

Python: Introdução

Álvaro Justen

a.k.a. Turicas

alvaro.justen@peta5.com.br



peta 5



pythonRio



<http://creativecommons.org/licenses/by-nc-sa/2.5/br/>

Sobre mim

- Engenharia de Telecomunicações
- UFF: CdD, PET-Tele, Meta, IF, MídiaCom, Monitoria
- Peta5:
 - Telefonia IP
 - Cidades digitais
 - TV Digital
- Software Livre
- Desenvolvedor Web



Python:
Introdução

E vocês?

- 1- Nome
- 2- Faz o quê?
- 3- Programa em quê?
- 4- Conhece Python?
- 5- Por que está aqui?



Python:
Introdução

Roteiro

- Python Tutorial:
 - 01 Whetting Your Appetite
 - 02 Using the Python Interpreter
 - 03 An Informal Introduction to Python
 - 04 More Control Flow Tools
 - 05 Data Structures
 - 06 Modules
 - 07 Input and Output
 - 08 Errors and Exceptions



Python:
Introdução

Roteiro (2)

- Python Tutorial:
 - 09 Classes
 - 10 Brief Tour of the Standard Library



Python:
Introdução

O que é Python?

- Linguagem interpretada
- VHLL: Very High Level Language
- Criada por Guido van Rossum em 1989
- Multiplataforma
- Procedural, OO e funcional
- CPython: software livre
- Baterias incluídas! (Extensa biblioteca padrão)



Python:
Introdução

O que é Python? (2)

- Nome baseado no seriado Monty Python's Flying Circus
- Usaremos Python 2.6
- Python 3.0 e a quebra de compatibilidade



Python:
Introdução

História

About Google!



Links

- [Research Papers about Google and the WebBase](#)
- [Pictures and stats for the Stanford Google Hardware](#)

Credits

- Current Development: [Sergey Brin](#), [Larry Page](#), and [Craig Silverstein](#)
- Design and Implementation Assistance: [Scott Hassan](#) and [Alan Steremberg](#)
- Faculty Guidance: [Hector Garcia-Molina](#), [Rajeev Motwani](#), [Jeffrey D. Ullman](#), and [Terry Winograd](#)
- Research Funding: [NSF](#), [NASA](#), [DARPA](#) and [Interval Research](#)
- Equipment Donations: [IBM](#), [Intel](#), and [Sun](#)
- Equipment Consulting: [Penguin Computing](#)
- Software: [GNU](#), [Linux](#), [Python](#), [Parasoft](#) (debugging), and [Gimp](#) (logo design)
- Collaborating Groups in the [Computer Science Department](#) at [Stanford University](#): [The Digital Libraries Project](#), [The Project on People Computers and Design](#), [The Database Group](#), [The Stanford InfoLab](#), [The MIDAS Data Mining Group](#), and [The Theory Division](#)
- Outside Collaborators: [Interval Research Corporation](#) and the [IBM Almaden Research Center](#)
- Technical Assistance: [The Computer Science Department's Computer Facilities Group](#), [Stanford's Distributed Computing](#) and [Intra-Networking Systems Group](#)

Last modified: Sun Sep 27 20:26:46 PDT 1998



Python:
Introdução

Características

- Sintaxe elegante, simples e clara
- Orientada a objetos (porém procedural e funcional também)
- Tipagem dinâmica
- Tipagem forte
- Estruturas de dados de alto nível: tuplas, listas e dicionários
- Blocos de código são delimitados por endentação! (?)



Python:
Introdução

Vantagens

- Fácil de aprender
- Poderosa
- Linguagem dinâmica -> quase tudo acontece em runtime!
- Code -> Test (não precisa compilar, porém Python compila automaticamente .pyc em bytecode)
- C API



Python:
Introdução

Vantagens (2)

- Facilidade de criação de módulos: comunidade pode facilmente criar e compartilhar funcionalidades
- Interpretador interativo: shell.
 - "python" (shell)
 - "python arquivo.py" (interpretador)
- `ipython -i a.py` ou `python -i a.py`
 - Executa `a.py` (`chmod +x a.py`) e retorna o shell para testes

Onde é usada?

- Empresas :
 - Google
 - NASA
 - Peta5 (. . .)
- Software :
 - Trac
 - GIMP
 - Blender
 - GNOME (. . .)



Python:
Introdução

Onde roda?

- UNIX likes:
 - GNU/Linux (maioria vem por padrão)
 - Mac OS X (sempre por padrão)
- Windows (tem que instalar)
- Nokia Series 60
- ...



