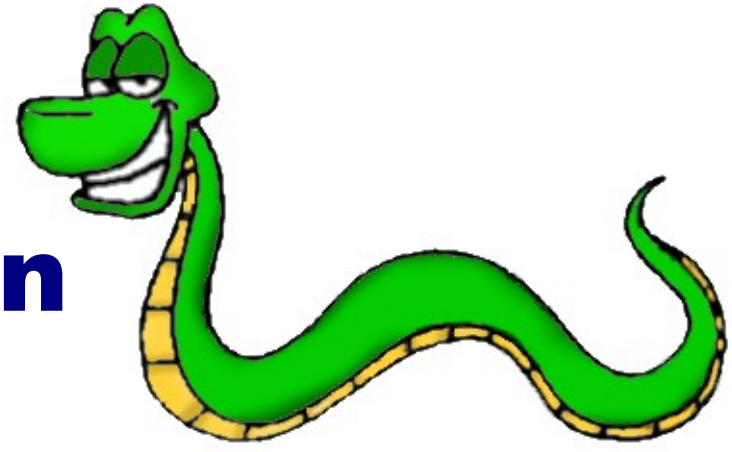
Matando o Java e...

Estrelando

Osvaldo Santana Neto
<osantana@gmail.com>

Ruda Sumé Tente de Moura
<ruda.moura@gmail.com>

... mostrando o Python



Histórico

Python

- Criada em 1991
- Guido Van Rossum
- *Monty Python Flying Circus*
- Ensino de Programação
- Simplicidade e fácil aprendizado

Java

- Criada em 1995
- James Gosling (Sun)
- Cafeteria Java
- *Java Everywhere*
- Plataforma unificada e Evolução de C++

Agile Programming Language

Terminologia criada por **Kevin Altis** e **Ward Cunningham** para definir linguagens como **Python**, **Ruby**, entre outras...

Uma linguagem de programação ágil é caracterizada por:

- **Fácil de usar para aprendizes e poderosa para programadores experientes**
- **Escalável, ideal tanto para projetos pequenos como para projetos grandes**
- **Permita o desenvolvimento rápido de aplicações**
- **Seja portável e multiplataforma**
- **Facilmente extensível**
- ***"Embeddable"***
- ***Orientada a objetos***
- ***Simples e ao mesmo tempo elegante***
- ***Permitir ao programador fazer o seu trabalho***
- ***Estável e madura***
- ***Biblioteca padrão poderosa***
- ***Riqueza de bibliotecas de terceiros***

Características *(parte I)*

Linguagem orientada a objetos com suporte aos paradigmas:

- Estrutural
- Funcional

Tipagem Forte e Dinâmica

Multiplataforma

- Windows, Unix/Linux, PalmOS, SymbianOS, Java

Implementação principal: Licença GPL-compatível

Linguagem orientada a objetos sem suporte a outros paradigmas.

Tipagem Forte e Estática

Multiplataforma

- Windows, Unix/Linux, PalmOS, SymbianOS

Implementação principal: Freeware / Proprietária

Características *(parte II)*

- Liberdade
  - Liberdade para o programador desenvolver o software à sua maneira
- Interpretada (bytecode + VM)
- Ambiente interativo
- Integra facilmente com:
  - C, C++, Java, Perl, Lua, ...

- “Policiamento”
  - Linguagens tipadas 'policiam' mais o desenvolvedor para evitar erros
- Compilada (bytecode + VM)
- Não possui um ambiente interativo
- Integra com:
  - C e C++

Exemplo Python


```
class Component(object):
    def __init__(self, *kargs):
        self._components = list(kargs)

    def add(self, component):
        self._components.append(component)

    def __str__(self):
        ret = ""
        for component in self._components:
            ret = "%s%s" % (ret, component)
        return ret

class Normal(Component): pass

class Bold(Component):
    def __str__(self):
        return "<b>%s</b>" % (Component.__str__(self))

class Italic(Component):
    def __str__(self):
        return "<i>%s</i>" % (Component.__str__(self))

para = Normal("Isto é um texto normal")
para.add(Bold(", este é Negrito"))
para.add(Bold(Italic(" e este é Negrito/Itálico")))
print para
```

```
Isto é um texto normal<b>, este é Negrito</b><b><i> e este é Negrito/Itálico</i></b>
```

Exemplo Java *(parte I)*


```
package br.com.pythonbrasil.java;
import java.util.Vector;

public class Component {
     private Vector _components = new Vector();

     Component(Component element) { this._components.add(element); }
     Component(String element) { this._components.add(element); }
     public String toString() {
          String ret = "";
          for (int i = 0; i < this._components.size(); i++) {
               ret += this._components.get(i).toString();
          }
          return ret;
     }
     public void add(Component element) { this._components.add(element); }
     public void add(String element) { this._components.add(element); }
}

public class Normal extends Component {
     Normal(String element) { super(element); }
     Normal(Component element) { super(element); }
}

public class Bold extends Component {
     Bold(String element) { super(element); }
     Bold(Component element) { super(element); }
     public String toString() {
          return "<b>" + super.toString() + "</b>";
     }
}
```

Exemplo Java *(parte II)*


```
public class Italic extends Component {
     Italic(String element) { super(element); }
     Italic(Component element) { super(element); }
     public String toString() {
          return "<i>" + super.toString() + "</i>";
     }
}

public class Html {
     public static void main(String[] args) {
          Normal texto = new Normal("Isto é um texto normal");
          texto.add(new Bold(", este é Negrito"));
          texto.add(new Bold(new Italic("e este é Negrito/Itálico")));
          System.out.println(texto.toString());
     }
}
```

```
Isto é um texto normal<b>, este é Negrito</b><b><i> e este é Negrito/Itálico</i></b>
```

Desenvolvimento (parte I)

Web

- Zope
- Webware (entre outros)
- TurboGears / Django

Banco de Dados

- DB-API (Relacional)
  - Oracle, MySQL, PostgreSQL, SQLite, ...
- Objeto (Objetos nativos)
  - ZODB

Web

- JBoss (J2EE)
- Tomcat
- CGI

Banco de Dados

- JDBC (Relacional)
  - Oracle, MySQL, PostgreSQL, SQLite, ...
- Objeto (Objeto Relacional)
  - JDO, Hibernate

Desenvolvimento (parte II)

GUI

- Tkinter
- wxPython
- PyQT, PyGTK, PyFLTK, ...

