

# Comandos Condicionais

*MC102*

bool 🧛

## Verdadeiro ou Falso

O tipo `bool` define dois valores constantes:

- `True`
- `False`

E várias operações devolvem um `bool`.

</> Code

Dado um número  $n$ , defina se é par ou ímpar.

# Pseudocódigo

## ! Warning

Antes do Python, vamos pensar abstratamente:

- O que precisa ser feito?

```
1 Leia n
2 Se n for par:
3     Imprima "par"
4 Senão:
5     Imprima "ímpar"
```

# Pseudocódigo

Temos que ter cuidado para que:

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

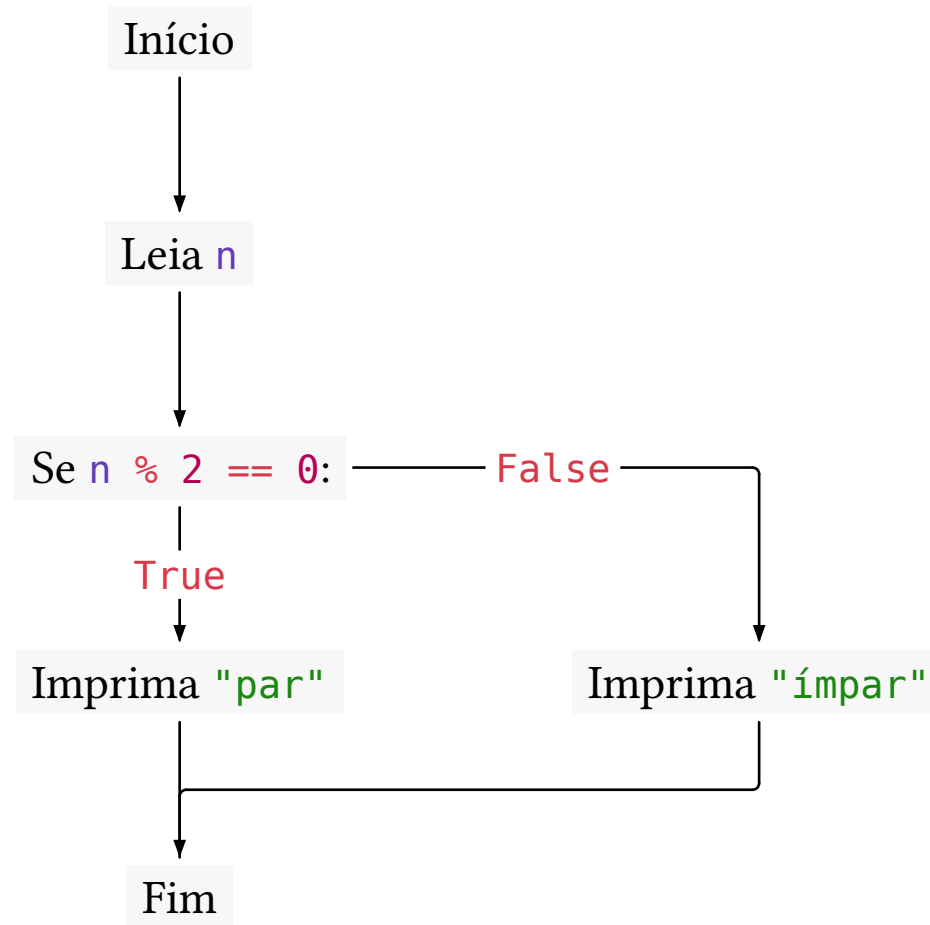
Sabemos que, se o resto da divisão de  $n$  por 2 for zero, ele é par. Caso contrário, é ímpar.

Lembre-se da aula passada, quando vimos o operador de resto `%`.

```
1 Leia n
2 Se n % 2 == 0:
3     Imprima "par"
4 Senão:
5     Imprima "ímpar"
```

Em Python, o operador de “igualdade” é o `==`.

# Fluxograma



**if ... else**

```
if ... else
```

## Em Python

```
1 n = int(input())
2
3 if n % 2 == 0:
4     print("par")
5 else:
6     print("impar")
```

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

- Não é opcional como em outras linguagens
- Precisa ser consistente: recomendo quatro **espaços**

# Em Python

## ⚡ Danger

```
1 if a > 2:  
2 print(a)
```

```
IndentationError: expected an indented  
block after 'if' statement
```

O `:` indica o final da linha do `if/else`:

- E que um bloco de código irá começar
- É utilizado em diversos outros comandos também

## ⚡ Danger

```
1 if a > 2  
2 print(a)
```

```
SyntaxError: expected ':'
```

if ... else

# Em Python

</> Code

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

```
1 a = int(input())
2 b = int(input())
3
4 if a > b:
5     print(a)
6 else:
7     print(b)
```

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`

if ... else

## Comparações

Além da 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 `==`, estes são os chamados **operadores de comparação**. Todos eles resultam em um valor `bool`: `True` ou `False`

if ... else

## Exercício

</> Code

Dado um inteiro  $n$ , verifique se  $n$  é divisível por 2 ou 3.

