Curso de Python em 5 Horas

Introdução à Linguagem, História e Propaganda

Gustavo Sverzut Barbieri

GPSL - UNICAMP

2 de outubro de 2006

Resumo

Esta é a aula introdutória à linguagem Python. Aborda-se um pouco da história, faz-se uma propaganda da linguagem, mostra-se comentários sobre ela e também uma explicação breve sobre a comunidade que dela cuida.

Não será abordado qualquer tipo de código ou construção sintática, apenas os aspectos filosófico e cultural que são peças importantes para um desenvolvedor Python.

Onde e Quando

- Criada em 1989 pelo holandês **Guido van Rossum** no *Centrum voor Wiskunde en Informatica* (CWI), em Amsterdã, Holanda.
- Influenciada pela linguagem ABC, desenvolvida no CWI por Guido e outros nas décadas de 70 e 80. ABC tinha um foco bem definido: ser uma linguagem de programação para usuários inteligentes de computadores que não eram programadores: Físicos, Cientistas Sociais e até Linguistas.
- O projeto de sistema operacional distribuído Amoeba precisava de uma linguagem de *script*. Nasce o Python.

Bases do Python

- Elementos que eram bem sucedidos no ABC.
- Estruturas de dados poderosas inclusas: Listas, Dicionários, *Strings*.
- Usar indentação para delimitar blocos, eliminando chaves.
- Fácil extensão (lição aprendida com os erros do ABC).
- Fácil de portar: além do Amoeba, gostaria de executar em Unix, Macintosh e Windows.
- Influências de Modula-2 e Modula-3: módulos e *namespaces*.

Ambiente de Concepção

- **Universidade:** pessoas altamente especializadas para desenvolver e opinar sobre os elementos do projeto
- **Descontraído:** o nome Python vem da série de humor *Monty Python's Flying Circus*
- **Sem prazos, Sem pressão:** o desenvolvimento não foi pressionado por estratégias de marketing, prazos, clientes ou qualquer outro fator que pudesse influenciar nas decisões de projeto, **resultando em maior qualidade.**
- **Software Livre:** garante a vida da tecnologia.

Características Básicas

- **Interpretada:** usa máquina virtual, facilita portabilidade.
- **Interativa:** pode-se programar interativamente, os comandos são executados enquanto são digitados. Facilita testes, desenvolvimento rápido e outros. Facilitadores estão presentes `help(obj)`.
- **Orientada a Objetos:** tudo[1] é objeto: números, *strings*, funções, classes, instâncias, métodos, ...
- **Fortemente Tipada:** Não existe *casts* e nem conversão automática, não se mistura tipos "automagicamente".
- **Tipagem Dinâmica:** A tipagem de um objeto é feita em tempo de execução. Um objeto tem tipo, uma variável não.

[1] Quase tudo é um objeto :-)

Características Importantes

- Sintaxe clara, sem caracteres "inúteis":
  - blocos são marcados por indentação
  - parênteses são opcionais, só precisam ser utilizados para eliminar ambiguidades.
  - palavras-chave (*keywords*) e formações que ajudam na leitura, como `for ... in ...`.
- Fácil extensão: codificar nos módulos é muito fácil, podendo utilizar bibliotecas nativas, aproveitando desempenho, características nativas das plataformas, etc.
  - API Python/C é bem simples
  - Diversos conversores automáticos (SWIG, SIP, ...)
  - **Jython:** usando Python em Java e vice-versa.
  - **PyRex:** pseudo linguagem para facilitar integração Python + C/C++.

Características Importantes (2)

- **Tipos básicos poderosos:** listas, dicionários (*hash tables*), *strings*, ... otimizados e de fácil uso.
- **Baterias Inclusas:** biblioteca padrão contém diversos recursos úteis: Interface Gráfica (Tk), XML, Servidores (TCP, UDP, HTTP, ...), HTML, protocolos de internet (email, http, ...), xmlrpc, ...
- Grande base de código e bibliotecas de terceiros
- Grande comunidade de desenvolvedores
- **Software Livre:** liberdade de uso, distribuição. Licença própria, compatível com GPL, porém pode distribuir somente o binário.
- **Independente:** a entidade sem fins lucrativos Python Software Foundation cuida da propriedade intelectual do Python.

