

# Fundamentos da Programação @ LEIC/LETI

## Semana 01 - Aula 03

### Elementos básicos de programação

*Expressões. Tipos elementares de informação. Nomes e atribuição. Predicados e condições. Comunicação com o exterior.*

Alberto Abad, Tagus Park, IST, 2021

## Interpretador de Python

### Modo interativo (read-eval-print loop)

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> print("Hello world!")
Hello world!
>>>
```

- O símbolo `>>>` indica que podemos introduzir o próximo comando, em **BNF**:

`<comando> ::= <expressão> | <instrução> | <definição>`

## Elementos básicos de programação - Expressões

### BNF:

`<expressão> ::= <constante> | <expressão composta> | <nome> | <chamada função>`

## Elementos básicos de programação - Expressões constantes

### Números inteiros e reais, valores lógicos e cadeias de caracteres (strings)

- 2016
- 2
- 2.0
- -2.0
  
- +2.0
- 10e-12
- 6376754588877162243232221200091999228887333
- 0.000000000000000000000000000000000001
  
- *True*
- *False* (#atenção *true* e *false* !)
  
- 'Hello world'
- "Hello"
- "Os strings são sequencias"

In [ ]:

## Elementos básicos de programação - Expressões compostas

### BNF:

```
<composed expression> ::=  
    <operator> <expression> |  
    <operator> (<expression>) |  
    <expression> <operator> <expression> |  
    (<expression> <operator> <expression>)
```

- **Operadores built-in:** not, -(simétrico), \*, /, //, %, +, -(subtração), <, >, ==, >=, <=, !=, and, or, etc.

### Exemplos

- -5
- -(5)
- not False
- 2012 - 1958
- 3 \* 24 + 12
- 3 \* (24 + 12)
- 3.0 \* (24 + 12)
- 7 > 12
- 23 / 7 \* 5 + 12.5
- 7 // 2

# Elementos básicos de programação - Expressões compostas

## Exemplos

- -5
- -(5)
- not False
- 2012 - 1958
- $3 * 24 + 12$
- $3 * (24 + 12)$
- $3.0 * (24 + 12)$
- $7 > 12$
- $23 / 7 * 5 + 12.5$

In [139]:

Out[139]: 6

## Expressões compostas: Prioridade dos operandos (1)

### Regra #1 (De maior a menor prioridade)

| Operator   | Description                         |
|--|-------------------------------------|
| ()   | Parentheses (grouping)              |
| $f(\text{args}..)$                               | Function call                       |
| $x[\text{index}:\text{index}]$                   | Slicing                             |
| $x[\text{index}]$                                | Subscription                        |
| $x.\text{attribute}$                             | Attribute reference                 |
| **   | Exponentiation                      |
| ~x   | Bitwise not                         |
| +x, -x   | Positive, negative                  |
| *, /, %  | Multiplication, division, remainder |
| +, -   | Addition, subtraction               |
| <<, >>   | Bitwise shifts                      |
| &  | Bitwise AND                         |
| ^  | Bitwise XOR                         |
|  | Bitwise OR                          |
| in, not in, is, is not, <, <=, >, >=, <>, !=, == | Comparisons, membership, identity   |
| not x  | Boolean NOT                         |
| and  | Boolean AND                         |
| or   | Boolean OR                          |
| lambda   | Lambda expression                   |

# Expressões compostas: Prioridade dos operandos (2)

## Regra #2

- Em caso de igualdade, de esquerda para direita

Official info: <https://docs.python.org/3/reference/expressions.html#operator-precedence>  
(<https://docs.python.org/3/reference/expressions.html#operator-precedence>)