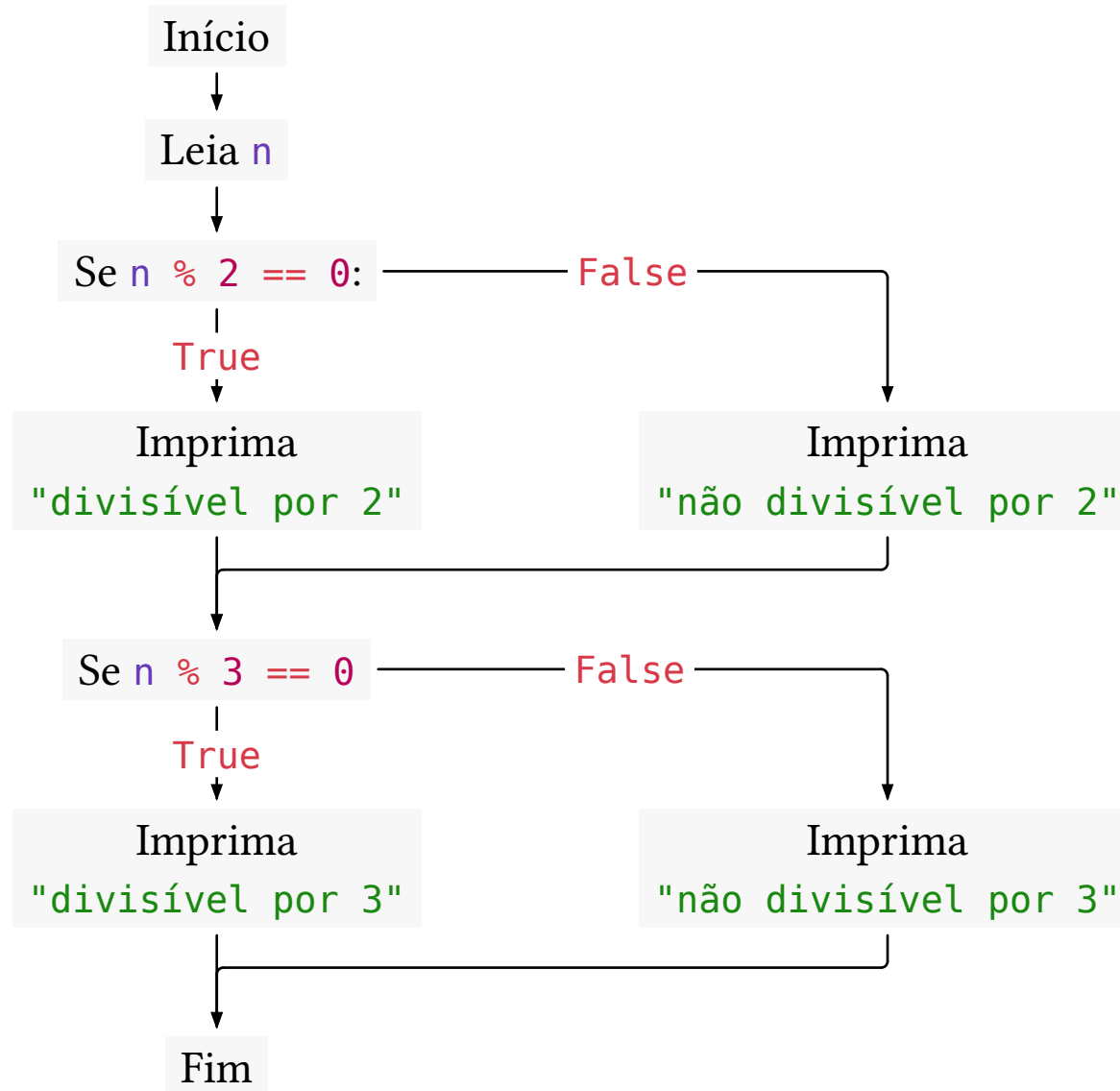
Um número  $a$  é divisível por  $b$  se  $a$  resto  $b = 0$ .

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

if ... else

# Exercício

if ... else



if ... else

## Omitir o `else`

É possível não colocar `else` para um `if`.

Perceba que isso é a mesma coisa que dizer: “Caso contrário, não faça nada.”

Dito isso, para utilizar o `else` você **precisa** ter um `if` (e cada `if` possui, no máximo, um `else`).

### Danger

```
1 if a < 2:  
2     print("Pequeno!")  
3 else:  
4     print("ok...")  
5 else:  
6     print(a)
```

```
1 a = 2  
2 else:  
3     print(a)
```

```
SyntaxError: invalid  
syntax
```

**Dúvidas?**

## Exercício

</> Code

Dado um inteiro  $n$ , determine se ele é divisível por 2 e não é divisível por 3.

# **Operadores Lógicos**

## and, or e not

2 verdades, 1 mentira!

```
1 a = "Eu tenho dois pinos em cada joelho"  
2 b = "Eu nunca virei uma noite"  
3 c = "Eu sou alérgico a kiwi"
```

## and, or e not

2 verdades, 1 mentira!

```
1 a = "Eu tenho dois pinos em cada joelho" # true!  
2 b = "Eu nunca virei uma noite"          # true!  
3 c = "Eu sou alérgico a kiwi"            # false...
```

Tá, mas agora, é verdade que...

```
1 d = not a    # Eu não tenho dois pinos em cada joelho  
2 e = b and c # Eu nunca virei a noite e sou alérgico a kiwi  
3 f = a or c   # Eu tenho dois pinos em cada joelho ou sou alérgico a kiwi
```

E se...

```
g = a or b # Eu tenho dois pinos em cada joelho ou nunca virei a noite
```

**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

- **a** e **b** podem ser quaisquer expressões booleanas, ou seja, que resultam em **True** ou **False**.
- Podemos escrever expressões do tipo **not a and b or c**.
- Existe uma precedência entre esses operadores...
  - ▶ Mas é melhor não se preocupar com isso e só usar parêntesis: **(not a) and (b or c)**

## and, or e not

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

### Danger

Eu já vi muito aluno fazendo isso:

```
if n % 2 or n % 3 == 0:
```

```
if a or b < 3:
```

Infelizmente, isso não é um erro de sintaxe em Python, mas também não faz o que queremos!