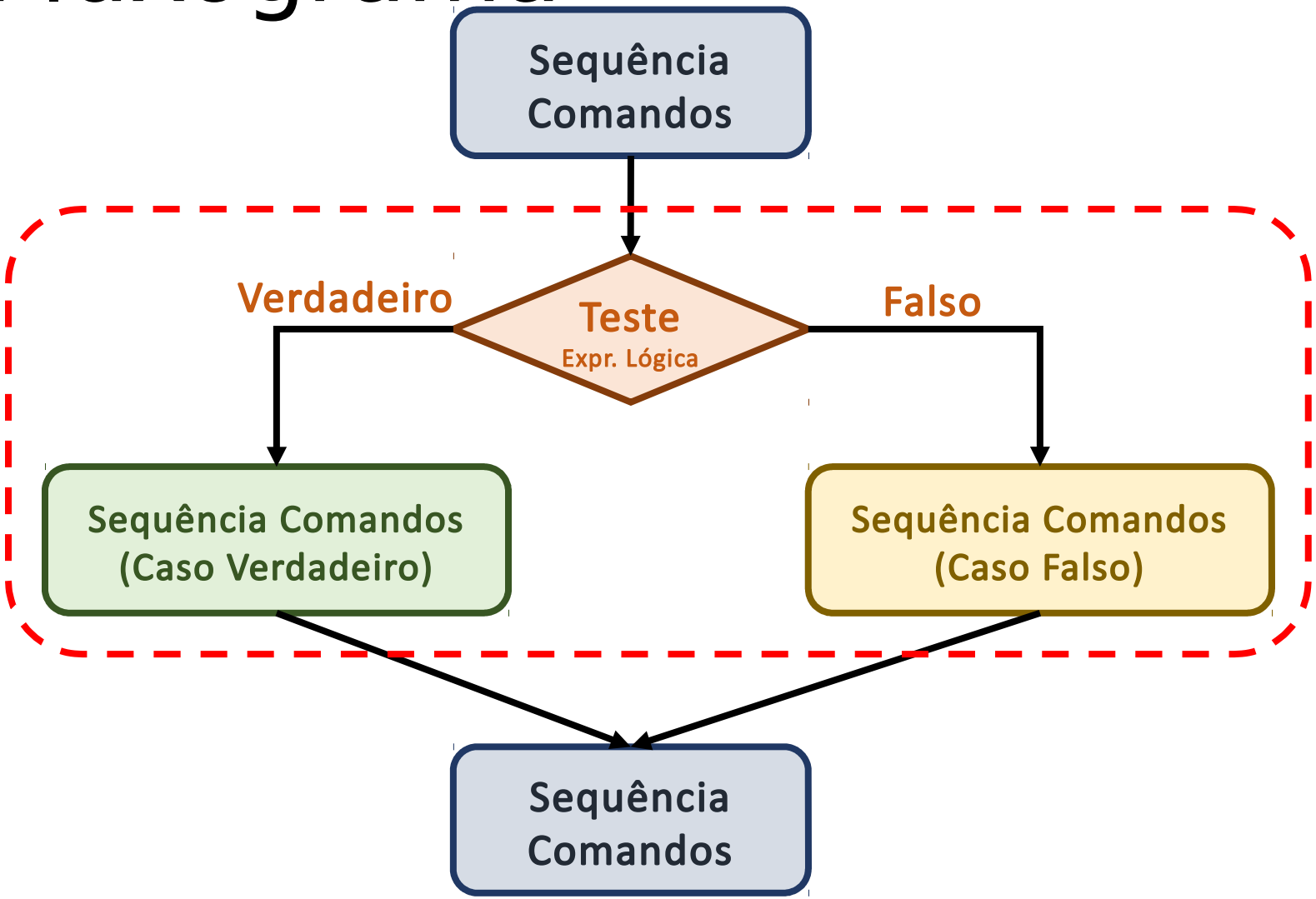
```
01 ano = int( input( "Ano: " ) )

02 bissexto = ano % 4 == 0 and \
          (ano % 100 != 0 or ano % 400 == 0)

03 print("É bissexto: ", bissexto)
```

Estruturas Condicionais

Estrutura Condicional

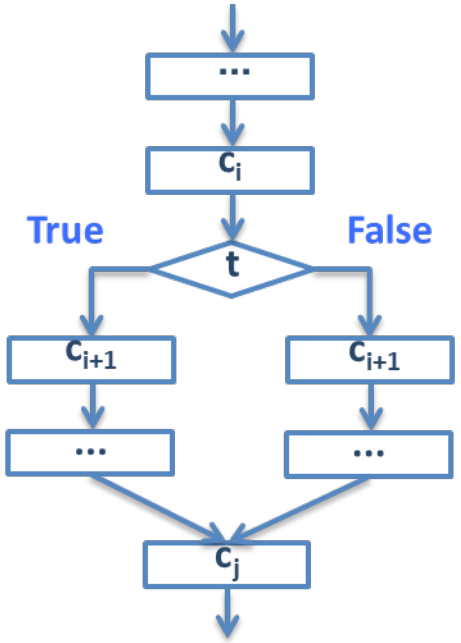
- As **estruturas condicionais** ou **comandos condicionais** permitem alterar a sequência de execução de um programa dependendo do resultado de uma **expressão lógica**.

