Programação de Computadores

INTRODUÇÃO AOS ALGORITMOS E À PROGRAMAÇÃO DE COMPUTADORES – PARTE 3

Renato Dourado Maia

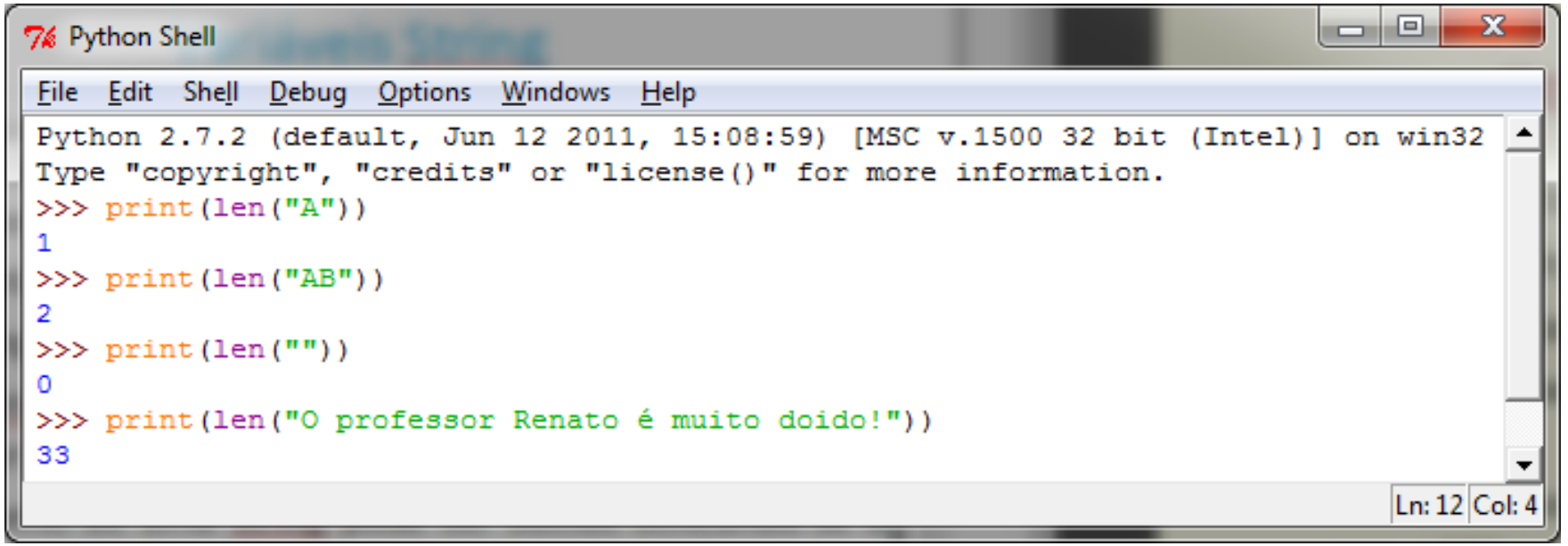
Instituto de Ciências Agrárias

Universidade Federal de Minas Gerais

Variáveis *String*

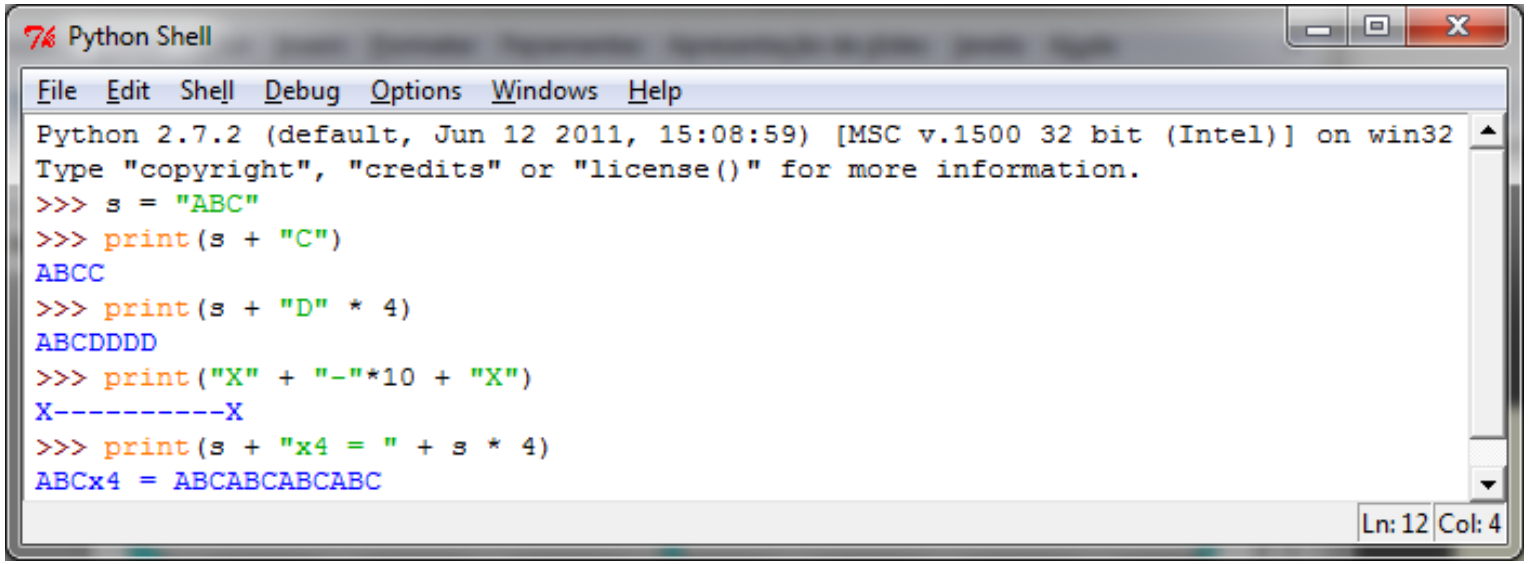
- ***Strings*** são **cadeias de caracteres** e constituem outro **tipo fundamental** da linguagem Python.
- Constantes `string` são escritas utilizando-se **aspas** ou **apóstrofos**:
  - Exemplo: “a” ou 'a'.
- Uma ***string*** tem um **tamanho associado** e o seu **conteúdo** pode ser **acessado caractere a caractere**.
- O **tamanho** de uma `string` pode ser obtido utilizando-se a **função `len`**.
  - Função? Já utilizamos a função **`print`**! **Estudaremos funções com mais detalhes posteriormente**. :-)

Variáveis *String*



Variáveis *String*

- O **operador +** pode ser utilizado para **concatenar** *strings*:
  - Exemplo: “a” + “b” é o mesmo que “ab”.
- O **operador *** pode ser utilizado para **repetir** *strings*:
  - Exemplo: “a”*10 é o mesmo que “aaaaaaaaaa”.



Variáveis *String*

- Caracteres **não imprimíveis** podem ser expressos utilizando-se a notação **"barra invertida"** (\):
  - **`\n`** = *new line*.
  - **`\r`** = *carriage return*.
  - **`\t`** = *tab*.
  - **`\b`** = *backspace*.
  - **`\\`** = \.
  - **`\x41`** = caractere cujo código **hexadecimal** é 41 ("A" maiúsculo).

Variáveis *String*

```
>>> "ab\rd"
