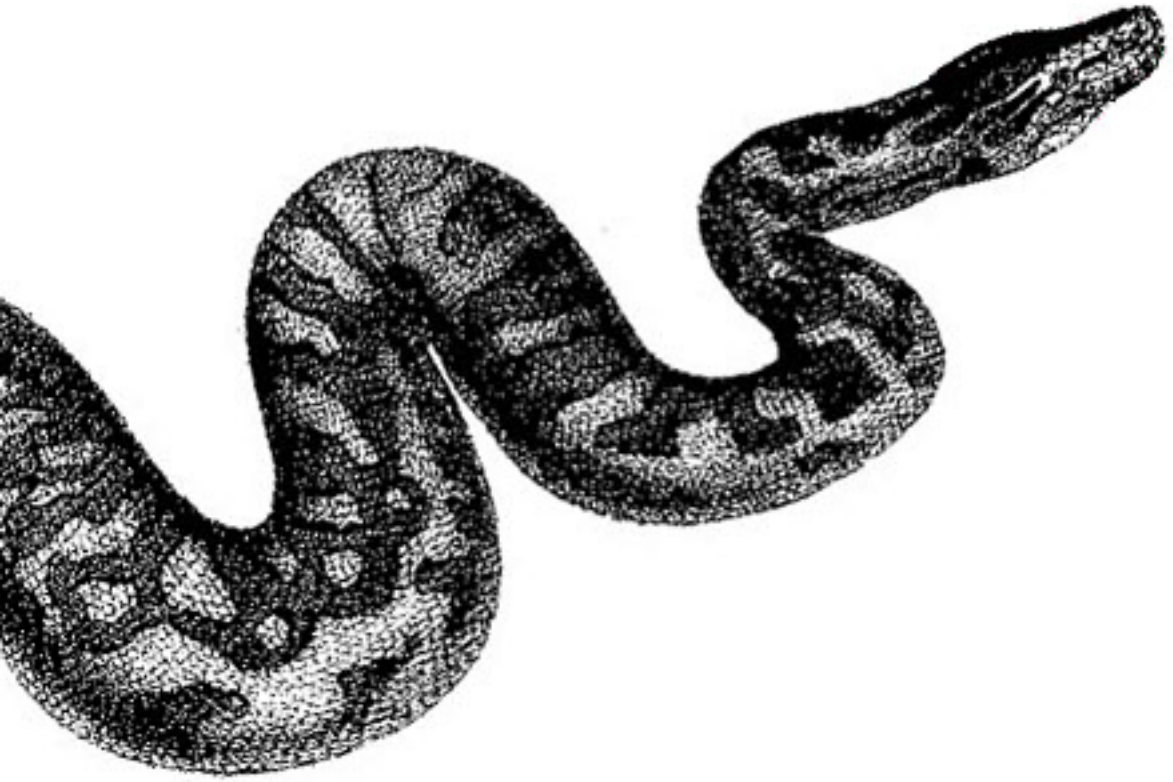
Python:
tipos de dados básicos



Luciano Ramalho
ramalho@python.pro.br

PythonPro

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...

PythonPro

PYCON 2012

SANTA CLARA, CA

PRESENTED BY

Google

heroku

Dropbox

Tutorials
March 7th - 8th

Talks
March 9th - 11th

Sprints
March 12th - 15th

facebook	GONDOR one does simply deploy	nasuni	Microsoft	Eventbrite Events Made Easy	loggly
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


Dropbox
heroku
Google
New Relic.
SurveyMonkey
nebula
facebook
GONDOR
one does simply deploy
nasuni
Microsoft
Eventbrite
Events Made Easy
loggly
OpDemand
CCP
WhiteOak
TECHNOLOGIES, INC.
J.P.Morgan
ENTHOUGHT
ActiveState
The Dynamic Language Experts
PyCharm
by JetBrains
caktus
GROUP
PiCloud
linode
CONTINUUM
ANALYTICS
OPENSHIFT
PaaS by Red Hat Cloud
NAO
DEVELOPER
DISQUS
Leapfrog
Spotify
snoball
Evite
REVSYS
revolution systems
PlaidCloud
CANONICAL
Milo
local shopping
splunk>
mozilla
Lab305
Walt Disney
ANIMATION STUDIOS
ZeOmega
CLOUD
FOUNDRY
bitly
dotCloud
Addison Wesley
PRENTICE HALL
informIT.com
COXMEDIA
GROUP
EUCALYPTUS
WINGWARE
PYTHON IDE
WISER
together
esri
aginteractive
AMERICAN GREETINGS
Vocollect
a business unit of Honeywell
10gen
mongoDB
BIGDOOR
THE OPEN
BASTION
Toast
Driven
net-ng
myYearbook
threadless
gocept
LEX MACHINA
Aarki
FRESHBOOKS
painless billing
github
SOCIAL CODING
fwix
Imaginary Landscape
openstack
CLOUD SOFTWARE
devsar
emma
communicate in style
SHINING
PANDA
Python
Academy
Training & Consulting
CISCO
Kontagent
Truveris
Stratasan
CREATIVE
TECH
EnergySavvy
Atlassian
bitbucket
OPSCODE
@mailgun
Urban Airship
MapMyFITNESS
MNX
SOLUTIONS
Linux Server Management
DataStax
CyberWeb
Consulting
O'REILLY
olark
reddit
cars.com
Confidence Comes Standard.
DreamHost
IP[y]:
IPython
Interactive Computing
pyladies
Pylons
OSL
OPEN SOURCE LAB
PostgreSQL
WOMEN WHO
CODE
openhatch

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.

PythonPro

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

PythonPro

Implementações (2)

- PyPy: Python em Python
- Implementação em linguagem de alto-nível facilitando a inovação
- Desempenho 6x melhor que CPython
- Porém... incompatível com as extensões em C (milhares de bibliotecas externas)

PythonPro

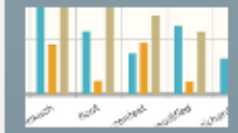
Changes

Track performance changes in the latest revisions

Timeline

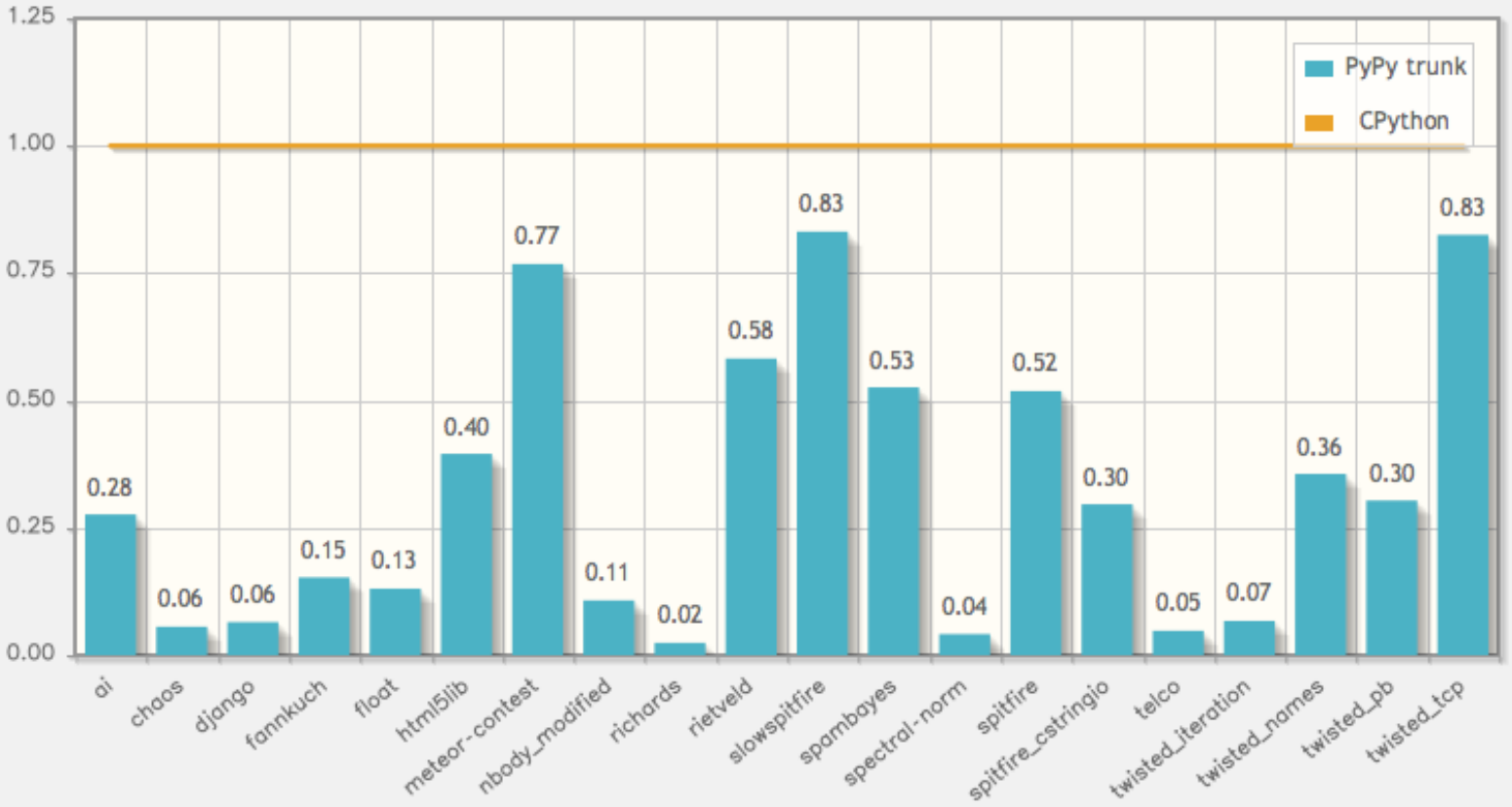
Analyze performance over time

Comparison



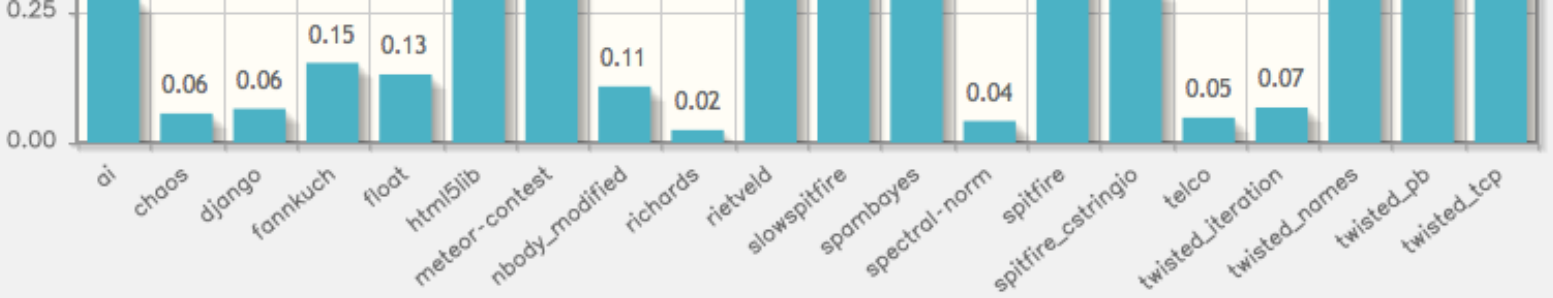
Compare different executables and revisions

How fast is PyPy?



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

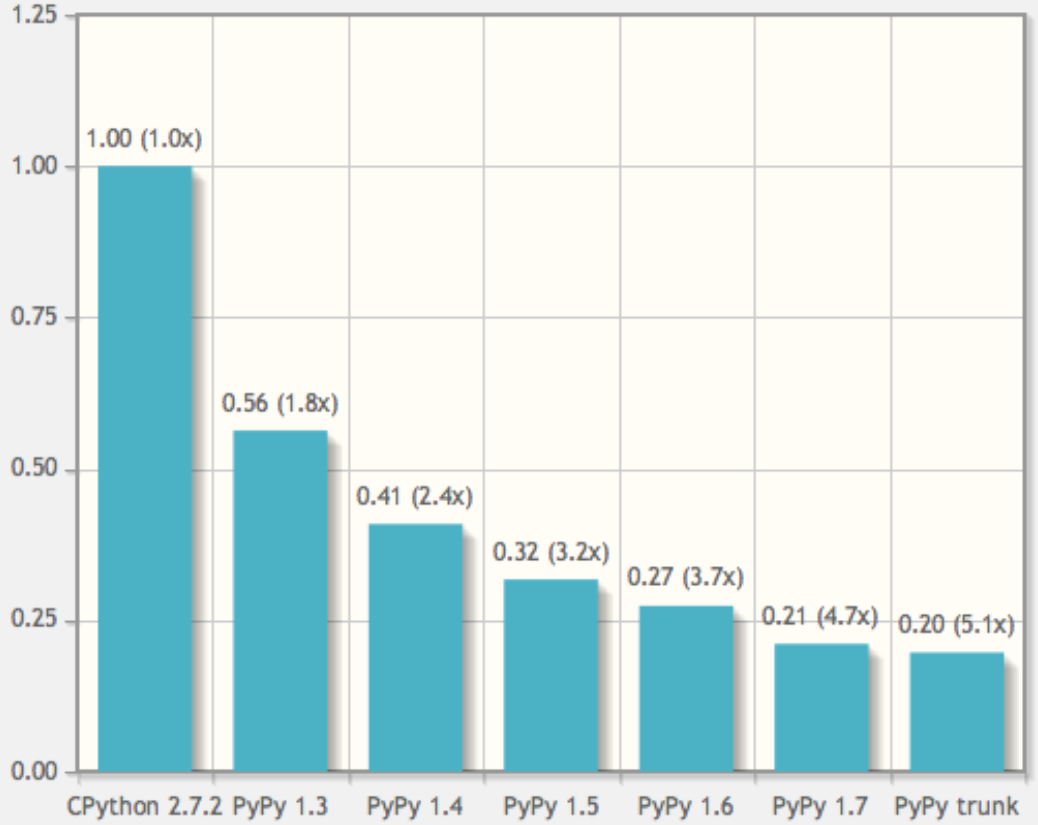
It depends greatly on the type of task being performed. The geometric average of all benchmarks is 0.20 or **5.1** times *faster*



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

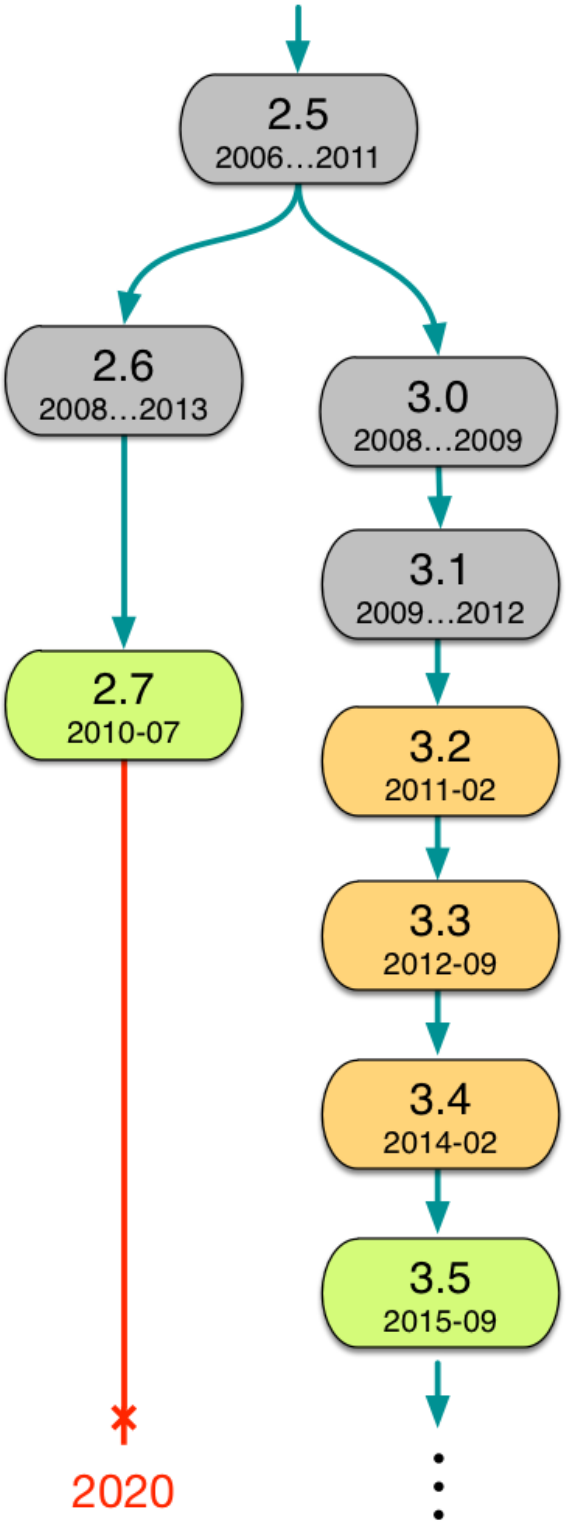
How has PyPy performance evolved over time?



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

Python: evolução

- Versões do CPython em set/2015
- 3.1, 2.6 e anteriores: fossilizadas
- 2.7: versão final da série 2.x
- 2.7, 3.2…3.4: manutenção ativa
- 2.7 e 3.5: versões atuais
- 3.6: prevista para o final de 2016 (ver atualizações em PEP-494)



PythonPro

CPython: 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
  - WinPython: muitos pacotes integrados
- Mac OS: 32 bits (x86 e PPC), 64 bits (x86)

Algumas questões práticas

- 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

PythonPro

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

PythonPro

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)
```

somar_args.py

```
$ ./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)
```

PythonPro

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

PythonPro

Leitura linha a linha

```python
#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()
```

```python
#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()
```

PythonPro

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 io.open

```
#/usr/bin/env python2.7
# coding: utf-8

import sys
import io

uni = u'avião'

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

for encoding in encodings:
    nome_arq = 'aviao-%s.dat' % encoding
    with io.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)
```

PythonPro

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):

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

PythonPro

Consoles interativos

- python
- integrado em IDEs
- iPython
- bpython
- IDLE

IDEs, algumas opções

- PyCharm (JetBrains) community/pro $$$
- PyDev (Eclipse)
- Atom
- Komodo Edit/IDE $$$
- WingIDE $$$
- Editor simples, produtivo e extensível:
    - SublimeText $

PythonPro

Em vez de IDE:
The Unix environment

- Linux ou OSX ou qualquer Unix:
    - janelas de editor, console e navegador ou sua aplicação + 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

PythonPro

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