Mini-Curso de Python

Alex Augusto da Luz dos Santos – nowayx@terra.com.br

Histórico

- Criada em 1991
- Ensino de Programação
- Simplicidade e fácil aprendizado
- *Monty Python Flying Circus*

Características

- Interpretada
- Orientada a Objetos
- Suporte a outros paradigmas
  - Estrutural
  - Funcional
- Multiplataforma
  - Windows, Unix/Linux, MacOS, PalmOS
- Fácil integração com outras linguagens

Características

- Licença GPL-compatível
- Tudo é objeto
- Fortemente tipada
- Tipagem dinâmica
- Poderosas estruturas de dados nativas
  - Listas
  - Dicionários
- Documentação permanece com o código

Vamos Começar!

```
>>> print 'Hello Python!‘
Hello Python!
```

Vamos Começar

- Esqueça declarações de tipos de variáveis;
- Esqueça begin e end;
- Esqueça { e };
- Se você já era organizado, não sofrerá!
- A identação é obrigatória! :)

Variáveis Numéricas

- Imutáveis

```
num_int = 13
num_int_long = 13L
num_real = 13.0
```

Strings

- Imutáveis

‘python’ + ‘ powered’
‘python powered’

‘python’ * 3
‘pythonpythonpython’

Strings

‘python’[0]
‘p’

‘python’[2:4]
‘th’

‘python’[4:]
‘on’

‘python’[-2:]
‘on’

Strings

- Imutáveis

Principais Métodos:

split, count, index, join, lower, upper, replace

var = ‘o guia do mochileiro das galaxias’
var.split()
['o', 'guia', 'do', 'mochileiro', 'das', 'galaxias']

Tuplas

- Formadas por elementos de qualquer tipo
- Delimitadas por parenteses. ‘(‘ e ‘)’
- Imutáveis

```
>>> tupla = (‘RS’, ‘Rio Grande do Sul’)
>>> tupla[1]
‘Rio Grande do Sul’
```

Listas

- Formadas por elementos de qualquer tipo
- Delimitadas por colchetes. ‘[‘ e ‘]’

```
>>> lista = [‘elemento1’, 0, 1, 2, (‘x’, ‘y’)]
>>> lista.sort()
>>> lista
[0, 1, 2, [‘x’, ‘y’], ‘elemento1’]
```

Listas

Principais Métodos:

append, count, index, insert, pop, remove, reverse, sort

```
lista = []
lista.append('laranja')
lista.append('kiwi')
lista.pop()
```

Listas

```
>>> lista = range(10)
>>> lista
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> max(lista)
9
>>> min(lista)
0
>>> len(lista)
10
>>> sum(lista)
45
```

Entrada de Dados

```
nome = raw_input(‘digite o seu nome: ‘)
idade = int(raw_input(‘digite a sua idade: ‘)
```

Condições

```
if exp:
    #comandos
else:
    #comandos
```

```
if exp:
        #comandos
elif exp:
        #comandos
else:
        #comandos
```

Expressões

var1 == var2

var1 != var2

>, <, >=, <=, isinstance(obj, class), is

var1 in var2 # var2 = string, lista, tupla ou dict.

var1 not in var2

Repetição

```
for num in range(200):
    print num

for letra in ‘python’:
    print letra
```

Repetição

```
while exp:
    #comandos
    if exp:
        break
    elif exp:
        continue
    #comandos
```

Funções

```
def exemplo():
    pass

def exemplo(a,b,c):
    return a + b + c

>>> exemplo(5,1,3)
9
```

Formatando a Saída

```
>>> print "numeros: %d e %05d" % (1,2)
numeros: 1 e 00002

>>> print "Linguagem: %s" % 'Python'
Linguagem: Python
```

Importante!

import modulo

from modulo import *

dir(modulo) - mostra os métodos de um módulo

dir() - mostra os módulos carregados

help(modulo) - mostra a documentação do módulo

Lendo um Arquivo

```
arq = open('teste.txt', 'r')
for linha in arq.readlines():
    print linha
arq.close()
```

Gravando em um Arquivo

```
arq = open('teste.txt', 'w') #  ** 'a','r+','w+','a+'
arq.write('linha1\n')
arq.write('linha2\n')
arq.close()
```