'ab\rd'
>>> print("ab\rd") # print exibe chars não imprimíveis
db
>>> print("abc\td")
abc d
>>> print("abc\nd")
abc
d
>>> print("abc\\nd")
abc\nd
>>> print("ab\bc")
ac
>>> print("\x41\xA1")
Aí
```

Variáveis *String*

- Para **desabilitar** a notação "barra invertida" (\), basta **preceder a constante *string* por um "r"** (erre minúsculo), o que transforma a *string* em uma ***raw string***.

```
>>> print("abc\ncd\tef")
abc
cd ef
>>> print(r"abc\ncd\tef")
abc\ncd\tef
```

Variáveis *String*

- Constantes `string` podem ser escritas com **várias linhas**, desde que **as aspas não sejam fechadas** e que **cada linha termine com uma barra invertida**.

```
>>> print("abcd\n\
... efgh\n\
... ijk")
abcd
efgh
ijk
>>> print("abcd\
... efgh\
... ijk")
abcdefghijk
```

Variáveis *String*

- **Também** é possível escrever constantes `string` em **várias linhas**, incluindo as quebras de linha, utilizando-se **três aspas (ou apóstrofos) como delimitadores**:

```
>>> print("""
Um tigre,
dois tigres,
três tigres""")
Um tigre
dois tigres
três tigres
```

Variáveis *String*

- Para **acessar os caracteres** de uma `string`, deve-se informar o **índice** entre **colchetes** ([ ]).
- O **primeiro caractere** tem **índice 0** e o **último** tem **índice igual ao tamanho da *string* – 1**.
  - Se uma `string` possui 10 caracteres, o último caractere possui índice 9.
  - O **último caractere** pode ser acessado pelo **índice -1**.