Rede

- Biblioteca Padrão
  - Sockets, ftp, http, smtp, pop3, ntp, imap, rpc, ...
- Twisted
  - Sockets, ftp, http, smtp, pop3, ntp, imap, ssl, ssh, ...

GUI

- Swing
- AWT
- SWT (Eclipse)

Rede

- API Padrão
  - Sockets, http, rmi, entre outros...
- Outros frameworks
  - Diversas implementações de protocolos

Desenvolvimento (parte III)

Webservices

- XML-RPC (padrão)
- SOAP (3rd party)
- outros...

Miscelâneos

- Threads
- Unicode (i18n)
- XML
- ...

Webservices

- Provido por diversos frameworks (3rd party)

Miscelâneos

- Threads
- Unicode (i18n)
- XML
- ...

Ferramentas

IDEs

Eric3, Spe, Boa-Constructor, IDLE, plugin para Eclipse

Outras Ferramentas

Depurador e Profiler

Testes automatizados:

doctest e unittest

IDEs

Eclipse, Jbuilder, Netbeans, ...

Outras Ferramentas

Depurador e Profiler

Testes automatizados:

unittest (3rd party)

Estudo comparativo

Estudo comparativo entre diversas linguagens:

Lutz Prechelt (Faculdade Karlsrure)

C, C++, Java, Perl, Python, Rexx e Tcl

Aplicação para busca/processamento de *strings*

Número de Programas utilizados para o estudo:

Programas: Número de programas estudados

Segunda: Número de participantes da segunda rodada (1 ano)

Não-usáveis: Programas não funcionais

Linguagem	Programas	Segunda	Não-usáveis	Total
Java	26	2	2	24
Python	13	1	0	13

Estudo comparativo (parte II)

Métrica	**Python***	**Java****
Tempo de Desenvolvimento (hrs)	3	9
Tamanho dos programas (comandos/LoC)	80	240
Confiabilidade***	100%	100%
Produtividade (LoC/hrs)****	35	25

* Python 1.5.2 ** Sun JDK 1.2.1 *** Amostragem de resultados corretos

**** O estudo explica a validade da métrica LoC/hrs para produtividade

Tipagem

Tipagem estática / corretude do código

Lenda: Tipagem estática garante corretude de código (*castings* errados)

Fato: O que garante a corretude do código são testes, não tipagem ("*Strong testing, not strong typing.*" - Bruce Eckel)

Tipagem estática / trabalho do programador

Lenda: Tipagem estática transfere trabalho de checagem de tipos do programador para o compilador (esse tipo de checagem deve ser feito pelos testes)

Fato: Tipagem estática faz o programador se preocupar com c*astings*

Fato: Tipagem estática gera código maior e com legibilidade inferior (*casting*).

Jython

O melhor de dois mundos!

- `http://www.jython.org`
- Compila código Python para bytecode Java
  - Escrever applets em Python?
- Integra código Python em aplicações Java
  - Permite que classes Java herdem classes Python
  - Utilizado para adicionar suporte a scripts a aplicações
- Integra código Java em aplicações Python
  - Permite que classes Python herdem classes Java
  - Utilizar frameworks Java em aplicações Python

Processo de desenvolvimento

Processo de desenvolvimento da linguagem Python:

- PSF (*Python Software Foundation*)
- PEP (*Proposal Enhancement Python*)
- Votação
- Pronunciamento do BDFL (*Benevolent Dictator for Life*) BDFL = Guido
- Escolha do Release

Processo de desenvolvimento da plataforma Java:

- JCP (*Java Community Process*)
- JSR (*Java Specification Requests*)
- JCP Member (assinatura de um NDA e pagamento de taxas para empresas)
- Discussão, votação, definições, burocracias, ...

Desvantagens

Desvantagens de Python com relação a Java:

- Falta de profissionais qualificados
- Linguagem pouco conhecida no meio empresarial
- Não possui suporte de uma empresa do porte da Sun
- Linguagem em desenvolvimento constante e ainda sem uma padronização forte.
- Documentação impressa em português é escassa (praticamente inexistente)
- Libera o programador para fazer o que bem entender, até mesmo cometer erros

Vantagens

Vantagens de Python com relação a Java:

- Linguagem de fácil aprendizado
- Linguagem em constante desenvolvimento. Novos conceitos sempre sendo implantados
- Documentação vasta, de qualidade e facilmente encontrada na internet (maioria em outros idiomas)
- A comunidade de Software Livre costuma ser mais eficiente que empresas
- Libera o programador para fazer o que bem entender, até mesmo desenvolver software de qualidade

Leitura Recomendada

"**Livre mas restrito: A Armadilha Java**"
- Richard Stallman

http://www.propus.com.br/news/40

(original em: **http://tinyurl.com/3bjv9**)

http://www.pythonbrasil.com.br

http://www.indt.org.br