Python em poucas horas!

Minicurso EnjoyTEM-MT 2017
Prof. Dr. Ivairton M. Santos
Curso de Ciência da Computação/UFMT

Introdução

- **Python** é uma linguagem de programação criada por Guido van Rossum em 1991.
- O objetivo do projeto foi **produtividade** e **legibilidade**.
- A expectativa é que a linguagem proporcione código **bom** e **fácil de manter**.
- É uma linguagem dinâmica e interpretada.
- Versão atual é a 3.6.1.
- Onde estudar mais sobre Python:
  - http://www.python.org/
  - http://www.pythonbrasil.com.br/
  - "*E viva o Google!!!*"

Histórico

- Concebida no fim dos anos 80.
- Surgiu com o objetivo de "preencher o vazio entre C e o Shell" (*Guido Van Rossum*).
- O nome surgiu em homenagem ao programa de TV "Monty Python's Flying Circus", mas o autor acabou desistindo de vincular à cobra píton, com a publicação do livro da O'Reilly "Programming Python".

Vamos começar!

- Baixar os arquivos de exercícios
  - Link:
- Verificar o interpretador:

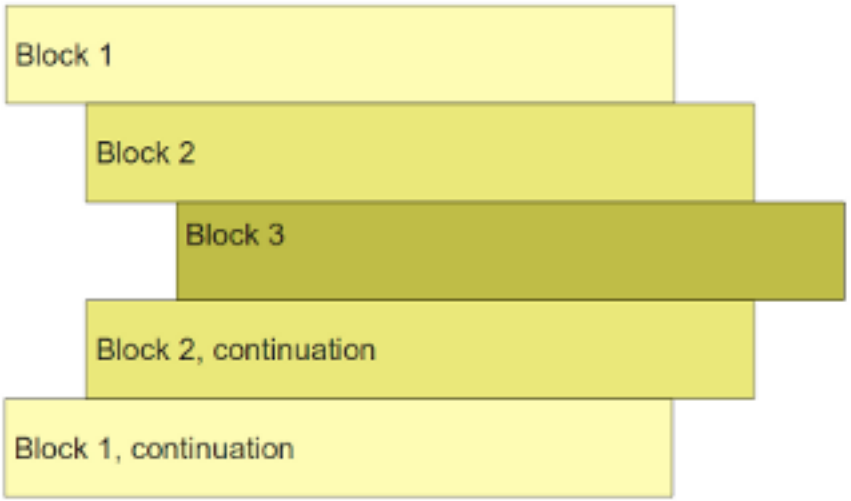
```
$ python
>>> 1 + 1
2
>>> print "Olá Mundo!"
Olá Mundo!
>>> quit() // Ou digite CTRL+D
$ python hello.py
```

- No Linux é possível definir o arquivo com permissão de execução e rodá-lo diretamente:

```
$ chmod +x hello.py
$ ./hello.py
```

Apresentando a linguagem

- Não há declaração de variáveis!
- Não é necessário uso de “;”
- Comentário inicia com “#”
- Extensão dos arquivos fonte deve ser “.py”
- Os blocos de comandos são definidos pela identação!



Apresentando a linguagem

- Experimente executar o seguinte código diretamente no interpretador:

```
$ python
>>> a = 6
>>> a
6
>>> a + 2.5
8.5
>>> a = ‘Oi’
>>> a
'Oi’
>>> len(a)
2
>>> a +  len(a)
###ERRO de Tipo!
>>> a + str( len(a) )
‘Oi2’
>>> teste
###ERRO teste não definido
>>>^D
```

