Python WTF?

também conhecido como *Python What a FAQ?*

Osvaldo Santana Neto <osvaldo.neto@openbossa.org>

20 de junho de 2007

Python

Uma introdução obrigatória

- Criada por Guido van Rossum em 1991
- Nome provêm do programa de TV *Monty Python and the Flying Circus*
- Sintaxe simples e fácil de ser assimilada
- Linguagem Orientada à objetos com suporte aos paradigmas estrutural e funcional
- Tipagem Forte e Dinâmica
- Multiplataforma
- Interpretada (compila para *bytecode* e interpreta)
- Ambiente Interativo

Python

Um exemplo obrigatório

```
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 e um texto normal")
para.add(Bold(", este e negrito"))
para.add(Bold(Italic(" e este e negrito/italico")))
print para
```

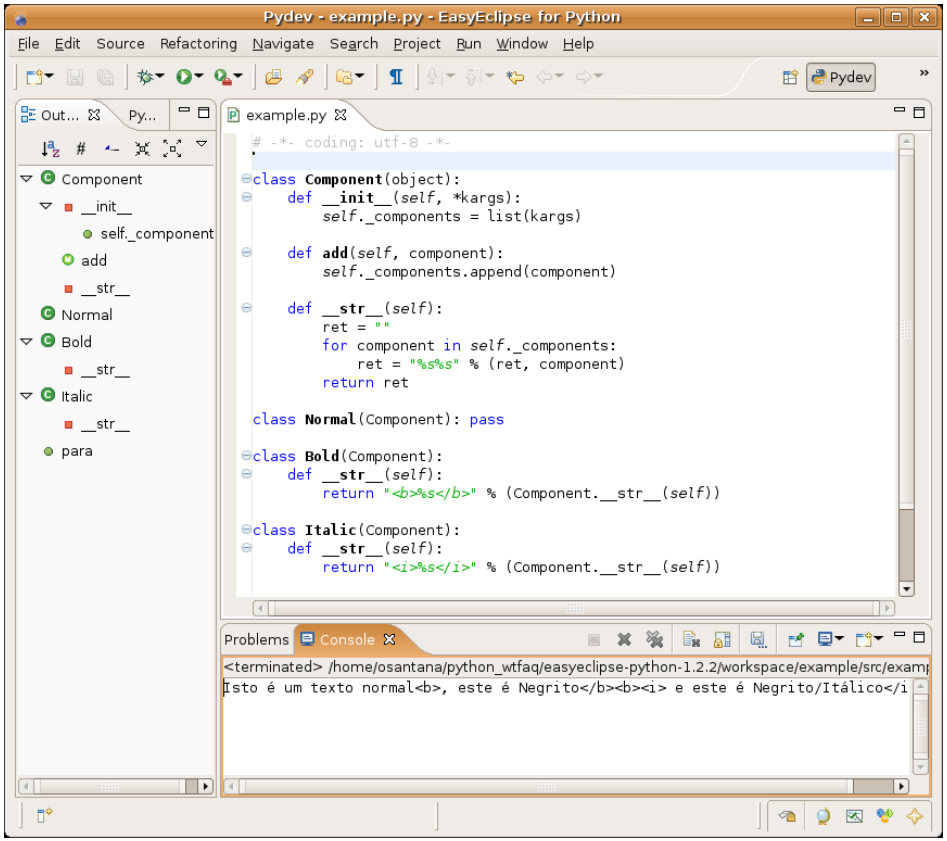
IDEs

- Linguagem de Programação não é IDE
- IDE não é linguagem de Programação

  "Uma coisa é uma coisa, outra coisa é outra coisa".
  – Desconhecido

- IDE não é só um editor de texto
- Nem toda IDE possui um editor de telas
- Nem sempre é necessário usar uma IDE
- IDEs: Eclipse, Visual Studio, Komodo, Delphi, Netbeans, ...
- Editores de Texto: Vi, Emacs, jEdit, Notepad, ...
- Linguagens de programação: Python, Java, Lisp, C, C++, ...
- Não irei comentar sobre todas as ferramentas disponíveis
- `http://pythonbrasil.com.br/moin.cgi/IdesPython`