No Brasil

- **Embratel:** monitoramento das interfaces de *backbone* e clientes de internet, também existem *scripts* de uso interno.
- **CPqD:** monitoramento de centrais telefônicas.
- **Conectiva:** Gerenciamento de pacotes da distribuição Linux e ferramentas de uso interno.
- **Async:** desenvolvimento de software de automação comercial
- **GPr Sistemas:** Desenvolvimento de aplicações sob encomenda, sistemas como monitoramento de transporte terrestre via satélite são as soluções já feitas
- **Outras** que utilizam o Python para sistemas Web, como Varig, Serpro, Câmara, Interlegis, ...

Os sistemas web de gestão de conteúdo usando o trio Python/Zope/Plone vem crescendo a cada dia, principalmente em empresas grandes e no governo.

No Mundo

- **Industrial Light & Magic:** automação interna: *"Python plays a key role in our production pipeline. Without it a project the size of Star Wars: Episode II would have been very difficult to pull off. From crowd rendering to batch processing to compositing, Python binds all things together"*
- **NASA:** repositório de CAD/CAE/PDM, gerência de modelos, integração e sistema colaborativo: *"We chose Python because it provides maximum productivity, code that's clear and easy to maintain, strong and extensive (and growing!) libraries, and excellent capabilities for integration with other applications on any platform."*
- **University of Maryland:** ensino: *"I have the students learn Python in our undergraduate and graduate Semantic Web courses. Why? Because basically there's nothing else with the flexibility and as many web libraries"*

No Mundo (2)

- **Apple:** ferramenta padrão desde o MacOS X.
- **Microsoft:** investimento no IronPython para a plataforma .NET.
- **Disney:** jogos e Sistemas internos de automação e criação, patrocínio do PyQT.
- **Bank Boston:** sistema Web usando Python e Zope.
- **Nokia:** sistema de programação para celulares da série 60, permite mais recursos que o Java.
- **Atari:** jogos, como "Temple of Elemental Evil".
- **Yahoo!:** Yahoo! Groups foi escrito inicialmente em puro python: 180.000 linhas de código cuidavam de tudo, tratando mais de 200 mensagens/segundo em um simples Pentium 400Mhz.

No Mundo (3)

- **Nortel:** sistemas web "ChartWare", "WebBook" e "WebTrack" são exemplos.
- **Philips:** automação da linha de semicondutores na fábrica de Fishkill.
- **Lawrence Livermore National Laboratories:** ambiente de engenharia numérica.
- **Red Hat:** diversas ferramentas para linux, o instalador das distribuições Red Hat e Fedora ("Anaconda").
- **Gentoo Linux:** sistema de gerência de pacotes "Portage".
- **Blender3D:** software pode ser estendido usando plugins Python.

No Mundo: Google

Maior *"case"* Python da atualidade:

"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language." — Peter Norvig, director of search quality at Google, Inc.

No Mundo: Google (2)

- Sistema de ajuda do GMail
- Google Groups
- Sistema de compilação de aplicativos (*build system*).
- Sistema de empacotamento e entrega de dados (*packaging system*).
- Sistema de monitoramento e manutenção do cluster
- Sistema de testes
- Análise de registros (*logs*), ié análise de falsos clicks no *Sponsored Ads*.
- Prototipação
- Recentemente liberando código em `http://code.google.com`, como Goopy.
- Requisito para contratar profissionais Java: saber Python! :-)

Bruce Eckel

Bruce Eckel é o autor de livros de renome, como "Thinking in Java" e "Thinking in C++", mas hoje prefere pensar em Python. Ele apresentou 10 razões por que ele ama esta linguagem, algumas delas:

- **Python is about me**
  - Outras linguagens: *"Sim, nós estamos tentando fazer sua vida mais fácil com esta linguagem, mas estas coisas são mais importantes"*.
  - Python: *"Nós tentamos fazer sua vida mais fácil, e é isto. Fazer sua vida mais fácil é o que não vamos comprometer."*.
  - **Exemplo:** C++ tentou fazer a vida do desenvovedor mais fácil, porém comprometimentos com performance e compatibilidade com C++, o que não ajudou em muito a vida do desenvolvedor. Qualquer problema do C++ pode ser explicado devido a estas duas características.
  - **Exemplo:** Java achou que o marketing era mais importante.