Variáveis *String*

```
>>> a = "ABCDE"
>>> a[0]
'A'
>>> a[1]
'B'
>>> a[5]

Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    a[5]
IndexError: string index out of range
>>> print(len(a))
5
>>> a[-1]
'E'
```

Variáveis *String*

- *Strings* podem ser **fatiadas**, o que permite que sejam **separados trechos**:
  - **Notação**: *string*[índice1: índice2].
    - Retorna os caracteres desde o índice1 (inclusive) até o índice2 (exclusive).
    - Se o primeiro índice é omitido, assume-se 0.
    - Se o último índice é omitido, assume-se o fim da *string*.

```
>>> a
'abcde'
>>> a[0:2]
'ab'
>>> a [2:]
'cde'
```

```
>>> a[:]
'abcde'
>>> a[-1:]
'e'
>>> a[:-1]
'abcd'
```

Lembrando da Última Aula...

$$y(t) = v_0\, t - \frac{1}{2} g\, t^2 \rightarrow \begin{cases} \textit{velocidade inicial } (\textit{em } t=0) \\ \textit{aceleração da gravidade} \end{cases}$$

$$\left.\begin{array}{r} v_0 = 5\ m/s \\ g = 9{,}81\ m/s^2 \\ t = 0{,}6\ s \end{array}\right\} \rightarrow y = 5 \cdot 0{,}6 - \frac{1}{2} \cdot 9{,}81 \cdot 0{,}6^2$$

```
# Programa para calcular a altura de uma bola num
# lançamento vertical
v0 = 5    # Velocidade inicial
g = 9.81 # Aceleração da gravidade
t = 0.6   # Tempo
y = v0*t - 0.5*g*t**2 # Posição vertical da bola
print(y)
```

Formatando Texto e Números

- O programa apresentado no *slide* anterior **apenas apresenta o valor de y**.
- Seria mais interessante apresentar uma **mensagem mais informativa**, como, por exemplo:
  - `Em t = 0.6 s, a altura y = 1.23 m.`
- Para **"construir"** mensagens como essa, utiliza-se a **composição de *strings***.
  - `print("Em t = %g s, y = %.2f m." % (t, y))`
- **`%d`** e **`%.2f`** são **marcadores de posição**, enquanto o último símbolo **`%`** indica a **composição de strings**.

Formatando Texto e Números

- Exemplos de **marcadores de posição**:
  - `%s`: `string`
  - `%d`: inteiro
  - `%0xd`: inteiro com x “posições”.
  - `%f`: notação decimal com seis casas decimais.
  - `%e`: notação científica compacta, com e no expoente.
  - `%E`: notação científica compacta, com E no expoente.
  - `%%`: o próprio símbolo %.

Formatando Texto e Números

```
# Programa para calcular a altura de uma bola num
# lançamento vertical
v0 = 5    # Velocidade inicial
g = 9.81 # Aceleração da gravidade
t = 0.6   # Tempo
y = v0*t - 0.5*g*t**2 # Posição vertical da bola
print ("""
Em t = %f s, uma bola com
velocidade inicial v0 = %.3E m/s
está localizada na altura %.2f m.
""" % (t, v0, y))
```

Saída

```
Em t = 0.600000 s, uma bola com
velocidade inicial v0 = 5.000E+00 m/s
está localizada na altura 1.23 m.
```

Objetos

- Tudo em Python é um **objeto** e as variáveis são os "nomes" dos objetos:
  - `a = 5 # a - objeto int.`
  - `b = 9 # b - objeto int.`
  - `c = 9.0 # c - objeto float.`
  - `d = b/a # d - Python 2: int; 3: float.`
  - `s = 'b/a=%g' % (b/a) # s - objeto str.`
- O conceito de objeto é **mais elaborado** e talvez o estudaremos com mais **detalhes** na disciplina **Programação de Computadores**.

Mais Sobre Tipos

- Vimos na aula passada que o **tipo** de uma variável **muda** de acordo com o valor a ela atribuído.
  - Python é uma linguagem de **tipagem dinâmica**.
    - Não confundir com **linguagens sem tipo**!
- Pode-se **conhecer** o tipo de um objeto por meio da função `type`:

```
>>> C = 1
>>> type(C)
<type 'int'>
```

Mais Sobre Tipos

- É possível a **"conversão"** de tipos (sempre que isso **fizer sentido**):

```
>>> C = 1
>>> type(C)
<type 'int'>
>>> C = float(C)
>>> type(C)
<type 'float'>
```

Funções Embutidas

- Além dos **operadores** que já vimos, é possível computar valores por meio de **funções**
- As funções podem ser **definidas**:
  - Pelo **programador** (estudaremos depois).
  - Em **módulos da biblioteca padrão** (estudaremos o que são módulos com mais detalhes depois).
  - Por ***default***, o que corresponde na verdade às **funções embutidas** (*built-in*).
    - As **funções embutidas** na verdade fazem parte do **módulo __builtins__**, que é **sempre importado em toda aplicação**.

