Minicurso Python

Elton Luís Minetto

IV Jornada de Ciência e Tecnologia - JOCETEC

Histórico

- Criada em 1991 por Guido van Rossum
- Ensino de Programação
- Simplicidade e fácil aprendizado
- Como se pronuncia? Deve-se dizer "Páifon",lembrando que "th" pronuncia-se colocando a língua entre os dentes e emitindo um som de "f". "Páiton" também é uma pronuncia aceitável.
- A origem do nome: Monty Python Flying Circus
http://pt.wikipedia.org/wiki/Monty_Python

Características

- **Interpretada**

Isso significa que não é necessária a compilação do código para que ele seja executado e isso trás várias vantagens e desvantagens embutidas.

Linguagens compiladas normalmente são mais rápidas, porque o código já está num formato que o computador entende. Linguagens interpretadas costumam funcionar de uma ou outra maneira (Python usa as duas):

- **Compilação Just-In-Time**
- **Interpretação pura ou em Bytecode**

Características

- **Fortemente tipada**

Isso significa que se uma variável adquire um determinado tipo não deixa mais de ser daquele tipo a menos que seja recriada. Isso o torna diferente de PHP, por exemplo, em que nunca se sabe o tipo exato de uma variável.
Apesar da sua tipagem ser forte, a declaração de variáveis não é necessária e a simples atribuição de um valor serve para criar ou recriar uma variável.
As conversões devem ser feitas explicitamente.

Características

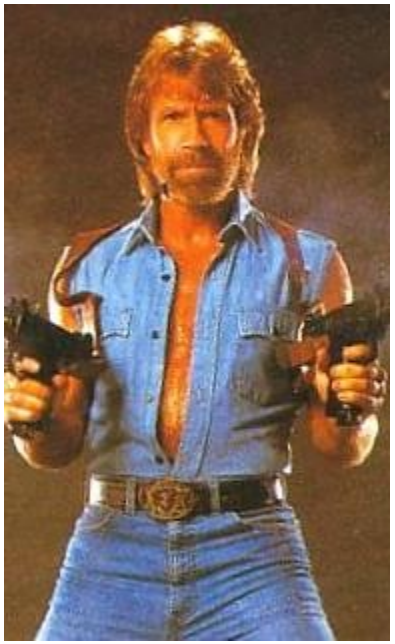
- Licença GPL-compatível
- Orientada a Objetos - Tudo é objeto
- Poderosas estruturas de dados nativas
  - Listas
  - Dicionários
- Documentação permanece com o código
- Suporte a outros paradigmas
  - Estrutural
  - Funcional
- Multiplataforma
  - Windows, Unix/Linux, MacOS, PalmOS
- Fácil integração com outras linguagens

Quem usa Python

- Industrial Light & Magic
- NASA
- Google (contratou recentemente o Guido)
- Microsoft (IronPython)
- Nokia (Serie 60, Maemo)
- Governo Brasileiro (Zope, Plone)
- Canonical (desenvolvimento do Ubuntu)
- Universidades (USP, UFSC)
- Mais exemplos :

http://www.python.org/about/success/

Quem usa Python



Vai contrariar?

Mão na massa!

- O interpretador Python

No Linux é só executar o comando python em um terminal.

No windows pode ser usado o IDLE que acompanha o Python e que pode ser usado também como IDE.

Outras IDEs para Python:

http://pythonbrasil.com.br/moin.cgi/IdesPython

Mão na massa!

- Hello World em Python

```
>>>print 'Hello World!`
```

- Hello World em Java:

```
public class hello {
        public static void main(String args[]) {
                System.out.println("Hello World!");
        }
}
```

(Python é amigo dos seus músculos)

Mão na massa!

- 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

- num_int = 13
- num_int_long = 13L
- num_real = 13.0

Strings

- >>> print 'python ' + ' powered'

python powered

- >>>print 'python' * 3

pythonpythonpython

- >>>print 'python'[0]

p

- >>> print 'python'[4:]

on

- >>>print 'python'[-2:]

on

- >>>print 'python'[2:4]

th

Strings - Principais Métodos

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

Exemplo:

```
>>>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

Exemplo:

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

Listas

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

Exemplo:

```
>>> 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

Exemplos:

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

Listas – Mais exemplos

```
>>> 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
```

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)]
```

Entrada de Dados

```
>>>nome = raw_input(‘digite o seu nome: ‘)

>>>idade = int(raw_input(‘digite a sua idade: ‘))
>>>nota = float(raw_input(‘digite a nota: ‘))
```

Condições

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

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

Condições - Exemplos

```
>>> var = 10
>>> if var == 10:
...     print "é 10"
...
é 10
>>> a = 2
>>> b = 6
>>> if var == 10 and a == 2:
...     print "aaa"
... elif b == 6:
...     print "bbb"
... else:
...     print "ccc"
...
aaa
```

Condições - Exemplos

Expressões :

var1 == var2

var1 != var2
>, <, >=, <=, isinstance(obj, class), is

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

var1 not in var2

Repetição

```
>>> for num in range(20):
...     print num
...

