Algoritmos e Programação de Computadores

Escrita, Leitura, Expressões e Operadores Aritméticos, Conversão de Tipos

Raquel Cabral

Dr. Eng. Eletrica, FEEC, Unicamp

mjara.perez@ic.unicamp.br

Ref.: material original (1o S., T. KLMN). por **Profa. Sandra Avila**, Instituto de Computação (IC/ Unicamp)

MC102 Turma Z, 09 Agosto, 2018

Agenda

- Saída de dados: `print()`
- Entrada de dados: `input()`
- Expressões e Operadores Aritméticos
- Conversão de Tipos

A Função print()

Escrevendo na Tela: `print()`

- Para imprimir um texto, utilizamos o comando **`print()`**.
- O texto pode ser um literal do tipo **string**.

```
>>> print(“De novo isso?”)
De novo isso?
```

- Em uma **string** podem-se incluir caracteres de formatação especial.
- O símbolo especial `\n` é responsável por avanço de linha (*newline*)

```
>>> print(“De novo isso? \n Olá Pessoal!”)
De novo isso?
Olá Pessoal!
```

Escrevendo o Conteúdo de uma Variável na Tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando **`print()`**.
- Separamos múltiplos argumentos a serem impressos com uma vírgula.

```
>>> a = 10
>>> print(“A variável contém o valor”, a)
A variável contém o valor 10
```

Escrevendo o Conteúdo de uma Variável na Tela

- A impressão com múltiplos argumentos inclui um **espaço extra** entre cada argumento.

```
>>> a = 10
>>> b = 3.14
>>> print(“a contém o valor”, a, “e b contém o valor”, b)
a contém o valor 10 e b contém o valor 3.14
>>> print(“a contém o valor ”, a, “ e b contém o valor ”, b)
a contém o valor  10  e b contém o valor  3.14
```

Escrevendo o Conteúdo de uma Variável na Tela

- Podemos converter todos os valores em strings e usar o **operador +** para concatenar strings de forma a imprimir sem estes espaços:

```
>>> a = 10
>>> b = 3.14
>>> print(“a contém o valor” + str(a) + “e b contém o valor” + str(b))
a contém o valor10e b contém o valor3.14
>>> print(“a contém o valor ” + str(a) + “ e b contém o valor ” + str(b))
a contém o valor 10 e b contém o valor 3.14
```

Formatos Ponto Flutuante

- Podemos especificar o número de casas decimais que deve ser impresso em um número ponto flutuante usando **%.Nf**, onde N especifica o número de casas decimais.

```
>>> pi = 3.1415
>>> r = 7
>>> area = pi * r * r
>>> print(“Área do círculo de raio %.2f ” %r + “é: %.2f” %area)
Área do círculo de raio 7.00 é: 153.93
>>> print(“Área do círculo de raio ” + str(r) + “é: ” + str(area))
Área do círculo de raio 7 é: 153.9335
```

Imprimindo sem Pula Linha

- A função **`print()`** sempre pula uma linha ao final da impressão.
- Se você não quiser que pule uma linha, inclua o argumento **end=' '** no **`print()`**.

```
>>> print(“3, ”, end=“”)
>>> print(“4, ”, end=“”)
>>> print(“5 ”, end=“”)
3, 4, 5
```

A Função input()

A Função `input()`

- Realiza a leitura de dados a partir do teclado.
- Aguarda que o usuário digite um valor e atribui o valor digitado a uma variável.
- **Todos** os dados lidos são do tipo string.

```
>>> print(“Digite um número: ”)
>>> numero = input()
>>> print(“O número digitado é: ” + numero)
```

A Função `input()`

- Podemos converter uma string lida do teclado em um número inteiro usando a função **int()**.

```
>>> print(“Digite um número: ”)
>>> numero = int(input())
>>> numero = numero * 10
>>> print(“O número digitado vezes 10 é: ”, numero)
```

A Função `input()`

- Podemos fazer o mesmo para números ponto flutuante usando a função **float()**.

```
>>> print(“Digite um número: ”)
>>> numero = float(input())
>>> numero = numero * 10
>>> print(“O número digitado vezes 10 é %.2f ” %numero)
```

A Função `input()`

- Nos dois exemplos anteriores é esperado que o usuário digite um número.
- Se o usuário digitar um texto não numérico o programa encerrará com um erro de execução.

```
>>> print("Digite um número: ")
Digite um número:
>>> numero = float(input())
mc102
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'mc102'
```