Variáveis

- Variáveis:
  - Referência x Cópia
- Números

```
>>> x = 4  #Python 2
>>> y = 3
>>> z = x / y
>>> print(z)
1
(...)
>>> z = x // y #Python 3
>>> print(z)
1
```

```
>>> x = 42
>>> id(x)
 10107136
>>> y = x
>>> id(x), id(y)
 (10107136, 10107136
>>> y = 78
>>> id(x), id(y)
 (10107136, 10108288
```

```
>>> i = 4321
>>> f = 2.45
>>> oct = 0o10
>>> hex = 0xA0F
>>> bin = 0b101010
>>> x = hex(19)
>>> type(x)     # <C
>>> x = bin(20)
>>> x = oct(60)
```

```
>>> x =
12345678910111213141516171
81920
>>> x = x * x * x
>>> print(x)
1881676376412480405375011
6312135846610383322617044
1424087480154237168444110
563454813388800 0
```

```
>>> x = 3 + 4j
>>> y = 2 - 3j
>>> z = x + y
>>> print(z)
(5+1j)
```

Strings

- String

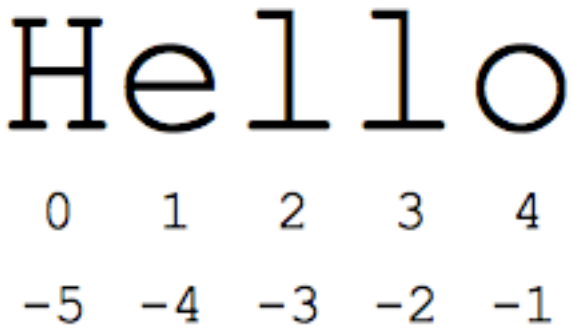
```
>>> s = ‘Oi’
>>> print s[1]              # i
>>> print len(s)            # 2
>>> print s + ‘ amigo’      # Oi amigo

>>> pi = 3.14
>>> print “O valor de PI eh ” + pi
 ERRO!!!
>>> print “O valor de PI eh ” + str(pi)

>>> str = ‘isso eh \t uma \nfrase’
>>> str
 isso eh \t uma \nfrase
>>> print(str)
 isso eh       uma
 frase
```

Strings (cont.)

- Métodos para manipulação de Strings:
  - s.lower() / s.upper()
  - s.strip()
  - s.isalpha() / s.isdigit() / s.isspace()
  - s.startswith('other') / s.endswith('other')
  - s.find('other')
  - s.replace('old', 'new')
  - s.split('delim')
  - s.join(list)

Strings (cont.)

- Manipulação dos caracteres da Strings:
  - s[1:4] ➔ 'ell'
  - s[1:] ➔ 'ello'
  - s[:] ➔ 'Hello'
  - s[1:100] ➔ 'ello'
  - s[-1] ➔ 'o'
  - s[-4] ➔ 'e'
  - s[:-3] ➔ 'He'
  - s[-3:] ➔ 'llo'



- Operador %:
  - text = ("%d cachorros fazem %s %s" % (4,'au','au'))

Comando IF

- Comando de decisão **if**
  - Não há { } para marcação de bloco. É utilizado o **:** com identação
  - Não há necessidade de ( ) no teste lógico
  - Um if pode conter um **elif** ou **else**
  - No teste lógico, o "Zero" significa falso (0, false, nulo, string vazia, etc)
  - Os operadores lógicos são **and**, **or**, **not**

```
if  velocidade > 80:
    print "Carteira de motorista e documento do carro!"
        if humor = 'terrivel' and  velocidade >= 100:
            print "Vc tem o direito de permanecer calado!"
        elif humor = 'ruim' or velocidade >= 90:
            print "Vc vai receber uma multa!"
        else
            print "Dirija sempre nos limites de velocidade."
```

Estrutura de um arquivo .py e função principal

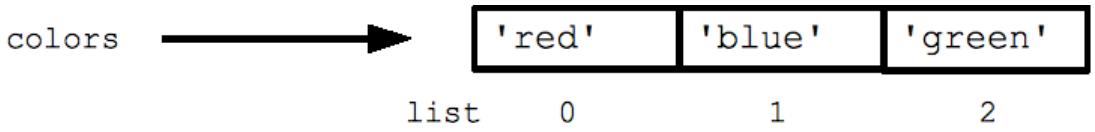
```
# UFMT – EnjoyTEC-MT 2017

import <nomeDoModulo>

def soma(x, y) :
    return x + y

def main() :
    a = 2
    b = 5
    print soma (a, b)

if __name__ == "__main__"
    main()
```

