



### Fundamentos de Programação

António J. R. Neves João Rodrigues

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

an@ua.pt / jmr@ua.pt
http://elearning.ua.pt/

Fundamentos de Programação 2017/2018

### Boolean expressions



• A **boolean expression** is an expression that is either true or false.

```
>>> n = 5  # this IS NOT a boolean expression!  
>>> n == 5  # this IS a boolean expression!  
True  
>>> 6 == n  # this is another boolean expression.  
False
```

- True and False are special values that belong to the type bool.
- · Boolean values may be stored in variables.

```
>>> isEven = n%2==0
```

· May be converted to string.

```
>>> str(isEven)
'False'
```

· Or to integer.

```
>>> int(False) # 0
>>> int(True) # 1
```

Zero and empty values convert to False: >>> bool(0) # False

Fundamentos de Programação 2017/2018

## Summary



- Boolean expressions
  - · The bool type
  - · Relational operators
  - · Logical operators
  - Properties
- · Conditional execution
  - · If statement
  - If-else
  - · If-elif-else

Fundamentos de Programação 2017/2018

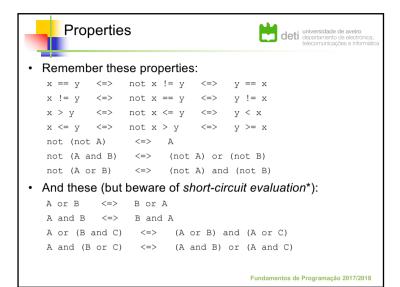
## Relational and logical operato褚 <sub>deti</sub>

eti universidade de aveiro departamento de electrónica,

Relational operators produce boolean results:

• There are three logical operators: and, or, not.

Fundamentos de Programação 2017/2018



## Short-circ

### Short-circuit evaluation



 Operators and and or only evaluate the second operand if needed!

```
X and Y # if X is false then X, otherwise Y X or Y # if X is true then X, otherwise Y
```

- This is called short-circuit evaluation.
- It can be very useful:

```
1/n>2 and n!=0 # ZeroDivisionError if n==0

n!=0 and 1/n>2 # False if n==0, 1/n not evaluated

n==0 or 3/n<4 # True if n==0, 3/n not evaluated
```

- · But remember:
  - · Commutative and distributive properties may not be valid!

Fundamentos de Programação 2017/2018

## Precedence rules



• Arithmetic > relational > not > and > or.

```
\begin{array}{c} x <= 1 + 2 * y * * 3 \text{ or } n! = 0 \text{ and not } 1/n <= y \\ (x <= 1 + 2 * y * * 3) \text{ or } (\underline{n! = 0} \text{ and not } 1/n <= \underline{y}) \\ (x <= (\underline{1 + 2 * y * * 3})) \text{ or } ((\underline{n! = 0}) \text{ and } (\underline{not } 1/n <= \underline{y})) \\ (x <= (1 + (\underline{2 * y * * 3}))) \text{ or } ((\underline{n! = 0}) \text{ and } (\underline{not } (\underline{1/n} <= \underline{y}))) \\ (x <= (1 + (2 * (\underline{y * * 3})))) \text{ or } ((\underline{n! = 0}) \text{ and } (\underline{not } ((\underline{1/n}) <= \underline{y}))) \end{array}
```

Fundamentos de Programação 2017/2018

# С

### Conditional execution (1)



- The ability to check conditions and change the behavior of the program accordingly is almost always used.
   Conditional statements give us this ability.
- The simplest form is the if statement:

```
if x > 0:
   print('x is positive')
```

- The boolean expression after if is called the condition.
- The indented statement(s) gets executed if the condition is true. If not, nothing happens.
- There is no limit on the number of statements that can appear in the body, but there has to be at least one.

Fundamentos de Programação 2017/2018



### Conditional execution (2)



 A second form of the if statement is alternative execution, in which there are two possibilities and the condition determines which one gets executed.

```
if x%2 == 0:
  print('x is even')
else:
  print('x is odd')
```

 Sometimes there are more than two possibilities and we need more than two branches (chained conditional).

```
if x < y:
    print('x is less than y')
elif x > y:
    print('x is greater than y')
else:
    print('x and y are equal')
```

Fundamentos de Programação 2017/2018

## Code transformations



· Transformations may simplify the code.

```
if Cond1:
                if not Cond1:
                                if not Cond1:
 if Cond2:
                 Suite3
                                 Suite3
   Suite1
                else:
                                elif Cond2:
                 if Cond2:
                                 Suite1
 else:
   Suite2
                  Suite1
                                else:
                 else:
                                 Suite2
else:
 Suite3
                   Suite2
```

Fundamentos de Programação 2017/2018

## 4

### Conditional execution (3)



One conditional can also be nested within another.

```
if x == y:
   print('x and y are equal')
else:
   if x < y:
     print('x is less than y')
   else:
     print('x is greater than y')</pre>
```

- Although the indentation of the statements makes the structure apparent, nested conditionals become difficult to read very quickly.
- Logical operators often provide a way to simplify nested conditional statements.

Fundamentos de Programação 2017/2018