Seleção condicional

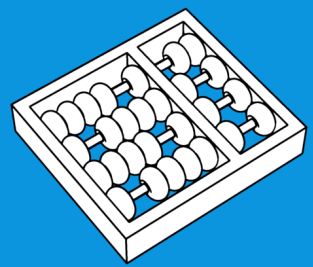
MC102 - Algoritmos e Programação de Computadores

03/24

4

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

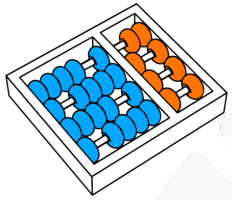
UNICAMP



"O interesse da matemática não é ser familiarizada com ideias de números e quantidades."

George Boole.

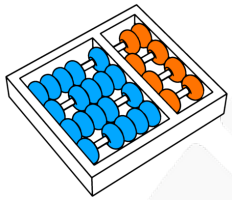
Dúvidas da aula anterior



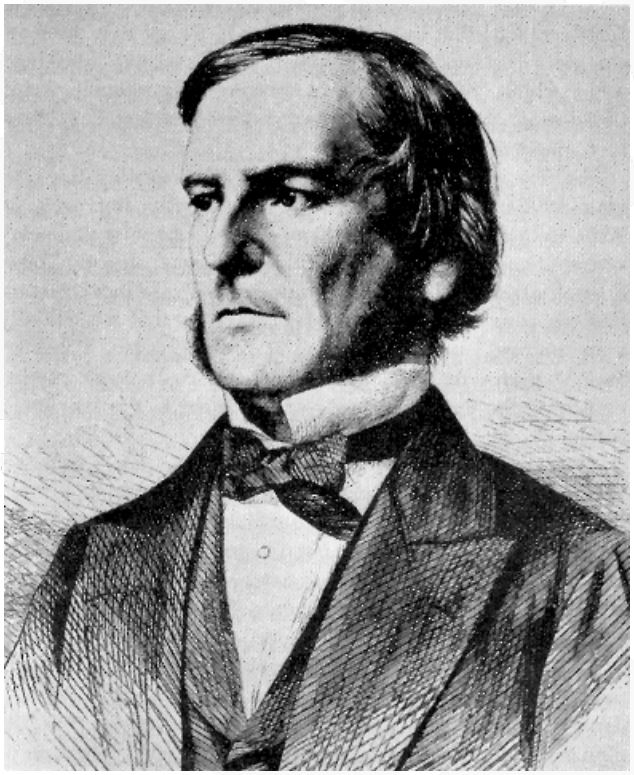
Dúvidas selecionadas

- É possível fazer um else que abrange todos os if's ou para cada um é necessário um próprio else?
- Quando usar ou não o else depois de um bloco if? Ou usar sempre?
- Por que não posso fazer mais uma comparação no if ao mesmo tempo, como if(a>x>y)?
- Por que se utiliza o == ao invés de = quando usamos o if?
- Por que python não tem função switch case? Era minha função favorita do c# fiquei indignado quando descobri q tinha que fazer um monte de elif seguido em python.
- Se eu converter o inteiro "10"para uma String e depois afirmar que é um Bool (por exemplo, "10"= True), quando eu utilizar o type(), ele continuará sendo uma string ou não?

Álgebra booleana

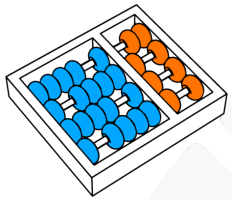


George Boole (02/09/1815 – 08/12/1864)



Principais áreas de atuação:

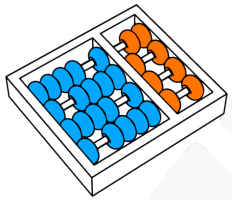
- Matemática.
- Filosofia.
- Lógica.