A Função `input()`

- O programa abaixo lê dois números e imprime a soma destes.
- Perceba que podemos incluir um texto a ser impresso diretamente no comando `input()`.

```
>>> numero1 = float(input("Digite um número: "))
>>> numero2 = float(input("Digite um número: "))
>>> print(“A soma dos números é: %.2f” %(numero1 + numero2))
```

Expressões

Expressões

- Já vimos que constantes e variáveis são expressões.
- Uma expressão também pode ser um conjunto de operações aritméticas, lógicas ou relacionais utilizadas para fazer “cálculos” sobre os valores das variáveis.
  - Exemplo: a + b

Expressões Aritméticas

- Os operadores aritméticos são: `+`, `-`, `*`, `/`, `//`, `%`, `**`

- **Adição**: expressão `+` expressão
- **Subtração**: expressão `-` expressão
- **Multiplicação**: expressão `*` expressão

```
>>> 30 + 5
35
>>> 30 - 5
25
>> 30 * 5
150
```

Expressões Aritméticas

- **Divisão**: expressão / expressão
  - O resultado é sempre um número ponto flutuante.
- **Divisão**: expressão / / expressão
  - Se os operandos forem inteiros, a divisão é inteira. Se um deles for ponto flutuante faz uma divisão truncada.

```
>>> 5 / 2
2.5
>>> 5 // 2
2
>>> 5 // 2.0
2.0
```

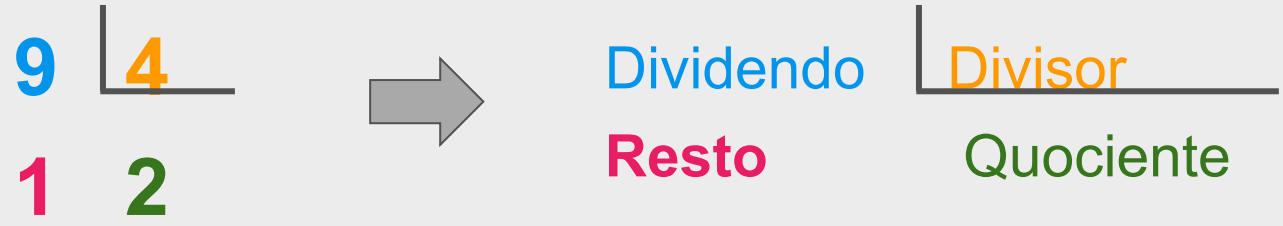
Expressões Aritméticas

- **Exponenciação (potenciação)**: expressão `**` expressão
  - Calcula o valor da expressão à esquerda elevado ao valor da expressão à direita.
  - $a^n = a \times a \times a \times \ldots \times a$ (n vezes)

```
>>> 2 ** 4
16
>>> 2.2 ** 4
23.425600000000006
```

Expressões Aritméticas

- **Resto da Divisão**: expressão % expressão
  - Calcula o resto da divisão inteira de duas expressões.



Dividendo = Divisor * Quociente + **Resto**

Expressões Aritméticas

- **Resto da Divisão**: expressão % expressão
  - Calcula o resto da divisão inteira de duas expressões.

```
>>> 5 % 2
1
>>> 9 % 7
2
>>> 2 % 5
2
>>> 4 % 2
0
```

Expressões Aritméticas

- Exemplo: Converter segundos em horas, minutos e segundos.
  - 87426 segundos = horas, minutos e segundos?

```
>>> segundos_str = input(“Por favor, digite o número de segundos que deseja converter: ”)
>>> total_segundos = int(segundos_str)
>>> horas = total_segundos // 3600
>>> segundos_restantes = total_segundos % 3600
>>> minutos =  segundos_restantes // 60
>>> segundos_restantes_final = segundos_restantes % 60
>>> print(“Horas =”, horas, “minutos =”, minutos, “segundos =” , segundos_restantes_final)
```

Expressões

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo
  a = b * ((2 / c) + (9 % d * 8))

```
>>> 5 + 10 % 3
6
>>> 5 * 10 % 3
2
```

Precedência

- Precedência é a **ordem** na qual os operadores serão avaliados quando o programa for executado.
- Em Python, os operadores são avaliados na seguinte ordem:
  1. **
  2. *, /, //, na ordem em aparecerem na expressão
  3. %
  4. +, -, na ordem em aparecerem na expressão
- Exemplo: 8 + 10 * 6 é igual a 68

