



academia
python

Introdução à linguagem Python

Slides extras

Linguagem de uso geral

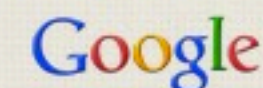
- Internet: YouTube, Globo.com, Bitly, Mozilla...
- Computação gráfica: Disney, ILM, AutoDesk...
- Desktop: Dropbox, BitTorrent, OpenOffice...
- Operações: Google, Rackspace, Ubuntu, RedHat...
- Enterprise: IBM, Oracle, ERP5, OpenERP...
- Games, computação científica, segurança, etc...

[ABOUT](#) [SCHEDULE](#) [REGISTRATION](#) [VENUE](#) [SPONSORS](#) [JOBS FAIR](#) [BLOG](#)

PYCON 2012

SANTA CLARA, CA

PRESENTED BY




Tutorials
March 7th - 8th

Talks
March 9th - 11th

Sprints
March 12th - 15th



GONDOR
one **does** simply deploy



Microsoft

Eventbrite
Events Made Easy



PLATINUM

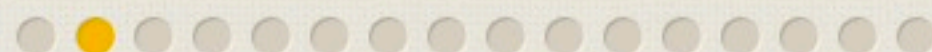
PLATINUM

PLATINUM

PLATINUM

PLATINUM

GOLD

[Learn how to become a sponsor...](#)

WHAT IS PYCON?

PyCon is the largest annual gathering for the community using and developing the open-source Python programming language. PyCon is organized by the Python community for the community. We try to keep registration far cheaper than most comparable technology conferences, to keep PyCon accessible to the widest group possible. PyCon is a [diverse conference](#) dedicated to providing an enjoyable experience to everyone. Our [code of conduct](#) is intended to help everyone maintain the PyCon spirit. We thank all attendees and staff for observing it.

SPONSORS



Sobre Python

- Linguagem dinâmica
 - compilador é parte do ambiente de runtime
- Tudo são objetos
 - Ex.: inteiros, funções, classes, exceções etc.
- Multi-plataforma
 - Interpretador, APIs, bytecode, GUIs etc.

Implementações

- CPython: a principal, escrita em C
 - incluída na maioria das distros GNU Linux e no Mac OS X; vários pacotes para Windows
- Jython: escrita em Java
 - parte de IBM WebSphere e Oracle WebLogic
- IronPython: escrita em C#, .net CLR
 - desenvolvida pela Microsoft

Implementações (2)



- PyPy: Python em Python
- Implementação em linguagem de alto-nível facilitando a inovação
- Desempenho 5x melhor que CPython
- Incompatível com as extensões em C



Changes

rotated_iterator	0.0311	2.12%	-7
co	0.3336	-3.30%	-3
mpg	0.2642	0.90%	-17
stringio	4.4612	6.13%	5
test_pb	0.0325	-1.39%	-2
test_names	0.0063	-2.00%	-2
testlib	10.1920	2.73%	2

Track performance changes in the latest revisions

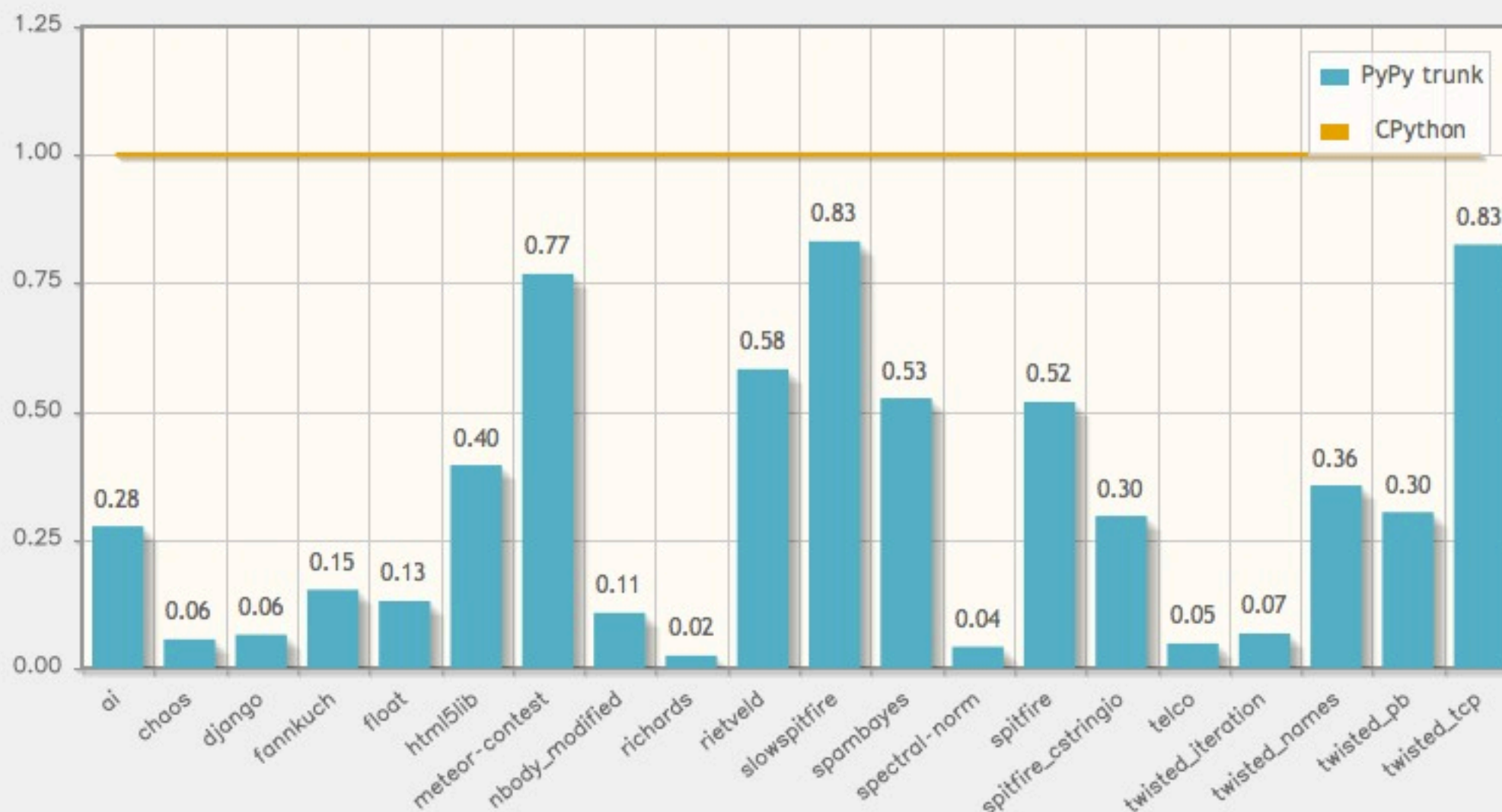
Timeline



Comparison

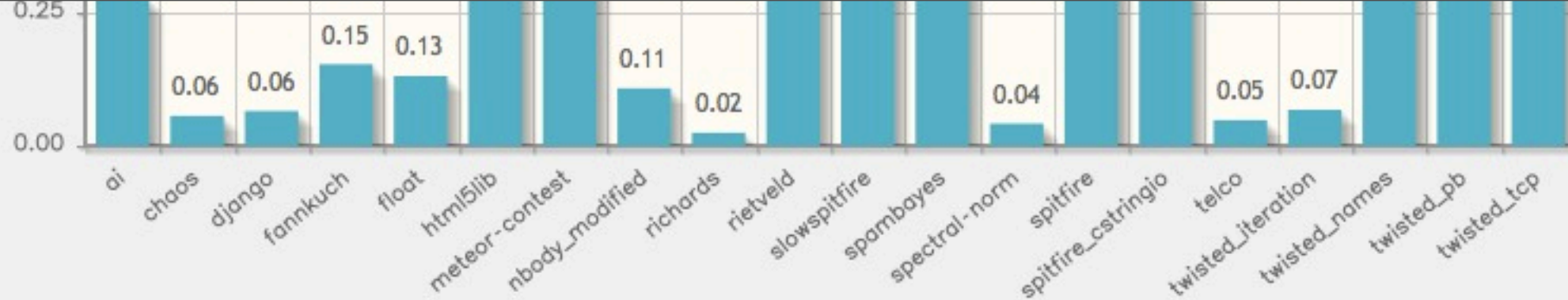


How fast is PyPy?



Plot 1: The above plot represents PyPy trunk (with JIT) benchmark times normalized to CPython. Smaller is better.

It depends greatly on the type of task being performed. The geometric average of all benchmarks is 0.20 or **5.1 times faster** than CPython.



Plot 1: The above plot represents PyPy trunk (with JIT) benchmark times normalized to CPython. Smaller is better.

It depends greatly on the type of task being performed. The geometric average of all benchmarks is 0.20 or **5.1 times faster** than CPython

How has PyPy performance evolved over time?



Plot 2: Geometric averages of normalized times, out of 20 benchmarks. Smaller is better. "times faster" inside parenthesis

Implementações (3)

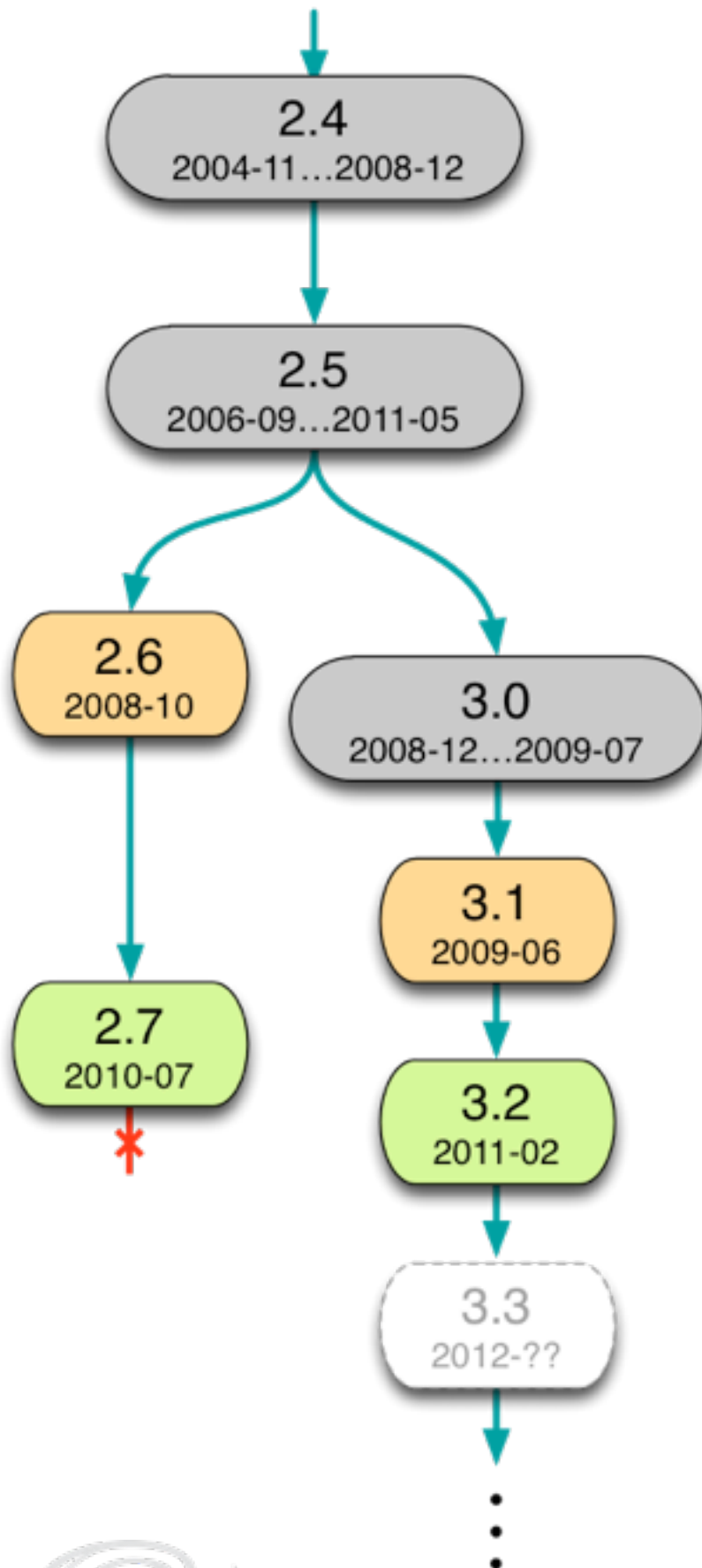
3.2	✓			
2.7	✓		✓	✓
2.5		✓		
	CPython	Jython	Iron Python	PyPy



nosso
foco

✓ = desenvolvimento ativo

Python: evolução



- Versões do CPython em jan/2012
- 3.0, 2.5 e anteriores: fossilizadas
- 2.6 e 3.1: manutenção ativa
- 2.7 e 3.2: versões atuais
- 2.7: versão final da série 2.x
- 3.3: prevista para ago/2012

CPython 2.7: instaladores

- GNU Linux: pré-instalado em quase toda distro
 - usar gerenciador de pacotes ou compilar (é fácil)
- Windows
 - Python.org: instaladores MSI para 32 e 64 bits
 - ActiveState ActivePython: 32 e 64 bits
- Mac OS: 32 bits (x86 e PPC), 64 bits (x86)

Módulo I: temas finais

- Manipulação de argumentos da linha de comando
- Leitura e gravação de arquivos:
 - binários e texto com encoding
- Introdução a testes automatizados com doctest
- Consoles interativos
- IDEs e editores populares com Python

Argumentos da linha de comando

- Uma lista de strings disponíveis através de `sys.argv`
 - `sys.argv[0]` é o nome do próprio script
- Para resultados profissionais, use o módulo `argparse`, incluído no Python 2.7
 - disponível no PyPI para Python ≥ 2.3
 - PyPI = Python Package Index

somar_args.py

```
import argparse
```

```
def media(numeros):  
    return sum(numeros, 0.0) / len(numeros)
```

```
parser = argparse.ArgumentParser(description='Somar números.')
```

```
parser.add_argument('numeros', metavar='N', type=float, nargs='+',  
                    help='números a somar')
```

```
parser.add_argument('-m', dest='operacao', action='store_const',  
                    const=media, default=sum,  
                    help='calcular a média (default: somar)')
```

```
args = parser.parse_args()  
print args.operacao(args.numeros)
```

```
import argparse
```

```
def media(numeros):
```

```
    return sum(numeros, 0.0) / len(numeros)
```

```
parser = argparse.ArgumentParser(description='Somar números.')
```

```
parser.add_argument('numeros', metavar='N', type=float, nargs='+',  
                    help='números a somar')
```

```
parser.add_argument('-m', dest='operacao', action='store_const',  
                    const=media, default=sum,  
                    help='calcular a média (default: somar)')
```

```
args = parser.parse_args()
```

```
print args.operacao(args.numeros)
```

```
$ ./somar_args.py
usage: somar_args.py [-h] [-m] N [N ...]
somar_args.py: error: too few arguments$
$ ./somar_args.py -h
usage: somar_args.py [-h] [-m] N [N ...]

Somar números.

positional arguments:
N                      números a somar

optional arguments:
-h, --help            show this help message and exit
-m                    calcular a média (default: somar)
```


Leitura e gravação de arquivos

- Função embutida `open()`
 - Capítulo 7 do tutorial
- `open()` devolve uma instância de `file()`
- `file.read()` devolve uma string de bytes
 - o decoding é problema seu

Leitura linha a linha

```
#coding: utf-8

import sys

nome_arq = sys.argv[1]

arq = open(nome_arq)
print '='*60
for lin in arq:
    print lin.rstrip()
arq.close()
```

```
#coding: utf-8

import sys

nome_arq = sys.argv[1]

with open(nome_arq) as arq:
    print '='*60
    for lin in arq:
        print lin.rstrip()
```

Leitura e gravação de textos com encoding

- Função `codecs.open()`
 - módulo `codecs`
 - aceita um argumento para definir o encoding
- `file.read()` devolve uma string de unicode
 - o decoding deixa de ser problema seu!

Escrevendo com codecs.open

```
#!/usr/bin/env python  
# coding: utf-8
```

```
import sys  
import codecs
```

```
uni = u'avião'
```

```
encodings = ['cp1252', 'utf-8']
```

```
for encoding in encodings:  
    nome_arq = 'aviao-%s.dat' % encoding  
    with codecs.open(nome_arq, 'wb', encoding) as saida:  
        saida.write(uni)
```

Exemplo de doctest

```
# coding: utf-8  
"""
```

Calcula a média de uma sequência de números

```
>>> media([10])  
10.0  
>>> media([10, 20])  
15.0  
>>> media([1, 2])  
1.5
```

```
"""
```

```
def media(seq):  
    return float(sum(seq))/len(seq)
```

Para executar doctests

- Pela linha de comando:
 - `$ python -m doctest meu_script.py`
- Usando um test-runner (unittest, nose, etc.)
 - veremos isto no módulo 2
- No próprio script (self-test):

```
if __name__ == '__main__':  
    import doctest  
    print doctest.testmod(optionflags=doctest.REPORT_ONLY_FIRST_FAILURE  
                             |doctest.REPORT_NDIFF  
                             |doctest.NORMALIZE_WHITESPACE)
```


doctest: diretivas + úteis

- `NORMALIZE_WHITESPACE`
 - útil para uso em todos os testes
- `SKIP`
 - pular um teste específico
- `ELLIPSIS`
 - ignorar parte de um output muito extenso ou irrelevante

Consoles interativos

- python
 - integrado em IDEs
 - iPython
 - bpython
 - IDLE
- online
 - <http://shell.appspot.com/>
 - <http://pythonwebconsole.thomnichols.org/>
 - <http://www.pythonanywhere.com/>

IDEs, algumas opções

- PyDev (Eclipse)
- PyCharm (JetBrains) \$\$\$
- Komodo Edit
- Komodo IDE \$\$\$
- TextMate \$ - OSX
- SublimeText \$
- WingIDE \$\$\$

Em vez de IDE: The Unix environment

- Linux ou OSX ou qualquer Unix:
 - janelas de editor, console e navegador ou sua aplicação
 - alternar entre janelas com alt-tab: simples, poderoso e eficaz
- Emacs: python-mode.el
- vi: http://www.vex.net/~x/python_and_vim.html
- Geany: um Gedit mais esperto para lidar com vários arquivos

Configure o editor para...

- Indentar com 4 caracteres de espaço ao usar a tecla TAB ou comandos de indentação multi-linha
- Salvar tabs como 4 espaços, nunca como tabs
- Limpar brancos no final das linhas ao salvar
- Indentação inteligente:
 - preservar indentação da linha acima
 - indentar automaticamente após: if, elif, else, for, while, try, except, finally, def, class, with