Seleção condicional básica

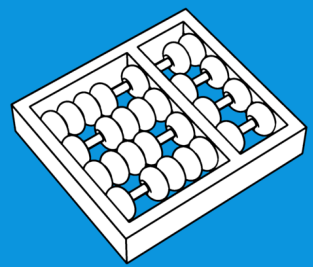
MC102 - Algoritmos e Programação de Computadores

03/24

3

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

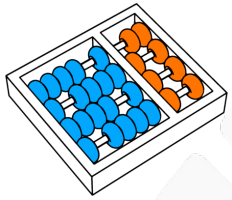
UNICAMP



“if...then...else
Precisavam ser inventado.”

Eric Fischer.

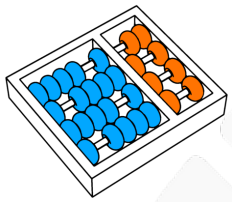
Dúvidas da aula anterior



Dúvidas selecionadas

- O que faz o python ficar atrás de outras linguagens como c++ no quesito velocidade de execução?
- Por quê operações entre inteiros e floats não causa problemas de aproximação? Como ocorreria entre dois floats.
- Falamos na aula sobre os possíveis erros ao operar com floats devido a leitura q o python faz deles. como o python os lê e qual a representação q a linguagem tem deles?
- Como isso funciona e qual a funcionalidade, sobre redirecionar entrada e saída?
- Dois inputs não conseguem ler se os números estiverem na mesma linha. Qual seria a explicação? E como posso ler número que estão na mesma linha?

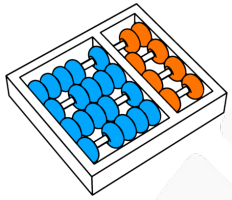
Verdadeiro ou falso



Testando o valor de uma variável

No terminal do Python:

```
>>> x = 2
>>> x == 2
True
>>> x == 3
False
>>> type(x == 2)
<class 'bool'>
>>> type(x == 3)
<class 'bool'>
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

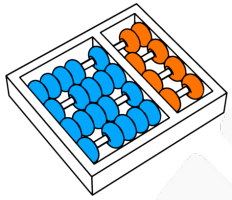


O tipo **bool**

O tipo **bool** define duas constantes:

- **True**.
- **False**.

E várias operações devolvem um **bool**.



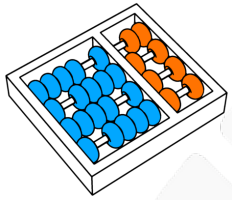
Testando se um número é par

Queremos definir se um número **n** dado é par ou ímpar:

- Isto é, $\mathbf{n = 2k}$ ou $\mathbf{n = 2k + 1}$ para **k** inteiro.

Problema (Par ou ímpar)

- ***Entrada:*** *Um número* **n**.
- ***Saída:*** *Um texto que informe se* **n** *é par ou ímpar.*



Pseudocódigo

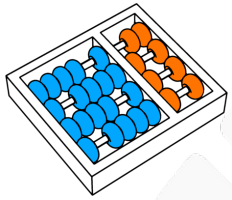
Antes do Python, vamos pensar abstratamente:

Algoritmo: PAR-OU-IMPAR