## Elementos básicos de programação - Tipos elementares

- **Tipos:** Conjuntos de entidades (valores) + operações
- Tipos elementares vs tipos estruturados
- 3 tipos elementares em Python:
  - tipo inteiro, `int`
  - tipo real, `float`
  - tipo lógico, `bool`
- Usar `type(value)` ou `isinstance(value, type)` para verificar o tipo duma expressão.
- Tipos não elementares (strings, tuplos, listas, dicionarios, etc.) nas próximas semanas

In [148]:

Out[148]: `bool`

# Elementos básicos de programação - Tipos elementares

## O tipo inteiro ( `int` )

| <i>Operação</i> | <i>Tipo dos argumentos</i> | <i>Valor</i>  |
|-----------------|----------------------------|---|
| $e_1 + e_2$     | Inteiros                   | O resultado de somar $e_1$ com $e_2$ .              |
| $e_1 - e_2$     | Inteiros                   | O resultado de subtrair $e_2$ a $e_1$ .             |
| $-e$            | Inteiro                    | O simétrico de $e$ .                                |
| $e_1 * e_2$     | Inteiros                   | O resultado de multiplicar $e_1$ por $e_2$ .        |
| $e_1 // e_2$    | Inteiros                   | O resultado da divisão inteira de $e_1$ por $e_2$ . |
| $e_1 \% e_2$    | Inteiros                   | O resto da divisão inteira de $e_1$ por $e_2$ .     |
| $\text{abs}(e)$ | Inteiro                    | O valor absoluto de $e$ .                           |

## Exemplos

- -12
- `7 // 2`
- `2 + 7*5`
  
- `7 % 2`
- `5 * (7 // 2)`
- `abs(-3)`

In [152]: `2 + 7//5`

Out[152]: 3

# Elementos básicos de programação- Tipos elementares

## O tipo real ( `float` )

| <i>Operação</i> | <i>Tipo dos argumentos</i> | <i>Valor</i>                                 |
|-----------------|----------------------------|--|
| $e_1 + e_2$     | Reais                      | O resultado de somar $e_1$ com $e_2$ .       |
| $e_1 - e_2$     | Reais                      | O resultado de subtrair $e_2$ a $e_1$ .      |
| $-e$            | Real                       | O simétrico de $e$ .                         |
| $e_1 * e_2$     | Reais                      | O resultado de multiplicar $e_1$ por $e_2$ . |
| $e_1 / e_2$     | Reais                      | O resultado de dividir $e_1$ por $e_2$ .     |
| $\text{abs}(e)$ | Real                       | O valor absoluto de $e$ .                    |

- Notação decimal e notação científica
- Atenção sobrecarga (*overloading*) operadores!
- Atenção conversão de tipos implícito (*coercion*)

In [154]: `15e-1`

Out[154]: 1.5



# Elementos básicos de programação - Tipos elementares

## O tipo lógico ( bool )

| $e_1$ | $e_2$ | $e_1$ and $e_2$ | $e_1$ or $e_2$ |
|-------|-------|-----------------|----------------|
| True  | True  | True            | True           |
| True  | False | False           | True           |
| False | True  | False           | True           |
| False | False | False           | False          |

## Exemplos

- True
- False
- not True
- not False
  
- not 5
- False and a (*short-circuit*)
- True or b
- not '' (*equivalent: not 0*)

In [ ]:

# Elementos básicos de programação - Nomes e atribuição

- Nome: identificar entidade computacional
- Atribuição: associar um nome a um valor/entidade

## BNF

$\langle \text{nome} \rangle ::= \langle \text{nome simples} \rangle \mid \langle \text{nome indexado} \rangle \mid \langle \text{nome composto} \rangle$

- Por enquanto veremos o  $\langle \text{nome simples} \rangle$  ...



# Elementos básicos de programação - Nomes

## Nome simples BNF

`<nome simples> ::= <inicial> <subsequente>*`

`<inicial> ::=`

```
_ | A | B | C | D | E | F | G | H | I | J | K | L | M |  
N | O | P | Q | R | S | T | U | V | W | X | Y | Z |  
a | b | c | d | e | f | g | h | i | j | k | l | m |  
n | o | p | q | r | s | t | u | v | w | x | y | z
```

`<subsequente> ::= <inicial> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

## Exemplos

- xpto, XPTO, Xpto, Taxa\_de\_Juro, \_largura,
- turma FP, duvida?, ola!
- ...

# Elementos básicos de programação - Nomes

## Nomes reservados (Keywords)

|        |          |         |          |        |
|--------|----------|---------|----------|--------|
| False  | class    | finally | is       | return |
| None   | continue | for     | lambda   | try    |
| True   | def      | from    | nonlocal | while  |
| and    | del      | global  | not      | with   |
| as     | elif     | if      | or       | yield  |
| assert | else     | import  | pass     |        |
| break  | except   | in      | raise    |        |

In [ ]:

# Elementos básicos de programação - Atribuição

## Atribuição simples e múltipla

```
<attribution instruction> ::= <name> = <expression> |  
    <name> , <attribution instruction> , <expression>
```

- A atribuição é uma INSTRUÇÃO:
  - As instruções são executadas (têm um *efeito*)
  - As expressões são avaliadas (têm um *valor*)
- Ordem: Primeiro avaliação da expressão, logo atribuição
- Ambiente ou espaço de nomes (*namespace*)

# Elementos básicos de programação - Atribuição

## Atribuição simples e múltipla

### Exemplos

- not = 9
- NOT = 9
- x
- x = 8
  
- x
- y
- y = x \* 2
- x = 7
  
- x , z = 10 , 3
- x + z
- x , z = z , x
- x
  
- z
- z , a = a + 3 , 1

In [ ]:

## Elementos básicos de programação - Predicados e condições

- Um *predicado* é uma operação cujo valor é lógico: *True* or *False*
- Uma *condição* é uma expressão cujo valor é lógico
- As condições podem ser combinadas com os operadores lógicos, ex: `and` , `or`
- Operadores relacionais em Python:

| Operação     | Tipo dos argumentos | Valor  |
|--------------|---------------------|--|
| $e_1 == e_2$ | Números             | Tem o valor <code>True</code> se e só se os valores das expressões $e_1$ e $e_2$ são iguais.                       |
| $e_1 != e_2$ | Números             | Tem o valor <code>True</code> se e só se os valores das expressões $e_1$ e $e_2$ são diferentes.                   |
| $e_1 > e_2$  | Números             | Tem o valor <code>True</code> se e só se o valor da expressão $e_1$ é maior do que o valor da expressão $e_2$ .    |
| $e_1 >= e_2$ | Números             | Tem o valor <code>True</code> se e só se o valor da expressão $e_1$ é maior ou igual ao valor da expressão $e_2$ . |
| $e_1 < e_2$  | Números             | Tem o valor <code>True</code> se e só se o valor da expressão $e_1$ é menor do que o valor da expressão $e_2$ .    |
| $e_1 <= e_2$ | Números             | Tem o valor <code>True</code> se e só se o valor da expressão $e_1$ é menor ou igual ao valor da expressão $e_2$ . |

In [ ]:

## Elementos básicos de programação - Predicados e condições

### Exemplos

- `nota = 17` (é isto um predicado?)
- `nota > 10`
- `3 < nota % 2`
- `3 < nota // 2`
- `nota < 9*2 and nota > 10`
- `nota < 9*2 < 25` (*syntactic sugar*)
- `not 10` (qq expressão em Python pode ser tomado por condição)

In [227]: `nota < 9*2 < 25`

Out[227]: `True`

# Elementos básicos de programação - Leitura e escrita

## Leitura de dados (do teclado)

### BNF

<leitura de dados> ::= input() | input(<informação>)

<informação> ::= <cadeia de caracteres>

- A função `input` retorna a *string* introduzida
  - A *string* pode conter caracteres de escape, e.g., `\n`, `\r`, `\t`, `\v`, etc.
- Exemplos:

```
input()  
input('Escreva alguma coisa\n\t ->')
```

In [ ]:

# Elementos básicos de programação - Leitura e escrita

## Função de avaliação de *strings*

### BNF

<função de avaliação> ::= eval(<cadeia de caracteres>)

### Examples

```
eval('200 + 2')  
type(eval('200 + 2'))  
x = eval(input("Introduza uma expressão:\n->\t"))
```

```
In [335]: num = int(eval(input('Introduzir numero:')))  
print(num, type(num))
```

```
Introduzir numero:3.4  
3 <class 'int'>
```

# Elementos básicos de programação - Leitura e escrita

## Função de escrita (no ecrã)

### BNF

```
<output> ::= print() | print(<expressions>)
```

```
<expressions> ::= <expression> | <expression>, <expressions>
```

### Exemplos

```
a = 2
b = 10
print("a = ", a, "b = ", b)
print("a = ", a, "\nb = ", b)
```

```
In [357]: a = 0.00000000000000000000000000000001
          print("a = ", a, "b = ", b, True)

          a = 1e-28 b = 2 True
```

# Elementos básicos de programação - Leitura e escrita

## Outro exemplo:

```
x = eval(input("Introduza uma expressão:\n\t"))
y = input("Introduza uma string:\n\t")
print(x, "e", y)
```

- Qual é o valor resultante de avaliar a função `print()` ?

```
val = print (x, "e", y)
print(val)
```

```
In [360]: val = print (x, "e", y)
          print(val)
```

```
154 e boa tarde
None
```

# Elementos básicos de programação - Tarefas próxima aula

- Trabalhar matéria apresentada hoje
- Ler seções 2.6-2.9 do livro da UC