Álgebra booleana

Ramificação de álgebra em que:

- As variáveis tomam valores de **verdade**: **Verdadeiro** ou **Falso** (**1** ou **0**).
- Os operadores são lógicos, sendo os principais: conjunção (**e**) denotado por $\wedge$, disjunção (**ou**) denotado por $\vee$ e negação (**não**) denotado por $\neg$.



Operações lógicas

Operador	Uso	Significado
$\wedge$	$x \wedge y$	x e y são verdade?
$\vee$	$x \vee y$	x é verdade ou y é verdade?
$\neg$	$\neg x$	x é falso?

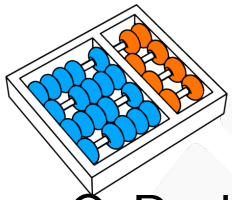
Tabelas de verdade:

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	1	0
	0	1

Expressões lógicas em Python



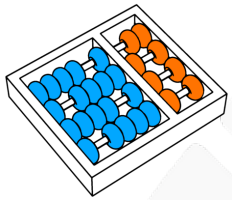
Operadores lógicos

O Python tem três operadores lógicos: **and**, **or** e **not**

a	b	a and b	a or b	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Observações:

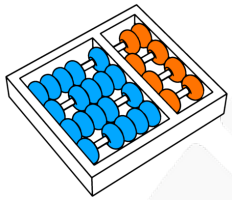
- **a** e **b** podem ser quaisquer expressões booleanas.
- Podemos escrever expressões do tipo **a and not b or c**.
- **not** precede **and** que precede **or**
- Mais fácil usar parênteses do que lembrar...
  - (**a and** (**not b**)) **or c**.



Precedência de operadores

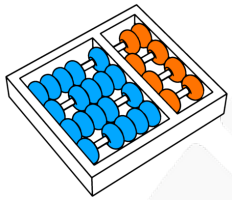
Ordem	Operadores
1º	**
2º	* / %
3º	+ -
4º	< > <= >= == !=
5º	not
6º	and
7º	or

Sugestão: agrupar sempre com parêntesis (ajuda na leitura)!!



Exemplo

```
1 n = int(input())
2 if n % 2 == 0 or n % 3 == 0:
3     print(n, "é divisível por 2 ou por 3")
4 else:
5     print(n, "não é divisível por 2 e", end=" ")
6     print("não é divisível por 3")
```



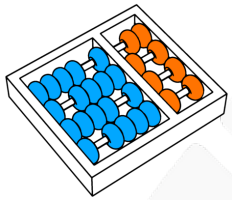
Três soluções de um problema

Determinar se n não é divisível por 3.

```
1  n = int(input())
2  if n % 3 != 0:
3      print(n, "não é divisível por 3")
4  else:
5      print(n, "é divisível por 3")
```

```
1  n = int(input())
2  if not n % 3 == 0:
3      print(n, "não é divisível por 3")
4  else:
5      print(n, "é divisível por 3")
```

```
1  n = int(input())
2  if n % 3 == 1 or n % 3 == 2:
3      print(n, "não é divisível por 3")
4  else:
5      print(n, "é divisível por 3")
```



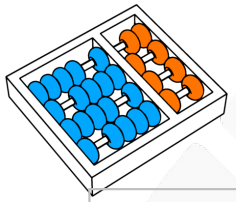
Regras de De Morgan

Uma regra útil da lógica:

- **not (a or b)** é equivalente a **(not a) and (not b)**.
- **not (a and b)** é equivalente a **(not a) or (not b)**.

Isso permite:

- Escrever expressões menores ou mais legíveis.
- Ou saber porque uma condição falhou.
  - Se o **if a and b** falhou é porque
  - **not a** ou **not b** é verdade.