Dicionários

- Formados por pares de chave-valor
  - Chave sempre um valor Imutável!
- Delimitados por chaves. ‘{‘ e ‘}’

```
>>> estoque = {‘peras’: 5, ‘laranjas’: 2}
>>> estoque[‘peras’]
5
>>> estoque[‘peras’] = 4
>>> estoque[‘peras’]
4
```

Dicionários

```
>>> estoque[‘macas’] = 2
>>> estoque.get(‘melao’,’nao temos’)
‘não temos’
>>> estoque
{‘macas’: 3, ‘laranjas’: 2, ‘peras’: 5}
```

Dicionários

Principais Métodos:

copy, get, has_key, items, keys, update, values

```
estoque.has_key('uvas')
False

estoque.items()
[('macas', 3), ('laranjas', 2), ('peras', 5)]
```

```
estados = {‘RS’: ‘Rio Grande do Sul’, ‘SC’:
   ‘Santa Catarina’, ....}

def imprime_estados():
   for uf, des in estados.items():
        print ‘Estado: %s - %s’ % (uf, des)
```

Programa

- Pedir Arquivo
- Ler Arquivo
- Contar quantas vezes aparece cada palavra
- Listar as palavras em ordem alfabética, junto com o seu número de repetições no texto

Contador de Palavras

```
palavra_count = {}

arqnom = raw_input('Digite o nome do Arquivo: ')
arq = open(arqnom, 'r')

for linha in arq.readlines():
    palavras = linha.split()
    for palavra in palavras:
        palavra_count[palavra] =
                    palavra_count.get(palavra,0) + 1
arq.close()
```

Contador de Palavras - Continuação

```
palavras = palavra_count.keys()
palavras.sort()
for palavra in palavras:
    print 'Palavra: %s, Quantidade: %05i' %
   (palavra, palavra_count[palavra])
```

Funções

```
def exemplo(a,b=5,c=0):
    return a + b + c
```

```
>>> exemplo(1,2,3)
6

>>> exemplo(5)
10
```

```
>>> exemplo(5,c=3)
13

>>> exemplo(5,2)
7
```

Exceções

```
try:
    arq=open(‘teste.txt','r')
    print arq.readlines()
    arq.close()
except:
    print ‘Erro lendo arquivo teste.txt'
```

Exceções

```
if numero < 10:
    raise “Numero menor que zero”
```

- O except vefica a exceção com ‘is’, por isso strings com o mesmo ‘conteúdo’ não retornam True
- Como solucionar?

Exceções

```
EXCECAO_NUM = “Numero menor que zero”
def verifica(numero):
    if numero < 10:
        raise EXCECAO_NUM

try:
    verifica(numero)
except EXCECAO_NUM:
    print “Digite um numero >= que 10!”
except:
    print “Erro ao validar numero!”
```

Exceções

```
try:
   arq = open('teste.txt', 'r')
except:
   print "Erro ao ler arquivo"
else:
   # Será executado se qnd não houver exceção
   le_arquivo()
```

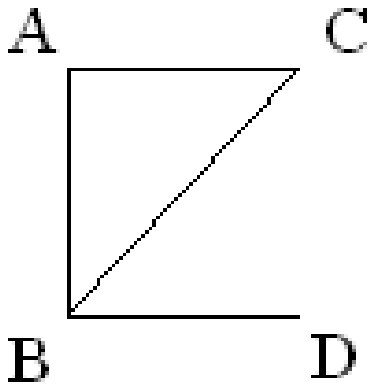
Exceções

```
try:
    algumaCoisa()
finally:
    # Será executado sempre! (com ou sem exceção)
    outraCoisa()

# Podemos usar except ou finally, nunca os dois!
```

Grafos com Dicionários

- Podemos utilizar dicionários para representar diversos tipos de dados, inclusive grafos!