>>> for letra in 'python':
...     print letra
...
```

Repetição

```
while exp:
    #comandos
    if exp:
        break
    elif exp:
        continue
    #comandos
print "fora do while"
```

Funções

```
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

estados = {'RS': 'Rio Grande do Sul', 'SC':
'Santa Catarina', ....}
def imprime_estados():
    for uf, des in estados.items():
        print 'Estado: %s - %s' % (uf, des)
```

Módulos e pacotes

Um módulo é um conjunto de funções, classes ou variáveis. Para importar usa-se o comando *import*:

```
>>import modulo
```

ou

```
>>from modulo import nome
```

O que difere de uma forma para outra é o acesso ao conteúdo do módulo. Na primeira o conteúdo é acessado usando-se: modulo.conteudo. Na segunda forma não é necessário usar o modulo. para acessar.

Módulos - exemplos

```
>>>import sys
>>>print sys.path

>>>import sys as t
>>>print t.path

>>>from sys import path
>>>print path

>>>from sys import *
>>>print path
>>>print version
```

Módulos

dir(modulo) - mostra os métodos de um módulo

Exemplo: dir('a')

dir() - mostra os módulos carregados

help(modulo) - mostra a documentação do módulo

Exemplos práticos

Exemplos de aplicações

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()
```

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

Exceções

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

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!
```

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_instancia
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_instancia
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

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)
```

OBS: c é uma tupla

Mais sobre funções

```
def teste(**d):
    print d
>>> teste()
{}
>>> teste(a=1,b=2)
{'a': 1, 'b': 2}
```

OBS: d é um dicionario

Mais funções - builtins

abs, divmod, pow, round
int, long, float, oct, hex, ord, chr, repr, str
cmp, len, id, input, type

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

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()
```

Glade

O Glade é uma ferramenta para geração de interfaces em GTK. Estas interfaces podem ser utilizadas em várias linguagens como C, Java, PHP e Python. Em Python é necessário instalar os módulos pygtk e pyglade

Acesso a bancos de dados

O Python não possui acesso nativo a bancos de dados SQL em sua biblioteca padrão, mas define uma API padrão que os drivers de acesso a esses bancos de dados devem seguir, assim qualquer que seja o banco de dados os procedimentos são parecidos.

Acesso a bancos de dados

```
import MySQLdb
con = MySQLdb.connect('localhost', 'root', '')
con.select_db('financas')
cursor = con.cursor()
cursor.execute('select nome_usu,email_usu
from usuario')
for linha in cursor:
    print linha
cursor.close()
con.close()
```

Acesso a bancos de dados

```
import cx_Oracle
uid = "user"    # usuário
pwd = "senha"   # senha
db = "db_conf"
connection = cx_Oracle.connect(uid+"/"+pwd+"@"+db)
cursor = connection.cursor()
cursor.execute("SELECT * from tab")
result = cursor.fetchone()
if result == None:
        print "Nenhum Resultado"
        exit
else:
        while result:
                print result[0]
                result = cursor.fetchone()
cursor.close()
connection.close()
```

Debug

```
# Importando o Modulo
import pdb
# definindo o rastreamento
pdb.set_trace()
# Definindo as variaveis
nome      = "Elton"
sobrenome  = "Minetto"
# concatenando as variaveis
junto   = "Meu nome eh %s %s"
%(nome,sobrenome)
# Imprimindo a variavel
print junto
```

Obs: Salvar o arquivo com exemplo1.py

Debug

Executando com python exemplo1.py ele irá parar na linha dbb.set_trace()
Agoda pode-se executar os comandos do debug:
n (para executar a próxima linha)
l (listar o código)
p nome (mostra o valor da variável nome)
c (continua a execução até o fim ou até um erro)
q (sai do programa)

Exercícios

1) Faça um Programa que peça um número e então mostre a mensagem O número informado foi [número].
2) Faça um Programa que peça dois números e imprima a soma.
3) Faça um Programa que verifique se uma letra digitada é vogal.

Exercícios

4) Faça um programa para a leitura de duas notas parciais de um aluno. O programa deve calcular a média alcançada por aluno e apresentar:
* A mensagem "Aprovado", se a média alcançada for maior ou igual a sete;
* A mensagem "Reprovado", se a média for menor do que sete;
* A mensagem "Aprovado com Distinção", se a média for igual a dez.

Exercícios

5) Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.
6) Crie uma classe chamada ContaEspecial que é uma subclasse da classe Conta. Esta classe deve aceitar como parâmetro o limite da conta.

Referências

- Mini-Curso de Python. Alex Augusto da Luz dos Santos
- Curso de Python. Gustavo Noronha Silva
- Python – Guia de Consulta Rápida. Marco Catunda
- Mergulhando no Python. Mark Pilgrim
- Python Brasil. http://www.pythonbrasil.com.br
- Python. http://www.python.org
- Debugando o Python. Fabio Rizzo. http://www.linhadecodigo.com.br/colunas.asp?id_colunista=136

python™

Contato

Elton Luís Minetto
eminetto@gmail.com
http://www.eltonminetto.net

python™