Precedência

- Operadores com a mesma precedência são executados da esquerda para a direita.
  - 6-3+2 é igual a ?
    - Esquerda para direita: 6-3 é igual a **3**, **3**+2 é igual a 5
    - ~~Direita para esquerda: 3+2 é igual a **5**, 6-**5** é igual a 1~~
- **Atenção**: Uma exceção é o operador exponenciação **.
  - 2 ** 3 ** 2 é igual a ?
    - ~~Esquerda para esquerda: 2 ** 3 é igual a **8**, **8** ** 2 é igual a 64~~
    - Direita para esquerda: 3 ** 2 é igual a **9**, 2 ** **9** é igual a 512

Alterando a Precedência

- **(expressão)** também é uma expressão, que calcula o resultado da expressão dentro dos parênteses, para só então calcular o resultado das outras expressões.
  - 5 + 10 % 3 é igual a 6
  - (5 + 10) % 3 é igual a 0
- Você pode usar quantos parênteses desejar dentro de uma expressão.
- Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada!

Conversão de Tipos

Conversão de Tipos

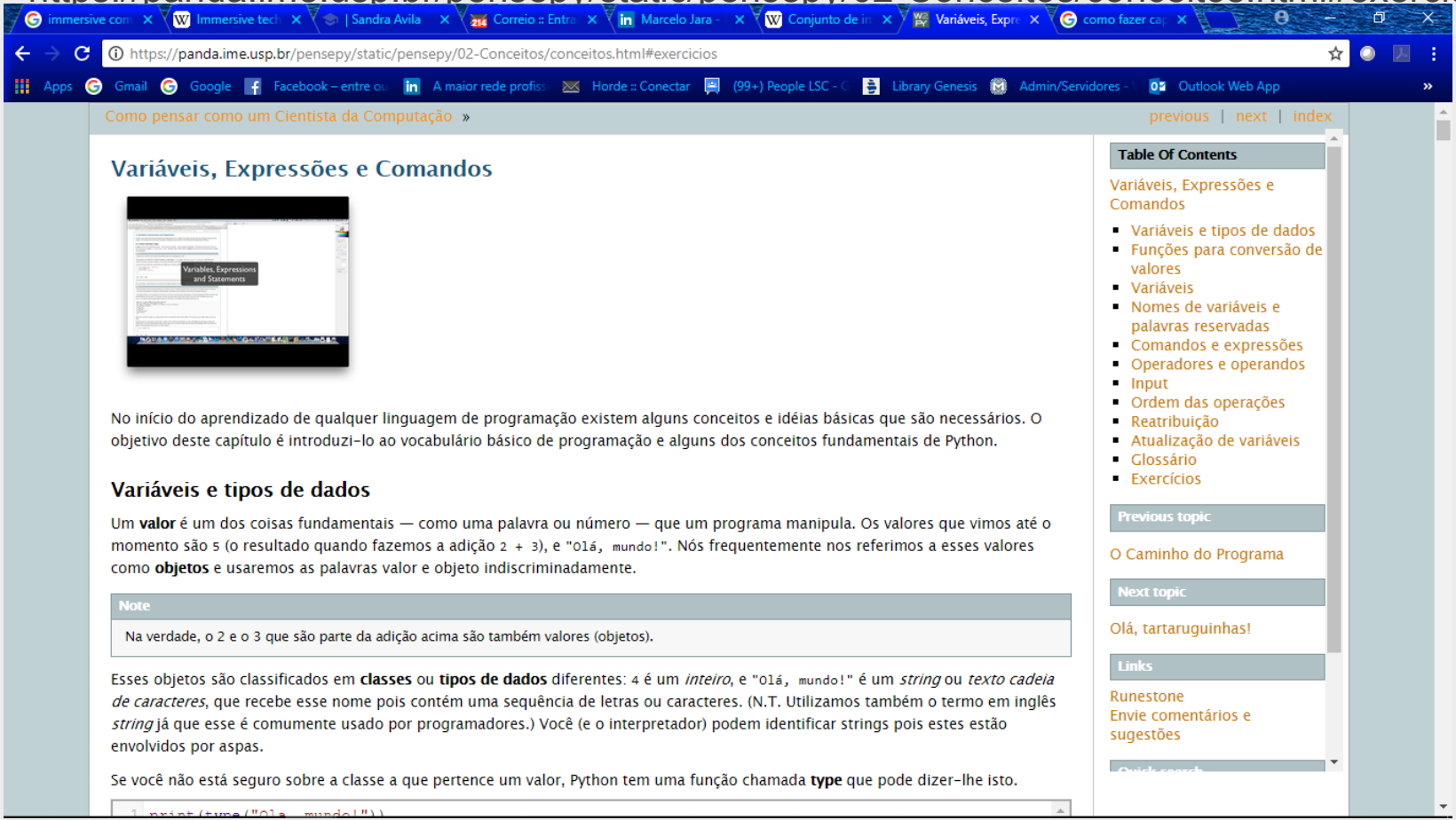
- Já vimos o uso das funções **int()**, **float()** e **str()** que servem para converter dados de um tipo no outro especificado pela função.
- A conversão só ocorre se o dado estiver bem formado. Por exemplo **int(“aaa”)** resulta em um erro.
- Ao convertermos um número float para int ocorre um truncamento, ou seja, toda parte fracionária é desconsiderada.