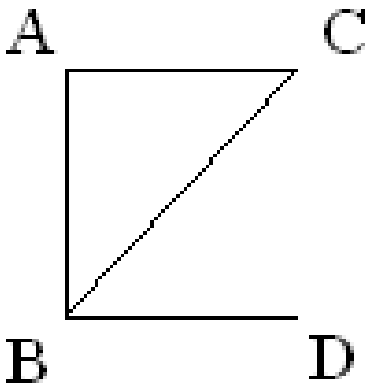
grafo = {‘A’: [‘B’,’C’], ‘B’: [‘A’,’C’,’D’], ‘C’: [‘A’,’B’], ‘D’: [‘B’]}

Grafos com Dicionários

```
def ache_caminho(grafo, inicio, final, caminho=[]):
    caminho = caminho + [inicio]
    if inicio == final:
        return caminho
    for nodo in grafo[inicio]:
        if nodo not in caminho:
            novocaminho = ache_caminho(grafo, nodo, final, caminho)
            if novocaminho:
                return novocaminho
    return None
```

Grafos com Dicionários

```
>>> ache_caminho(grafo, 'A', 'D')
['A', 'B', 'D']
>>> ache_caminho(grafo, 'A', 'D')
['A', 'B', 'D']
>>> ache_caminho(grafo, 'D', 'C')
['D', 'B', 'A', 'C']
```



Classes

```
class A:
    atributo1 = 'atributo1 da classe A'
    atributo2 = 'atributo2 da classe A'

    def __init__(self, val_ini=1):
        "Construtor da classe A"
        self.atributo_de_instacia = val_ini

    def metodo(self):
        print self.atributo_de_instacia
        print A.atributo1
```

Classes

```
>>> import classe1
>>> classe1.A.atributo1
'atributo1 da classe A'
>>> classe1.A.atributo_de_instacia
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
AttributeError: class A has no attribute 'atributo_de_instacia‘
>>> classe1.A.atributo2
'atributo2 da classe A'
```

Classes

```
>>> x = classe1.A()
>>> x.atributo_de_instacia
1
>>> x.outro_atributo = 3
>>> x.outro_atributo
3
>>> x.metodo()
1
'atributo1 da classe A'
```

Classes

```
class A:
    atributo = 'atributo da classe‘

    def metodo_statico():
        print A.atributo
    metodo_statico = staticmethod(metodo_statico)
```

Classes

```
>>> import classe2
>>> classe2.A.metodo_statico()
‘atributo da classe‘
>>> x = classe2.A()
>>> x.metodo_statico()
‘atributo da classe‘
```

Classes

hasattr(objeto, nome)

delattr(objeto, nome)

getattr(objeto, nome [, default])

setattr(objeto, nome, valor)

Mais sobre Funções

```
def teste(a,b=0,*c):
    print a,b,c
```

```
>>> teste(1)
1 0 ()
>>> teste(1,2)
1 2 ()
```

```
>>> teste(1,2,3)
1 2 (3,)
>>> teste(1,2,3,4)
1 2 (3, 4)
```

Mais sobre Funções

```
def teste(**d):
    print d
```

```
>>> teste()
{}
>>> teste(a=1,b=2)
{'a': 1, 'b': 2}
```

Mais Funções (Builtins)

abs, divmod, pow, round

int, long, float, oct, hex, ord, chr, repr, str

cmp, len, id, input, type

```
>>> dir(__builtins__)
```

Threads

```
import thread
import time

def t1(a,b,c):
    t1id = thread.get_ident()
    for i in range(10):
        time.sleep(1)
        print t1id, i

thread.start_new_thread(t1, (1,2,3))
```

Threads

```
var_lock = thread.allocate_lock()
var_lock.acquire()
var_lock.release()
```

Threading

```
import threading
t1 = threading.Thread(target=funcao,args=())
t1.start()

sem = threading.Semaphore(2)
sem.acquire()
sem.release()
```

Sockets

- Comunicação entre processos
- Cliente
  - Endereço e porta de conexão
- Servidor
  - Endereço e porta de ‘listen’
- Comunicação
  - read()
  - write()
- socket(tipo end. e protocolo, tipo socket)

Sockets - Cliente

```
from socket import socket, AF_INET, SOCK_STREAM

HOST = 'localhost'
PORT = 2223
s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT))
s.send('Mensagem do Cliente!')
data = s.recv(1024)
print data
s.close()
```

Sockets - Servidor

```
from socket import socket, AF_INET, SOCK_STREAM

HOST = ''
PORT = 2223
s = socket(AF_INET, SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1) # Numero de Conexoes
conn, addr = s.accept()

data = conn.recv(1024)
print data
conn.send('Mensagem do Servidor!')
conn.close()
```