```
1 leia n
2 se n for par
3 |  escreva 'n é par'
4 senão
5 |  escreva 'n é ímpar'
```

Porém, precisamos ter cuidado para que:

- Cada passo seja suficientemente simples.
- E que possa ser executado pelo computador.



Pseudocódigo

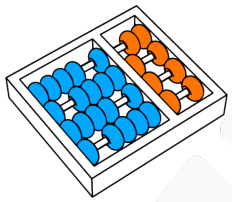
Qual é uma boa forma de testar se **n** é par ou não?

- Se **n** for par, então **n % 2** é **0**.
- Se **n** for ímpar, então **n % 2** é **1**.

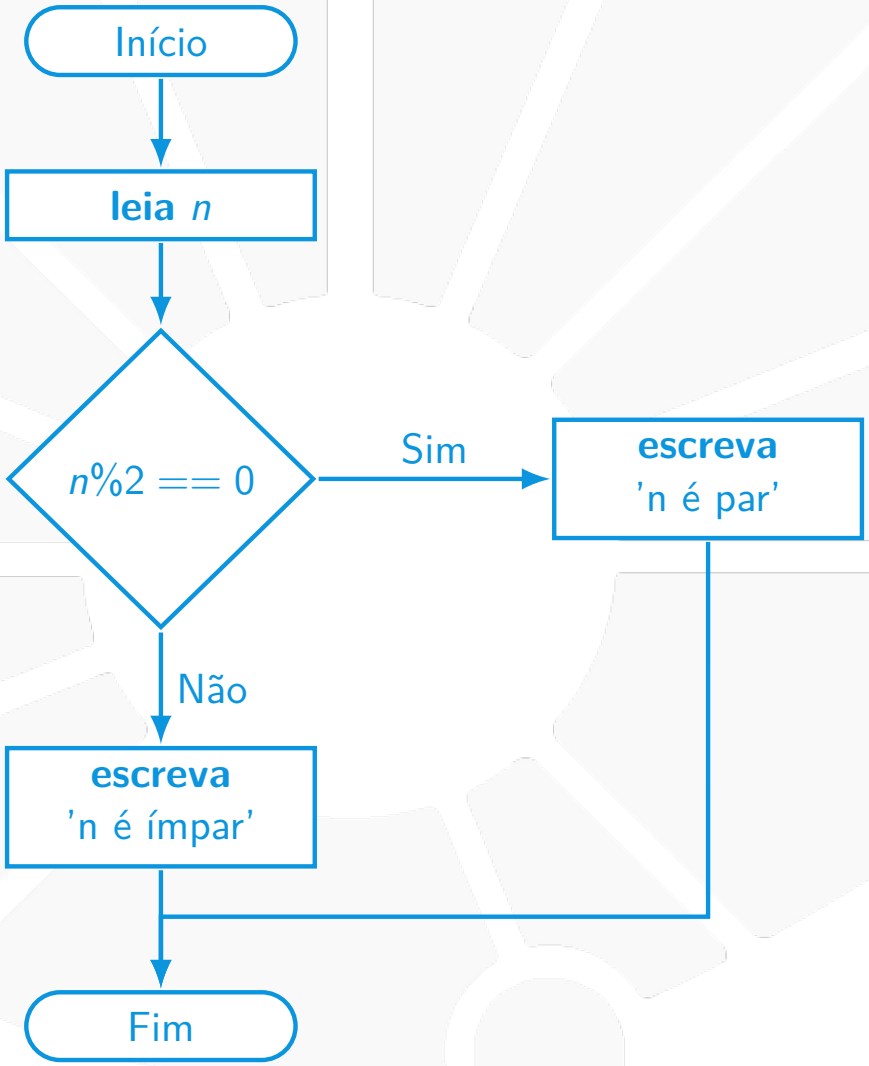
Um pseudocódigo mais claro seria:

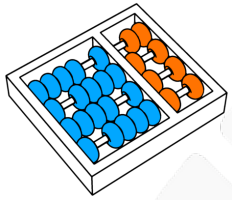
Algoritmo: PAR-OU-IMPAR