Bruce Eckel (2)

- **Menos Porcaria:** Dizem que uma pessoa armazena 7 ± 2 informações ao mesmo tempo, quanto menos porcaria, melhor.
  - **Java:** 3 passos para abrir um arquivo. Muito mais para iterar sobre suas linhas.
  - **Python:** 1 chamada para abrir um arquivo. Construções fáceis para iterar sobre o conteudo:
    `for line in file('fname').readlines(): print line`.
  - A falta de caracteres inúteis e a obrigatoriedade de uma identação consistente também ajuda a se concentrar no problema.
- **Produtividade é mais importante que performance:** otimizar prematuramente é um grande erro.
  - A maioria dos programas não precisa de alta-performance
  - A maioria dos programas precisa de alta-produtividade
  - Dos programas que precisam de performance, a maioria se resolve com um *profiler* e otimizações algorítmicas.

Bruce Eckel (3)

- **Não tenta adivinhar como descobrir erros:** tipagem estática foi um grande avanço em relação ao Assembly, mas pode-se fazer melhor.
  - Tipagem estática serve para digitar mais
  - *Casts* induzem a erros
  - Sistema de *templates* ajudaria... se fosse implementado direito
  - Conferência de tipo tardia agiliza o desenvolvimento: a maiora dos casos que o compilador acusa erro na conferência prematura nunca chegam a acontecer.
  - Em geral, os erros mais difíceis de descobrir são os com testes reais: com Python você tem um programa funcionando mais rapidamente, podento testar com dados reais mais cedo e resolver estes problemas mais rápido.
- **Não tem pessoal de marketing envolvido:** todas as decisões da linguagem são com fundamentos técnicos, vêm de quem usa a linguagem.

Bruce Eckel (4)

- **Digite menos, Olhe mais:** com Python você consegue expressar mais com menos, digitando menos, faz mais rápido, com maior densidade, consegue analisar mais idéias ao mesmo tempo. Eckel diz que tem um ganho de 5 a 10 vezes.
- **Os chutes geralmente estão certos:** As construções, chamadas e módulos em python costumam ser bem consistentes e após pouco tempo de prática torna-se pouco freqüente pausas para consultas a manuais ou mesmo nomes de funções ou métodos. Geralmente é mais rápido digitar o que você acha correto e rodar o teste do que procurar na documentação.
- **Python não fica entre você e o problema:** por ser praticamente um pseudo-código, você não tem que se preocupar com detalhes da linguagem e esquecer do problema. Pense no problema e muito provavelmente este será um código Python.

Paul Graham

Paul Graham desenvolveu a primeira aplicação Web, em 1995, o "Viaweb" feita em LISP. Em 1998 sua empresa foi comprada pelo Yahoo!, onde hoje é pesquisador. Foi o inventor do filtro de Spam Bayesiano. Apesar de gostar de LISP, desenvolve uma linguagem concorrente chamada Arc. É o autor do livro *"Hackers & Painters"*.

Paul é influente e tem vários amigos que trabalham em diversas áreas, tendo contato com programadores Java, Lisp, Python e outros. Ele nota alguns padrões interessantes os quais descreve em alguns ensaios dos quais apresento algumas partes.

Paul Graham sobre os desenvolvedores Python

Great Hackers:

> *When you decide what infrastructure to use for a project, you're not just making a technical decision. You're also making a social decision, and this may be the more important of the two. For example, if your company wants to write some software, it might seem a prudent choice to write it in Java. But when you choose a language, you're also choosing a community. The programmers you'll be able to hire to work on a Java project won't be as smart as the ones you could get to work on a project written in Python. And the quality of your hackers probably matters more than the language you choose. Though, frankly, the fact that good hackers prefer Python to Java should tell you something about the relative merits of those languages.*

The Python Paradox:

> *I didn't mean by this that Java programmers are dumb. I meant that Python programmers are smart. It's a lot of work to learn a new programming language. And people don't learn Python because it will get them a job; they learn it because they genuinely like to program and aren't satisfied with the languages they already know.*

The Zen of Python

Para entender os "porquês" da linguagem Python basta abrir o intepretador e digitar: `import this`. O seguinte texto de Tim Peters descreve as decisões que governam a linguagem:

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Explicando o Zen do Python

Explicit is better than implicit.
É uma das explicações para a passagem do `self` como primeiro argumento para os métodos de uma classe.

Errors should never pass silently. Unless explicitly silenced.
Não obrigar o usuário a declarar que uma exceção será pega ou ter que silenciá-la. O que parecia ser uma ótima técnica teoricamente demonstrou-se um fiasco na prática, estimulando o uso de *try-catch* que silenciam os erros em Java.
O uso de exceções é um ponto forte do Python.

In the face of ambiguity, refuse the temptation to guess.
Javascript: `"10" + 5 == "105"`, mas `"10" * 5 == 50`.
Esta é a frase-explicação para a divisão de inteiros retornar inteiro: `3/2 = 1`, `3.0/2 = 1.5`.

Comunidade Mundial

- URL: `http://python.org/`
- News: comp.lang.python
- Mail: python-list@python.org, python-help@python.org
- IRC: irc.freenode.org, canal #python

Pessoas Importantes

Algumas pessoas que são bem conhecidas na comunidade mundial:

- **Guido van Rossum:** criador da linguagem e BDFL (*Benevolent Dictator for Life*).
- **Tim Peters:** um dos mais ativos desenvolvedores, também responsável pelo ZODB. É um dos que mais otimizam o interpretador, a implementação hyper-otimizada de *hast table* (dict) é dele.
- **Jim Fulton:** criador do Zope, core python developer desde 1994.

Comunidade Brasileira

- URL: `http://www.pythonbrasil.com.br/`
- Mail: python-brasil@yahoogrupos.com.br
- IRC: irc.freenode.org, canal #python-br

Pessoas Importantes

Algumas pessoas que são bem conhecidas na comunidade brasileira:

- **Gustavo Niemeyer:** Python Core Developer, trabalha na Conectiva, criador do Smart, mantenedor do módulo de expressões regulares, dentre outros.
- **Rodrigo Senra:** Desenvolvedor Python há anos, grande conhecedor dos "internos" da linguagem
- **Osvaldo Santana Neto:** criador da Python-Brasil, mantenedor do site e da lista
- **Pedro Werneck:** Desenvolvedor Python há anos, conhece muito sobre interface gráfica com o Tkinter.
- **Fabio Rizzo:** Desenvolvedor Python, Zope e Plone de longa data. Possui diversos artigos sobre o assunto.
- **Luciano Ramalho:** Programador desde 77, fundador da Hiperlógica e da Simples consultoria, especialista em Zope.

- Procura na documentação; `http://starship.python.net/crew/theller/pyhelp.cgi`
- Antes de Perguntar: `http://www.pythonbrasil.com.br/moin.cgi/AntesDePerguntar`
- Histórico da Lista Python-Brasil: `http://news.gmane.org/gmane.comp.python.brasil`

- Python `http://python.org`
- Uso de Python no Brasil: `http://www.pythonbrasil.com.br/moin.cgi/EmpresasPython`
- Uso de Python no Mundo: `http://pythonology.org/success`
- Python Quotes: `http://www.python.org/Quotes.html`
- Bruce Eckel 1: `http://www.artima.com/intv/aboutme.html`
- Bruce Eckel 2: `http://www.artima.com/intv/prodperf.html`
- Bruce Eckel 3: `http://www.artima.com/intv/typing.html`
- Bruce Eckel 4: `http://www.artima.com/intv/tipping.html`

- Paul Graham — Great Hackers: `http://www.paulgraham.com/gh.html`
- Paul Graham — The Python Paradox: `http://www.paulgraham.com/pypar.html`

Contato

Gustavo Sverzut Barbieri

Email:	barbieri@gmail.com
Website:	http://www.gustavobarbieri.com.br
ICQ:	17249123
MSN:	barbieri@gmail.com
Jabber:	gsbarbieri@jabber.org

Obtenha esta palestra em:
http://palestras.gustavobarbieri.com.br/python-5hs/