DBM

- anydbm
  - dumbdbm (lenta e limitada, todas plataformas)
  - dbm (somente em UNIX)
  - gdbm (somente em UNIX)
  - dbhash (biblioteca BSD, em UNIX e Windows)

DBM

```
import anydbm

dados = anydbm.open('dados.dat','c')
dados['she'] = 'ela'
dados['he'] = 'ele'
dados.close()
```

first, next, previous, last, has_key, keys

Bancos de Dados

- Os bancos possuem uma API padrão
- Conexão
- Cursor
  - Comandos SQL
- Commit

Sqlite

```
import sqlite

banco = sqlite.connect('test.db')
cur = banco.cursor()
cur.execute('create table empresa (cod numeric not null,
                    des character not null, primary key (cod))')
cur.execute('insert into empresa values(1,“empresa
          teste")')
banco.commit()
banco.close()
```

Sqlite

```
cur.execute('select * from empresa order by des')

result = cur.fetchall()
for regs in result:
        print regs

cur.execute('select * from empresa where cod = %i or
   cod = %i order by des', [3,2])
```

Outros Bancos de Dados

```
import MySQLdb

con = MySQLdb.connect('servidor', 'usuario',
   'senha')
con.select_db('banco de dados')

cur = con.cursor()
```

Bancos de Dados

- Alguns Cursores permitem iteração

```
cur.execute('SQL')
for registros in cur:
    print registros
```

- Alguns Cursores retornam um uma lista de dicionários

```
registros = cur.dictfetchall()
for registro in registros:
    print registro['coluna1']
```

Interfaces Gráficas

- Tkinter – Padrão
- wxPython (antiga wxWindows)
  - GTK no Linux
  - MFC no Windows
- pyGTK
- pyQT (Windows requer Licença)
- pyFLTK, FxPy, Anygui

Tkinter


```
import Tkinter

tela = Tkinter.Tk()
tela.title('Hello')

edit = Tkinter.Entry(tela)
label= Tkinter.Label(tela, text='Tkinter!!!!')

label.pack()
edit.pack()

tela.mainloop()
```

wxPython


```
from wxPython.wx import *

class App(wxApp):
    def OnInit(self):
        frame = wxFrame(id=wxNewId(), name='frame', parent=None,
              pos=wxPoint(0, 0), size=wxSize(200, 200),
              style=wxDEFAULT_FRAME_STYLE, title='Hello World')
        frame.Show()
        return True

app = App(0)
app.MainLoop()
```

wxPython

wxFrame

EVT_CLOSE(func)

wxPanel

wxStaticText

wxTextCtrl

wxButton

EVT_BUTTON(id, func)

(parent, id, ...)

wxPython


Hello World
Digite
Informe um Valor
Ok


Mensagem
Obrigado por Pressionar!
OK

Prog_ini.py

```
from wxPython.wx import *
import Tela

class App(wxApp):
    def OnInit(self):
        tela = Tela.create(None)
        self.SetTopWindow(tela)
        tela.Show()
        return True

app = App(0)
app.MainLoop()
```

Tela.py

```
from wxPython.wx import *

def create(parent):
    return Tela(parent)

class Tela(wxFrame):
    def __init__(self, parent):
        wxFrame.__init__(self, id=wxNewId(), parent=parent,
              pos=wxPoint(0, 0), size=wxSize(200, 120),
              style=wxDEFAULT_FRAME_STYLE, title='Hello
    World')
```

Tela.py - Continuação

```
    panel = wxPanel(self, -1)

    B1 = wxButton(panel, 20, 'Ok', wxPoint(75, 40), wxSize(50,
45))
    EVT_BUTTON(panel, 20, self.OnClickB1)
    B1.SetToolTipString('Pressione!')

    L1 = wxStaticText(panel, -1, 'Digite', wxPoint(10,10))
    E1 = wxTextCtrl(panel, -1, 'Informe um Valor', wxPoint(50,7),
wxSize(100,20))

def OnClickB1(self, event):
    wxMessageBox('Obrigado por Pressionar!', 'Mensagem',
wxOK)
```

Onde buscar mais informações?

http://www.python.org

http://www.pythonbrasil.com.br

http://www.pythonology.com/

http://www.tchezope.org

http://www.google.com (Como sempre!)

OBRIGADO!

Alex Augusto da Luz dos Santos – nowayx@terra.com.br