Vamos ao exercício 1!
Arquivo: string1.py

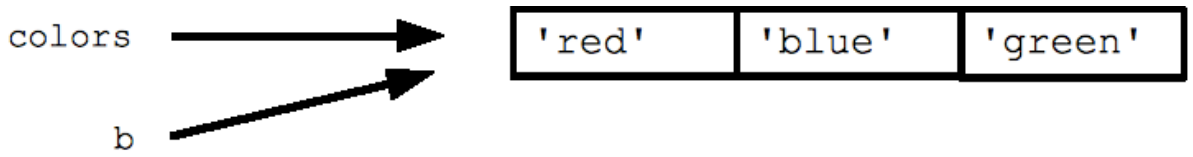
Listas

- Python é generoso na manipulação de listas

```
colors = ['red', 'blue', 'green']
print colors[0]    # red
print colors[2]    # green
print len(colors)  # 3
```




```
b = colors    # não é feita uma cópia, é feita uma referência
```



- Uma lista vazia é representada por **[ ]**
- O operador **+** concatena. Ex: [1, 2] + [3, 4] resulta em [1, 2, 3, 4]

Laço PARA e operador IN

- Muito útil para processar listas:

```
squares = [1, 4, 9, 16]
sum = 0
for num in squares:
    sum += num
print sum      # 30
```

```
list = ['Joao', 'Maria', 'Pedro']
if 'Maria' in list:
    print 'Sim, encontrei!'
```

Função RANGE e laço WHILE

- A função range retorna um intervalo de valores, mas não o último!

```
# Imprime os números de 0 a 99
for i in range(100) :
    print i
```

- Laço com while

```
# Acessa todo terceiro elemento da lista
i = 0
while i < len(a):
    print a[i]
    i = i + 3
```

Métodos para listas

- Métodos mais comuns:
  - **list.append(elem)** – Adiciona um elemento no fim.
  - **list.insert(index, elem)** – Adiciona um elemento na posição, movendo os demais para direita.
  - **list.extend(list2)** – Adiciona os elementos de list 2 em list.
  - **list.index(elem)** – Busca pelo elemento e retorna sua posição. Teste antes com *in*.
  - **list.remove(elem)** – Busca e remove a 1a ocorrência do elemento. Teste antes com *in*.
  - **list.sort()** – Ordena a lista.
  - **list.reverse()** – Inverte a lista
  - **list.pop(index)** – Remove e retorna o elemento na posção definida. Retorna o último se o parâmetro for nulo.
- Atenção: esses métodos não retornam nada!

Ordenação

- Função sorted(list) (retorna a lista ordenada):

```
a = [5, 1, 4, 3]
print sorted(a)  # [1, 3, 4, 5]
print a          # [5, 1, 4, 3]
```

```
strs = ['aa', 'BB', 'zz', 'CC']
print sorted(strs)                  # ['BB', 'CC', 'aa', 'zz']
print sorted(strs, reverse=True)    # ['zz', 'aa', 'CC', 'BB']
```

```
strs = ['ccc', 'aaaa', 'd', 'bb']
print sorted(strs, key=len)         # ['d', 'bb', 'ccc', 'aaaa']
```

- Método sort() (modifica a lista):

```
lista.sort()        # modo correto de utilizar
```

Tuplas

- Estrutura muito útil no uso com banco de dados:
  - É uma estrutura semelhante às listas
  - Mas é imutável e fixa em relação ao tamanho