Funções Embutidas

- Exemplos:
  - `abs(x)` retorna o **valor absoluto** do número x.
  - `cha(x)` retorna **uma `string`** com um único caractere cujo código ASCII é x.
  - `ord(s)` retorna o **código ASCII** do caractere s.

```
>>> abs (10)
10
>>> abs (-19)
19
>>> chr (95)
'_'
```

```
>>> chr (99)
'c'
>>> ord ('a')
97
```

Voltando ao Exemplo da Bola

- Quanto tempo a bola demora para alcançar a altura y_c?

$$y(t) = v_0\, t - \frac{1}{2}\, g\, t^2 \rightarrow \begin{cases} velocidade\ inicial\ (em\ \ t=0) \\ aceleração\ da\ gravidade \end{cases}$$

$$y_c = v_0\, t - \frac{1}{2}\, g\, t^2 \rightarrow \begin{cases} t_1 = \left(v_0 - \sqrt{v_0^2 - 2\, g\, y_c} \right) / g \\ t_2 = \left(v_0 + \sqrt{v_0^2 - 2\, g\, y_c} \right) / g \end{cases}$$

Precisaremos da função "raiz quadrada"?

Voltando ao Exemplo da Bola

```
v0 = 5
g = 9.81
yc = 0.2
t1 = (v0 - (v0**2 - 2*g*yc)**0.5)/g
t2 = (v0 + (v0**2 - 2*g*yc)**0.5)/g
print ('Em t = %g s e %g s,\
a altura da bola é %g m.' % (t1, t2, yc))
```

Importando Módulos

- **Funções** como **raiz quadrada**, **seno** e **diversas outras** estão disponíveis no módulo `math` da **biblioteca padrão**.
- Um módulo pode conter também **variáveis** e **classes**.
  - O módulo `math`, por exemplo, define a constante `pi`.
- Para se **utilizar** os elementos de um módulo, pode-se utilizar o comando `import`:
  - `import module`
    - `import module as nome`
  - `from module import nome, …, nome`
  - `from module import *`

Importando Módulos

- Exemplos:
  - `from math import *`

    `# importa todos os elementos do módulo math`
  - `from math import sin`

    `# importa apenas a função sim`
  - `import math`

    `# importa o módulo math como um todo`

    `# (os elementos têm que ser citados pre pre-`
    `# cedidos por math)`
  - `import math as m`

    `# análogo ao caso anterior`

Importando Módulos

```
>>> import math
>>> a = sin(30)
Traceback (most recent call last):
File "<stdin>", line 1, in ?
NameError: name 'sin' is not defined
>>> a = math.sin(30)
>>> from math import sin
>>> a = sin(30)
>>> a = sin(radians(30))
Traceback (most recent call last):
File "<stdin>", line 1, in ?
NameError: name 'radians' is not defined
>>> from math import *
>>> a = sin(radians(30))
```

Explorando Módulos

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import math
>>> dir()
['__builtins__', '__doc__', '__name__', '__package__', 'math']
>>> dir(math)
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot
', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan'
, 'tanh', 'trunc']
>>> help(math.cos)
Help on built-in function cos in module math:

cos(...)
    cos(x)

    Return the cosine of x (measured in radians).

>>> help(math)
Help on built-in module math:

NAME
    math

FILE
    (built-in)

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(...)
        acos(x)

        Return the arc cosine (measured in radians) of x.

    acosh(...)
        acosh(x)
```

Ln: 245 Col: 4

Voltando Novamente ao Exemplo da Bola

$$y_c = v_0 t - \frac{1}{2} g t^2 \rightarrow \begin{cases} t_1 = \left(v_0 - \sqrt{v_0^2 - 2 g y_c} \right) / g \\ t_2 = \left(v_0 + \sqrt{v_0^2 - 2 g y_c} \right) / g \end{cases}$$

```
import numpy as np
v0 = 5
g = 9.81
yc = 0.2
p = np.poly1d([0.5*g, -v0, yc])
print (p)
print(p.r)
```

Por Hoje é Só!