Voltando a um exercício anterior

```
1 n = int(input())
2 if n % 2 == 0 and (not n % 3 == 0):
3     print(n, "é divisível por 2 e não por 3")
4 if not n % 2 == 0:
5     print(n, "não é divisível por 2")
6 if n % 3 == 0:
7     print(n, "é divisível por 3")
```

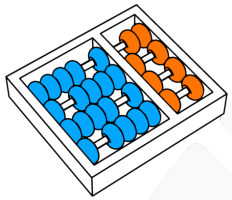
Usamos De Morgan para verificar porque não executa a linha 4.

Note que um **if** não precisa ter um **else**!

- Mas todo **else** precisa ter um **if**...

O que é impresso se **n** for **3**?

- Isso não acontecia no programa anterior...



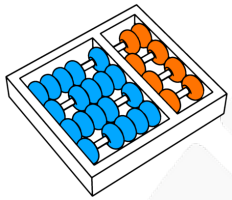
Uma versão melhor

```
1  n = int(input())
2  if n % 2 == 0 and (not n % 3 == 0):
3      print(n, "é divisível por 2 e não por 3")
4  elif n % 2 == 0:
5      print(n, "é divisível por 2 e por 3")
6  elif not n % 3 == 0:
7      print(n, "não é divisível por 2 e por 3")
8  else:
9      print(n, "não é divisível por 2 e é por 3")
10
```

O **elif** (de *else if*) testa uma nova condição:

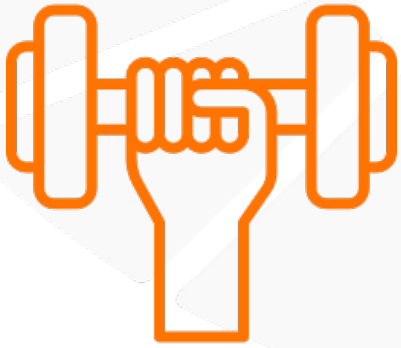
- Apenas se o **if** e os **elif**s anteriores falharam.

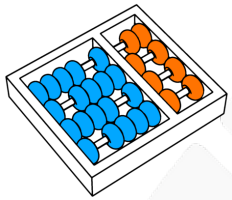
Exercício: Como fazer sem usar **elif**?



Seleção condicional

Vamos fazer alguns exercícios?





Exercício

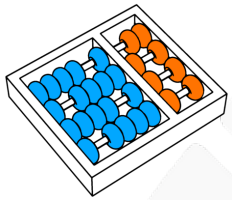
Vamos fazer um programa em Python que:

- Lê três números a, b e c.
- Calcula as raízes de $ax^2 + bx + c = 0$.
- Use a fórmula de Bhaskara.

$$\mathbf{x} = \frac{-\mathbf{b} \pm \sqrt{\mathbf{b^2} - \mathbf{4ac}}}{\mathbf{2a}}$$

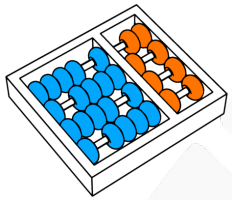
Cuidados:

- a não pode ser zero!
- $b^2 - 4ac$ não pode ser negativo!



Solução

```
1  # equação a x**2 + b x + c = 0
2  a = float(input())
3  b = float(input())
4  c = float(input())
5
6  delta = b**2 - 4 * a * c
7
8  if (a == 0):
9      # Expressão da forma b x + c = 0
10     print("Não é uma equação de segundo grau!")
11 elif (delta < 0):
12     print("As raízes são números complexos!")
13 else:
14     print("Raizes:")
15     print((- b - delta**(1 / 2)) / (2 * a))
16     print((- b + delta**(1 / 2)) / (2 * a))
```



Exercícios

1. Escreva um programa que lê dois números inteiros x e y e diz qual quadrante do espaço (x, y) está.
   - Use operadores lógicos dessa vez...

2. Escreva um programa que lê três números e encontra o maior dos três.

3. Dê um exemplo onde utilizar **if x ... else** é diferente de usar **if x** seguido de **if not x**.

Seleção condicional

MC102 - Algoritmos e Programação de Computadores

03/24

4

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP