Fundamentos da Progamação

Aula 4

Elementos básicos de programação

Expressões. Tipos elementares de informação. Nomes e atribuição.

Alberto Abad, IST, 2024-25

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>`

```
In [ ]:
```

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 carateres (strings)

- 2016
- 2
- 2.0
- -2.0
- +2.0
- 10e-12
- 6376754588877162243232221200091999228887333
- 0.00000000000000000000000000000000000001
- *True*
- *False* (# **atenção** *true* e *false* !)
- 'Hello world'
- "Hello"
- "As strings são sequencias"

```
In [ ]:
```

Elementos básicos de programação - Expressões compostas

BNF:

```
<expressão composta> ::=
        <operador> <expressão> |
        <operador> (<expressão>) |
        <expressão> <operador> <expressão> |
        (<expressão> <operador> <expressão>)
```

- **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

In [ ]:
```
5 != 5
```

Expressões compostas: Prioridade dos operadores (1)

Regra #1 (De maior a menor prioridade)

Operator	Description
`()`	Parentheses (grouping)
`f(args...)`	Function call
`x[index:index]`	Slicing
`x[index]`	Subscription
`x.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 operadores (2)

Regra #2

- Em caso de igualdade, de esquerda para direita

Official info: 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, dictionarios, etc.) nas próximas semanas

In [ ]:
```
isinstance(3.5, float)
```

Elementos básicos de programação - Tipos elementares

O tipo inteiro (`int`)

Operação	*Tipo dos argumentos*	*Valor*
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.
abs(e)	Inteiro	O valor absoluto de e.

Exemplos

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

```
In [ ]: type(5 * (7 // 2))
```

Elementos básicos de programação - Tipos elementares

O tipo real (`float`)

Operação	*Tipo dos argumentos*	*Valor*
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.
`abs`(e)	Real	O valor absoluto de e.

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

```
In [ ]: 1 + 1.0
```

Elementos básicos de programação - Tipos elementares

O tipo real (`float`)

Exemplos

- 7.7
- 7.
- .4
- 2.7e4
- 2000000000000000000000000000000000.
- .00000000000000000000000000000000000000000000001
- 4 + 7.3 *(coercion)*
- 1 * 3 *(overloading 1.0*3.0)*
- 1.0 *(vs 1)*

```
In [ ]: 1 == 1.0
```

Elementos básicos de programação - Tipos elementares

Conversão explícita de tipos inteiros e reais (casting)

Operação	*Tipo do argumento*	*Tipo do valor*	Operação
`round`(e)	Real	Inteiro	O inteiro mais próximo do real e.
`int`(e)	Real	Inteiro	A parte inteira do real e.
`float`(e)	Inteiro	Real	O número real correspondente a e.

Exemplos

- round(3.4)
- int(3.4)
- float(2)

```
In [ ]:
```

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

```
<nome> ::= <nome simples> | <nome indexado> | <nome composto>
```

- Por enquanto veremos o `<nome simples>` ...

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 [ ]:
```
False = True
```

Elementos básicos de programação - Atribuição

Atribuição simples e múltipla

```
<instrução
                <nome> , <instrução atribuição>, <expressão>```

- 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 - Tarefas próxima aula

- Trabalhar matéria apresentada hoje
- Ler seções 2.4-2.5 do livro da UC