Python:
Introdução

Documentação

- Faça documentação com comentários!
- Python tutorial:
<http://docs.python.org/tut>
- Thinking Like a Computer Scientist (Python)
- Learning Python
- Dive Into Python
- Computação Científica com Python



Python:
Introdução

www.python.org

- PEPs
- Python Library Reference
- Python Manual Reference
- PyPI
- CPython
- Jython
- IronPython
- PyPy



Python:
Introdução

Botando a mão na massa

- Variáveis
- Números:
 - Inteiro
 - Ponto flutuante
 - Complexo
- Strings
 - ' , " " " e \
 - Concatenação
 - Slices



Python:
Introdução

Botando a mão na massa (2)

- Listas:
 - ['a', 42, 'b', 3.14, [2009, 'c']]
 - “Ponteiros”
 - l[:]
 - Slice replace
 - Acesso: l[n], l[n:], l[:n], l[n:m]
 - Len(), l.append()



Python:
Introdução

Controle de Fluxo

`if` expressão:

 Comando1

 Comando2

 ...

`elif` expressão2:

 Comando3

 ...

`else`:

 comandoN



Python:
Introdução

Controle de Fluxo (2)

```
for i in iterable:
```

```
    Comando1
```

```
    Comando2
```

```
    ...
```

```
while expressão2:
```

```
    Comando3
```

```
    ...
```

```
iterable → list, range(), dict, ...
```

```
break, continue, pass, for ... else
```

Funções

```
def nome(args):
```

```
    Comando1
```

```
    ...
```

```
    [return X]
```

- *args
- **args
- arg1=valor, arg2=valor, ...
- lambda
- docstrings



Python:
Introdução

Listas

- `.append()`
- `.insert()`
- `.remove()`
- `.pop()`
- `.count()`
- `.index()`
- `.sort()`
- `.reverse()`



Programação Funcional

- `filter(filtro, sequência)`
 - True or False
- `map(função, sequência)`
 - Lista com return de “função”
- `reduce(função, sequência)`
 - Opera de 2 em 2
- `zip(a, b)`
 - `[(a[0], b[0]), (a[1], b[1]), ...]`
- List comprehension



Python:
Introdução

Tuplas

- (a, b, c, ...)
- Bem mais rápidas
- Imutáveis



Python:
Introdução

Dicionários

- `{'a': 1, 'c': 10, 10: 'a', (1, 2): 'z', 'Z': (1, 2)}`
- `dict(key1=value1, key2=value2, ...)`
- `d[key]`
- `key in d` → True or False
- Não acessa em ordem!
- `for k, v in d.iteritems():`
comandos...

Módulos

- `arq.py`
- `import arq`
- `arq.X`
- `arq.f()`
- `f2 = arq.f`
- `arq.__name__`
- `from arq import X, f`
- `from arq import *`



Python:
Introdução

Módulos (2)

- `import arq as abc`
- ```
if __name__ == '__main__':
 print 'Sou um programa.'
else:
 print 'Sou um módulo.'
```
- Standard modules
  - `dir(módulo)`



**Python:**  
**Introdução**

# Pacotes

sound/

\_\_init\_\_.py

formats/

\_\_init\_\_.py

wavread.py

wavwrite.py

effects/

\_\_init\_\_.py

echo.py

surround.py

filters/

\_\_init\_\_.py

equalizer.py



**Python:**  
**Introdução**

# Pacotes (2)

- `from sound.formats import wavread`
- `import sound.effects.echo`
- ...
- Não esquecer do `__init__.py`
  - Vazio
  - `__all__ = [...]`
    - `from sound.formats import *`

# Arquivos

- `open('nome', 'formato')`
  - `formato = rw, r, ...`
- `fp.`
  - `read()`
  - `readlines()`
  - `write()`
  - `writelines()`
  - `close()`
- `for l in fp: print l`



# Classes

```
class Nome:
```

```
 """Essa é a minha classe...
 que não faz muita coisa"""
```

```
 self.atributo = valor
```

```
 def método(self, args...):
```

```
 cmd...
```

-

# Classes (2)

- `__init__()`
- Overload of operators
- Métodos fora das classes:  

```
def f1(self, a, ...):
 ...
class teste:
 f = f1
 def f2(self, b, ...): ...
```
- Herança [múltipla]

# Classes (3)

- `__abc` e `_abc`
- Introspecção: `dir()`, `type()`, `locals()`, `globals()`
- Polimorfismo
- Tudo é objeto!
- Exceções são classes



**Python:**  
**Introdução**

# Baterias incluídas

- `sys`, `os`, `re`, `string`, `zlib`
- `socket`, `urllib`, `httpplib`, `imaplib`
- `time`, `datetime`, `math`, `random`
- `doctest`, `unittest`, `xml`, `xmlrpclib`
- `struct`, `pickle`, `cPickle`, `threading`
- `logging`, `decimal`
- ...



**Python:**  
**Introdução**

# Outras Bibliotecas

- NumPy
- SciPy
- Matplotlib
- PyODE
- Python-OpenGL
- . . .



**Python:**  
**Introdução**

# Aplicações

- Scripts
- Computação gráfica
- Web
- Jogos
- Programas em geral (GUI)



**Python:**  
**Introdução**

# The Zen of Python

```
>>> import this
```



**Python:**  
**Introdução**