

# Tipos básicos

**Programação I**  
**2017.2018**

*Teresa Gonçalves*  
[tcg@uevora.pt](mailto:tcg@uevora.pt)

Departamento de Informática, ECT-UÉ

# Sumário

**Revisão**

**Tipos numéricos**

**Tipo Booleano**

**String**

**Conversão de tipos**

# Revisão

# Valores e variáveis

## Valor

Elemento básico

## Variável

Nome que representa um valor

## Atribuição

Instrução que associa um valor à variável

Ex:  $x = a$

## Valores e variáveis têm um tipo

Função `type()`

`int`, `float`, `str`

# Instruções, expressões e operações

## **Instrução**

Unidade de código a ser executada

## **Expressão**

Conjunto de operações sobre valores e variáveis

## **Operação**

Cálculo indicado através de operadores e operandos

# Operadores, operandos e precedência

## Operador

Símbolo que representa um cálculo

Aritméticos: + - \* / % // \*\*

Relacionais: == != < <= > >=

## Operando

Argumentos dos operadores

Podem ser valores ou variáveis

## Precedência

Regras que definem a ordem de avaliação das expressões

() > \*\* > \* / > + -

# Comentários

## Comentário

Anotação, em língua natural, para ajudar a entender o código fonte (programa)

Até final linha: #

Multi-linha: """ ..... """

# Tipos numéricos



# Inteiros

## int

Inteiro que pode ser positivo ou negativo

Não tem limite máximo

Python 2: representado com 32 bits (ou 64 nas arquiteturas amd64)

Quantos valores se consegue representar com 32 bits?

Em inteiros com sinal, qual o maior e o menor números representáveis?

# Reais

## float

Valores numéricos com parte decimal

A melhor aproximação aos números reais

## Sintaxe

Simples: 0.000578

Forma científica:  $5.78e-4 \rightarrow 5.78 * 10^{-4}$

Outra:  $2^{*-4} \rightarrow 2^{-4}$

# Complexos

## complex

Para representar números complexos (parte real e parte imaginária)

```
>>> z = complex(2, 3)
```

```
>>> z
```

```
(2+3j)
```

```
>>> z.real
```

```
2.0
```

```
>>> z.imag
```

```
3.0
```

# Tipos booleanos

# Booleano

**Representam um valor verdade**

**Constantes**

True, False

**Operadores lógicos (booleanos)**

x **or** y

Se x == False então y, senão x

x **and** y

Se x == False então x, senão y

**not** x

Se x == False então True, senão False

# Operadores booleanos

**São aceites outros tipos**

**Expressões com valor **False****

0 (em qualquer tipo numérico)

Sequência vazia

**Expressões com valor **True****

Restantes

# Operadores booleanos

## Exemplos

```
>>> not(10)
```

```
False
```

```
>>> not(0)
```

```
True
```

```
>>> x=1
```

```
>>> b=True
```

```
>>> b and x
```

```
1
```

# Operadores relacionais

## Dão origem a um valor booleano

$x == y \rightarrow \text{True}$  se  $x$  é igual a  $y$

Não confundir com a atribuição!

$x != y \rightarrow \text{True}$  se  $x$  é diferente de  $y$

$x < y \rightarrow \text{True}$  se  $x$  é menor que  $y$

$x > y \rightarrow \text{True}$  se  $x$  é maior que  $y$

$x <= y \rightarrow \text{True}$  se  $x$  é menor ou igual a  $y$

$x >= y \rightarrow \text{True}$  se  $x$  é maior ou igual a  $y$



# Precedência

**not > and > or**

```
>>> False or not False and True  
True
```

## Precedência menor que operadores não booleanos

```
not 0+1 == not (0+1)
```

## Outros exemplos

```
>>> 1 == 2
```

```
False
```

```
>>> 1==1 and True
```

```
True
```

# Utilização

## Comparação ou teste

Igualdade

`idade == 18`

Desigualdade

`altura > 1.80`

## Expressar conjunto de características

Conjunções, disjunções e outras combinações

Inteiro par e menor que 10

`type(x) == int and x<10 and x%2==0`

# String

## Cadeia de caracteres

### Operações elementares

+ (concatenação)

```
>>> print ('Hello ' + 'World' + '!')  
'Hello World!'
```

*Que propriedade da soma de números não é válida na concatenação?*

\* (repetição)

```
>>> 'Spam ' * 3  
'Spam Spam Spam'
```

# Conversão de tipos

# Conversão explícita

**float( expr )**

Converte expr para um valor em vírgula flutuante

**int( expr )**

Converte expr para um valor inteiro

**str( expr )**

Devolve uma cadeia de carateres, resultado da expressão expr

```
>>> str( 1+2/3 )  
'1.6666666666666665'
```

**eval( string )**

Avalia string como expressão

```
>>> eval ( '1+2/3' )  
1.6666666666666665
```

# Conversão implícita

## Conversão automática

Para permitir aritmética entre operandos de tipo diferente

Conversão para o tipo “mais abrangente”

int → float

float → complex