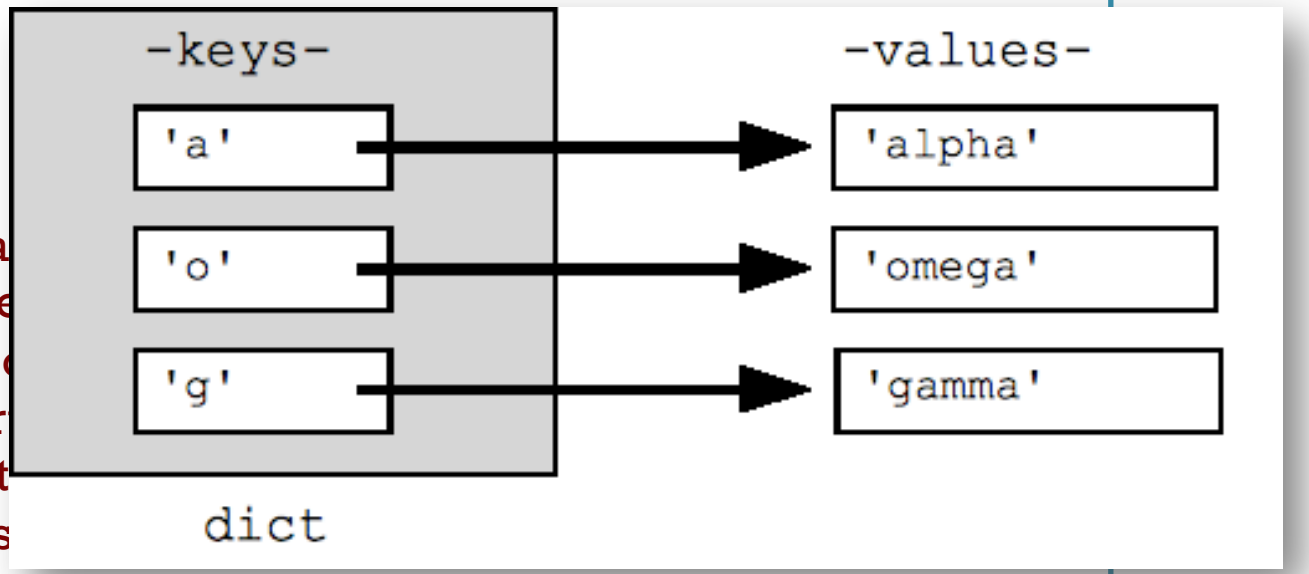
```
tuple = (1, 2, 'oi')
print len(tuple)      # 3
print tuple[2]        # oi
tuple[2] = 'bye'      # ERRO: tupla não pode ser modificada
tuple = (1, 2, 'bye') # Isto está correto
```

Vamos ao exercício 2!
Arquivo: list1.py

Dicionário (Tabela Hash)

- Dicionário consiste numa série de valores indexados por uma chave.
  - Exemplo: dic = {chave1:valor1, chave2:valor2,..., chave*n*:valor*n*}

```
dict = {}
dict['a'] = 'alpha'
dict['g'] = 'gamma'
dict['o'] = 'omega'
print dict          # {'a
print dict['a']     # Re
dict['a'] = 6       # C
'a' in dict         # Tr
# print dict['z']   # Ist
if 'z' in dict:     # Us
    print dict['z']
```



Arquivos

- A manipulação de arquvios segue o mesmo princípio das outras linguagens:
  - Abertura, com especificação do modo de leitura/escrita
  - Leitura/escrita
  - Fechamento
- Modo de abertura: r, w, a

```
arquivo = open('arqEntrada.txt', 'rU')
for linha in arquivo:
    print linha
arquivo.close()
```

- Métodos:
  - arqvuio.readlines() – Lê o arquivo para
  - arqvuio.read() – Lê o arqu
  - arqvuio.write() – Escreve

Vamos ao exercício 3!
Arquivo: wordcount.py

Referências

- A Histório do Python (http://mindbending.org/pt/a-historia-do-python)
- Google for Education – Python Course (https://developers.google.com/edu/python)
- CodeCademy – Python (https://www.codecademy.com/learn/python)
- Coursera – An Introduction to Interactive Programming in Python (https://www.coursera.org/learn/interactive-python-1)

Obrigado!!!

Python em poucas horas!

Minicurso EnjoyTEM-MT 2017

Prof. Dr. Ivairton M. Santos

Curso de Ciência da Computação/UFMT