```
1 leia n
2 se n%2 = 0
3 |   escreva 'n é par'
4 senão
5 |   escreva 'n é ímpar'
```



Fluxograma

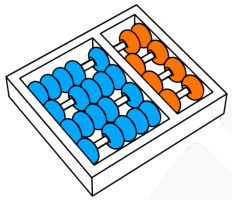




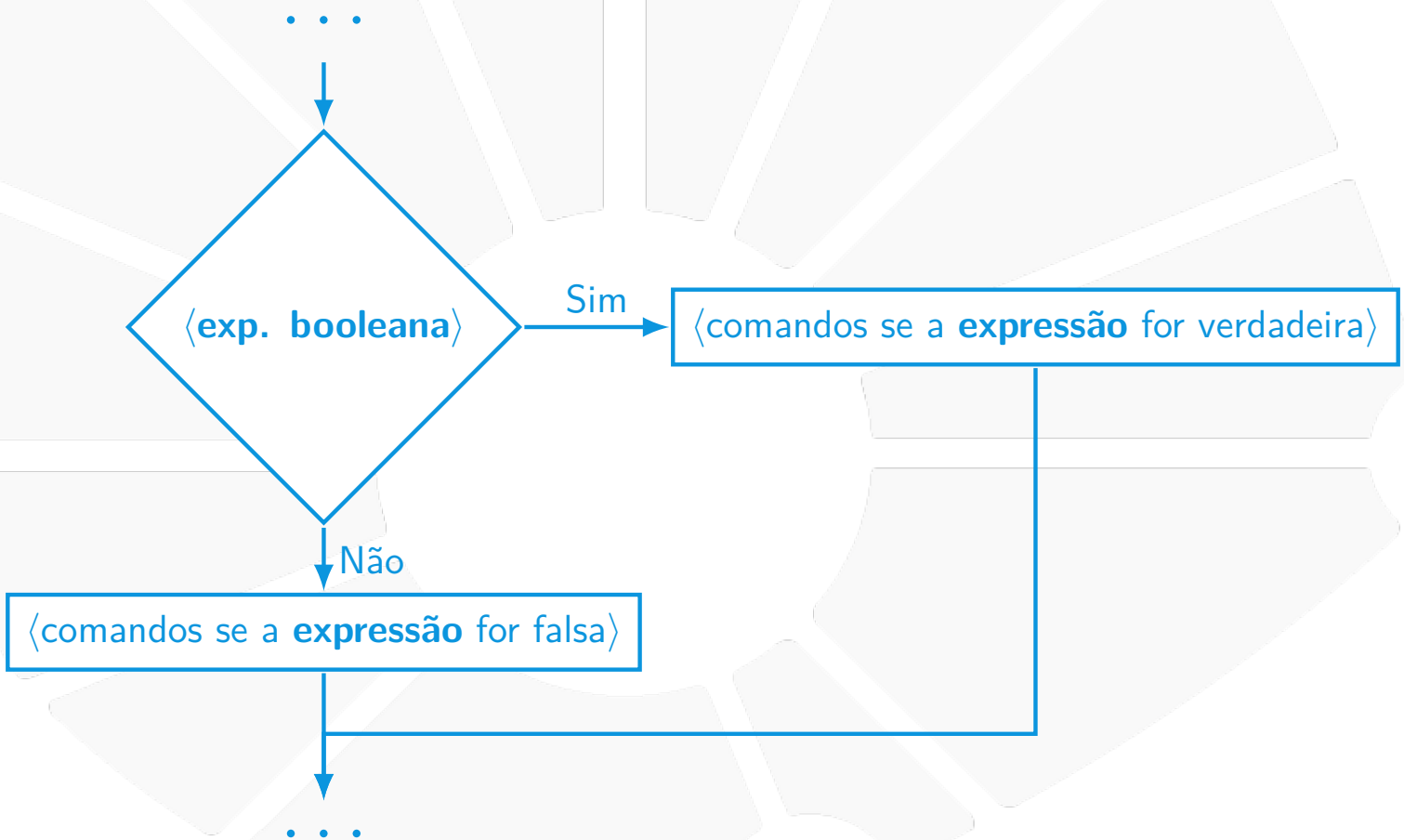
Pergunta

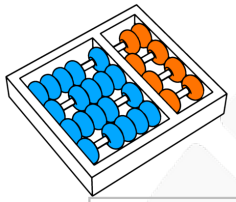
Como selecionar em Python?

Seleção condicional



Funcionamento





Em Python

```
1 ...
2 if <exp. booleana>:
3     <comandos se a expressão for verdadeira>
4 else:
5     <comandos se a expressão for falsa>
6 ...
```

O **if ... else**:

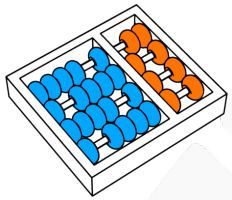
- Verifica o valor da expressão booleana.
- Se for **True**, executa o bloco de código do **if**.
- Se for **False**, executa o bloco de código do **else** (opcional).

O Python utiliza a indentação para criar **blocos de código**:

- Ela não é opcional como em outras linguagens.
- E precisa ser consistente (quatro espaços é o recomendado).

O **:** indica que a linha do **if**/**else** terminou:

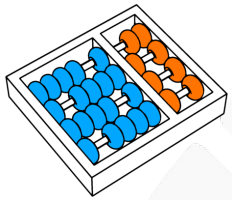
- Também indica que um bloco de código irá começar.
- Ele é usado em vários outros comandos.



Código em Python

Então, uma solução para determinar se um número é par ou ímpar seria:

```
1 n = int(input('Entre o número: '))
2 if n % 2 == 0:
3     print(n, 'é par')
4 else:
5     print(n, 'é ímpar')
```

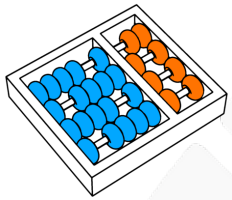


Outras comparações

Além de igualdade (==), podemos usar também:

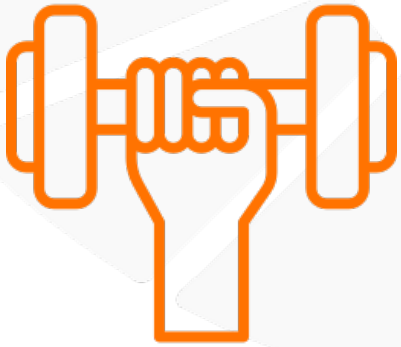
- `<` para saber se **a** $<$ **b**.
- `>` para saber se **a** $>$ **b**.
- `<=` para saber se **a** $\leq$ **b**.
- `>=` para saber se **a** $\geq$ **b**.
- `!=` para saber se **a** $\neq$ **b**.

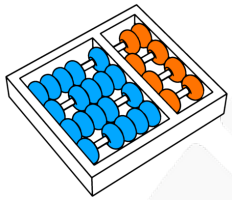
Juntamente com o ==, são chamados de **operadores de comparação**.



Seleção condicional

Vamos fazer alguns exercícios?



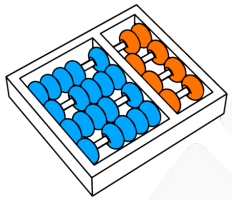


Exercício 1

Um inteiro **n** é divisível por um inteiro **q** se existe um inteiro **a** tal que $\mathbf{n} = \mathbf{aq}$.

- Isto é, se $\mathbf{n} \,\%\, \mathbf{q} = \mathbf{0}$.

Queremos verificar se **n** é divisível por **2** ou por **3**.



Uma solução

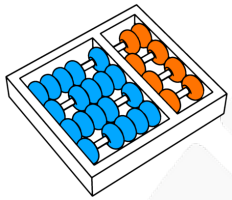
Queremos verificar se **n** é divisível por **2** ou por **3**:

- Podemos usar dois **if**s em sequência:

```
1  n = int(input())
2  # n ser divisível por 2 é o mesmo que o resto da divisão por 2 ser 0
3  if n % 2 == 0:
4      print(n, "é divisível por 2")
5  else:
6      print(n, "não é divisível por 2")
7  if n % 3 == 0:
8      print(n, "é divisível por 3")
9  else:
10     print(n, "não é divisível por 3")
```

A linha 2 é de comentários:

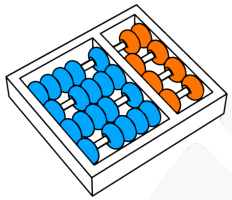
- Servem para entender melhor o código.
- São ignoradas pelo Python.
- Comentários **devem** ser usados, mas com **moderação**.



Exercício 2

Queremos verificar se **n**:

- É divisível por **2**.
- E não é divisível por **3**.

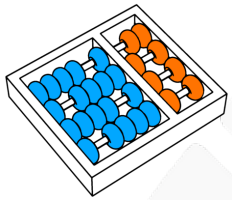


Uma solução

Queremos verificar se **n** é divisível por **2** e não é divisível por **3**.

Podemos usar um **if** dentro do outro!

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

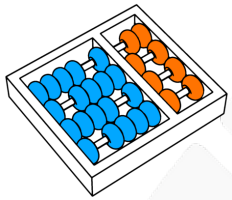


Exercícios 3, 4 e 5

1. Escreva um programa que lê dois números e encontra o maior dos dois.

2. Escreva um programa que lê dois números inteiros **x** e **y**, sendo que **y** tem apenas um dígito (na base 10) e verifica se **y** é o último dígito (na base 10) de **x**.

3. Escreva um programa que lê dois números inteiros **x** e **y** e diz em qual quadrante do plano o ponto **(x, y)** está.



Exercício 6

O tempo Unix nos diz quantos segundos se passaram desde a Época Unix (00:00 de 01 de Janeiro de 1970 — UTC).

Exemplo:

- Se o tempo Unix atual é 3600, então estamos em 01:00 de 01/01/1970 (UTC)
- Se o tempo Unix é 86400, então estamos em 00:00 de 02/01/1970 (UTC).

Escreva um programa que dado um tempo Unix diz qual é o dia da semana daquele tempo.

Seleção condicional básica

MC102 - Algoritmos e Programação de Computadores

03/24

3

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

UNICAMP