Conversão de Tipos

```
>>> a = “ola”
>>> int(a)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: ’ola’
>>> int(2.99)
2
>>> int(-2.99)
-2
>>> float(“3.1415”)
3.1415
```

Exercícios

- https://panda.ime.usp.br/pensepy/static/pensepy/02-Conceitos/conceitos.html#exercicios



Sobre a aula passada ...

Representações Numéricas

Representações no sistema **decimal** : 0, 1, ... , 9, 10, 11 , ..
No sistema **hexadecimal**: 0, 1, .., 9, A, B, C, D, E, F, 10, ...

Ex. $(9230)_{10} = (22016)_8 = (240E)_{16} = (0010\ 0100\ 0000\ 1110)_2$

Base **10** (sistema **Decimal**) ex. **9230** > **9** x $\mathbf{10}^3$ + **2** x $\mathbf{10}^2$ + **3** x $\mathbf{10}^1$ + **0** x $\mathbf{10}^0$

Base **8** (**Octal**) ex. **22016** > **2** x $\mathbf{8}^4$ + **2** x $\mathbf{8}^3$ + **0** x $\mathbf{8}^2$ + **1** x $\mathbf{8}^1$ + **6** x $\mathbf{8}^0$

Base **16** (**Hexadecimal**) ex. **240E** > **2** x $\mathbf{16}^3$ + **4** x $\mathbf{16}^2$ + **0** x $\mathbf{16}^1$ + **E** x $\mathbf{16}^0$

Base **2** (**Binário**) > 0010_0100_0000_1110 i.e. representado em 16 bits (***binary digits)***
Digitos mais significativos: MSB (***Most Significant Bits***): desde esquerda a direita, ex. 0010 ...

Isto significa que o no. $(\mathbf{9230})_{10}$, precisa de no mínimo **14 bits** para ser representado no **sistema binário** (assumindo que os 2 dígitos mais significativos, **00**, podem ser desprezados)

Floating Point Math: https://0.3000000000000004.com/

```
>>> 0.1 + 0.2 - 0.3
5.551115123125783e-17
```

- Números no nosso sistema numérico são representados em base ao **sistema decimal** (base **10**)
- Números em um computador digital são representados no **sistema binário** (base **2**)
- Números reais (ponto flutuante) em um computador digital devem ser aproximados para ser adequadamente processados e representados internamente pela máquina.
- Os sistemas numéricos mais conhecidos são : decimal, octal, hexadecimal, binário

Aritmética de ponto flutuante: problemas e limitações

http://turing.com.br/pydoc/2.7/tutorial/floatingpoint.html

```
>>> 0.1 + 0.2
0.30000000000000004
```

Números reais (ponto flutuante) são representados no computador como frações **binárias** (utilizando a **base 2**). Por exemplo, a fração decimal:

($\mathbf{0.125}$)$_{10}$ (número fracionário, representação no sistema **decimal**)

Tem o valor calculado na máquina pela expressão> 1/10 + 2/100 + 5/1000 (frações **base 10**)

($\mathbf{0.001}$)$_2$ (mesmo número fracionário anterior, agora em representação no sistema **binário**) tem o valor calculado pela expressão > 0/2 + 0/4 + 1/8 (frações em **base 2**)

- muitas frações decimais **não podem** ser representadas de forma **precisa** ou **exata** como frações binárias no computador, portanto estes números devem ser **aproximados**.

Referências

- Livro: Python 3 for Absolute Beginners, por Tim Hall & J-P Stacey, Apress (2009).

- O slides dessa aula baseados no material de MC102 , 1o semestre, da Prof. Sandra Avila e do Prof. Eduardo Xavier (IC/Unicamp)

- Representação de números em ponto flutuante e fixo (PyMOTW-3, em inglês)

  https://pymotw.com/2/decimal/

- Variáveis, Expressões e Comandos:
  - https://panda.ime.usp.br/pensepy/static/pensepy/02-Conceitos/conceitos.html#