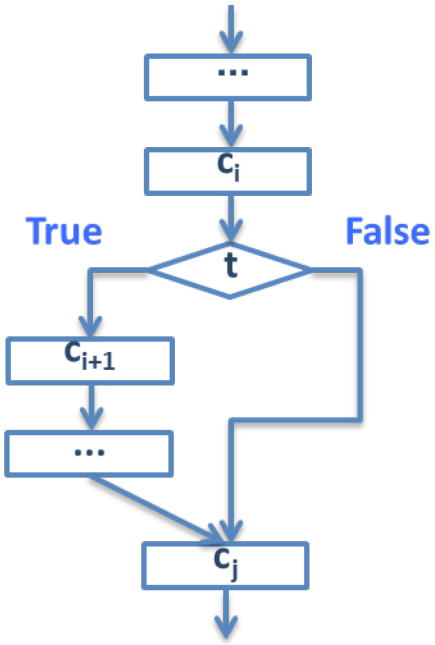
Estrutura Condicional: Fluxograma



Estrutura Condicional em Python

```
if expressão lógica:
    bloco de código
else:
    bloco de código
```






```
if expressão lógica:
    bloco de código
```

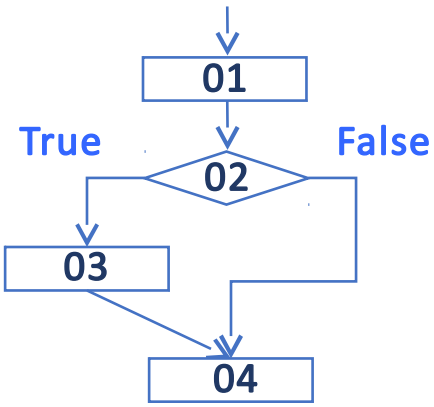
Estrutura Condicional em Python

```
if expressão lógica:
    bloco de código
elif expressão lógica:
    bloco de código
else:
    bloco de código
```

```
if expressão lógica:
    bloco de código
elif expressão lógica:
    bloco de código
```

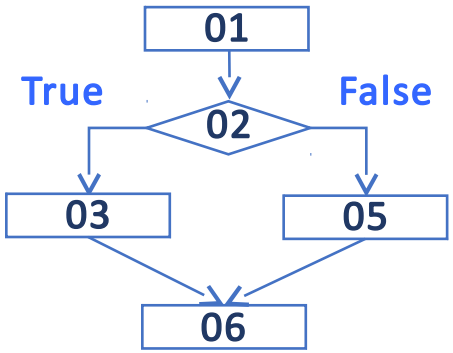
Estrutura Condicional Simples: `if`

```
01  ndvi = float( input("NDVI: ") )

02  if (ndvi > 0.3) and (ndvi < 0.8):
03      print("vegetação densa!")

04  print("NDVI: ", ndvi)
```



Estrutura Condicional Composta: if..else

```
01  ndvi = float( input("NDVI: ") )

02  if (ndvi > 0.3) and (ndvi < 0.8):
03      print("vegetação densa!")
04  else:
05      print("pouca vegetação!")

06  print("NDVI: ", ndvi)
```



Comandos Condicionais Encadeados
if..elif..else

```
01 ndvi = float( input("NDVI: ") )

02 if (ndvi < -1.0) or (ndvi > 1.0):
03     print("NDVI fora do intervalo!")
04 elif (ndvi > 0.3) and (ndvi < 0.8):
05     print("vegetação densa!")
06 else:
07     print("pouca vegetação!")

08 print("NDVI: ", ndvi)
```

Exemplo: Ano bissexto?

```
ano = int( input( "Ano: " ) )

if ano % 400 == 0:
    print("É bissexto!")
elif ano % 100 == 0:
    print("Não é bissexto!")
elif ano % 4 == 0:
    print("É bissexto!")
else:
    print("Não é bissexto!")

print("Fim!")
```

Estruturas Condicionais: Considerações

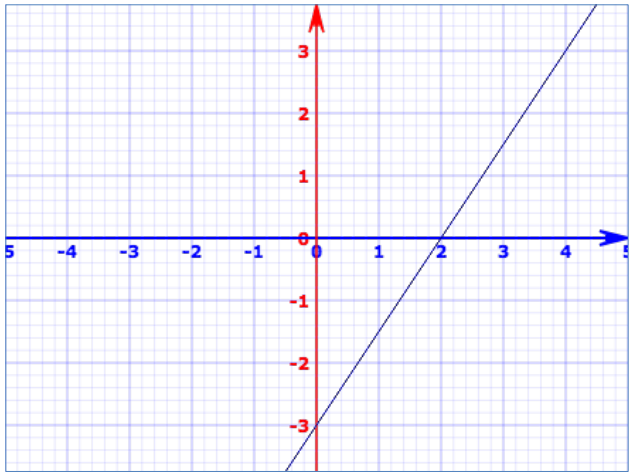
- As estruturas condicionais podem ser aninhadas, isto é, podem ser instruções dentro das cláusulas `if`, `else` e `elif`.

- A seção de código ou bloco de comandos dentro das cláusulas `if`, `else` e `elif` podem conter diversas instruções.

- Atente-se para a **identação** das instruções.

Hands-on

- Altere o programa da equação geral de uma reta para incluir um teste condicional que escreva na tela a mensagem *"Sobre a Reta"*, caso o ponto informado esteja sobre a reta; caso o ponto encontre-se acima da reta, deverá ser escrita a mensagem *"Acima da Reta"*; e, caso o ponto encontra-se abaixo da reta, deverá ser escrita a mensagem *"Abaixo da Reta"* .



$$3x-2y-6=0$$

Considerações Finais

Considerações Finais

- O tipo lógico e as expressões lógicas são muito utilizadas na construção de programas.
- Todos devem dominar a tabela verdade dos operadores `and`, `or` e `not`.
- As estruturas condicionais e de repetição são importantes para controlar o fluxo de execução de um programa.
- Nas próximas aulas iremos reforçar o uso das estruturas condicionais e de repetição.