IDEs Livres

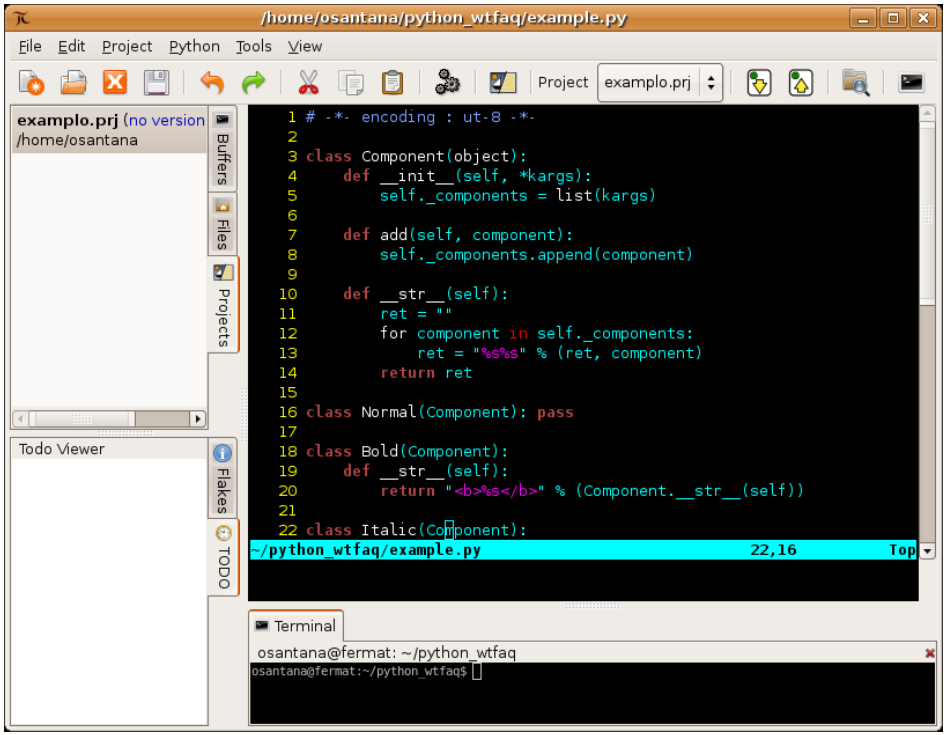


- **Eclipse**
- Características:
  - Autocompletion
  - Syntax Highlight
  - Navegação em código
  - Verificação de código (PyLint)
  - *Unit Test*
  - Integração com Depurador
  - Integração com VCSs

`http://www.easyeclipse.org/`

A versão do Eclipse empacotada pelo grupo EasyEclipse já acompanha diversos plugins úteis no desenvolvimento Python, entre eles o PyDev.

IDEs Livres

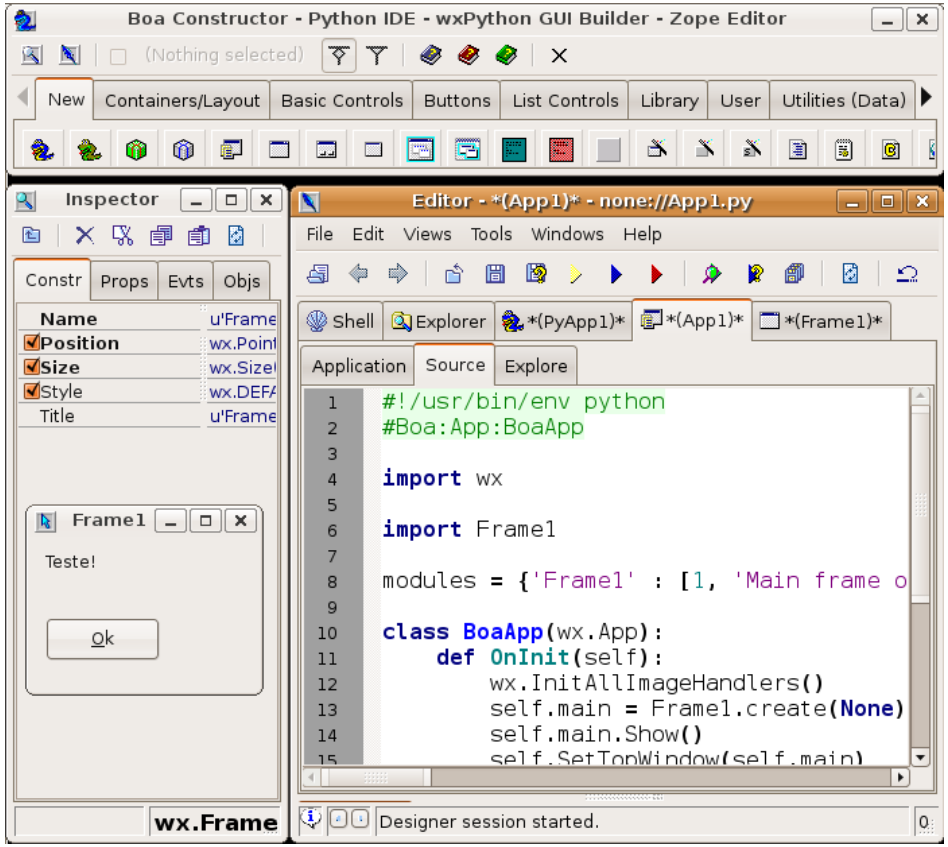


- **Pida**
- Características:
  - Autocompletion
  - Syntax Highlight
  - Verificação de código (PyFlakes)
  - Integração com editor de tela (Gazpacho)
  - Integração com VCSs

`http://pida.berlios.de/`

Um dos pontos fortes do Pida é a possibilidade de escolher entre um editor de textos do próprio projeto ou o Vi para trabalhar com a edição dos arquivos. Ele trabalha com o Gazpacho que é um desenhador de telas que usa o toolkit GTK+.

IDEs Livres

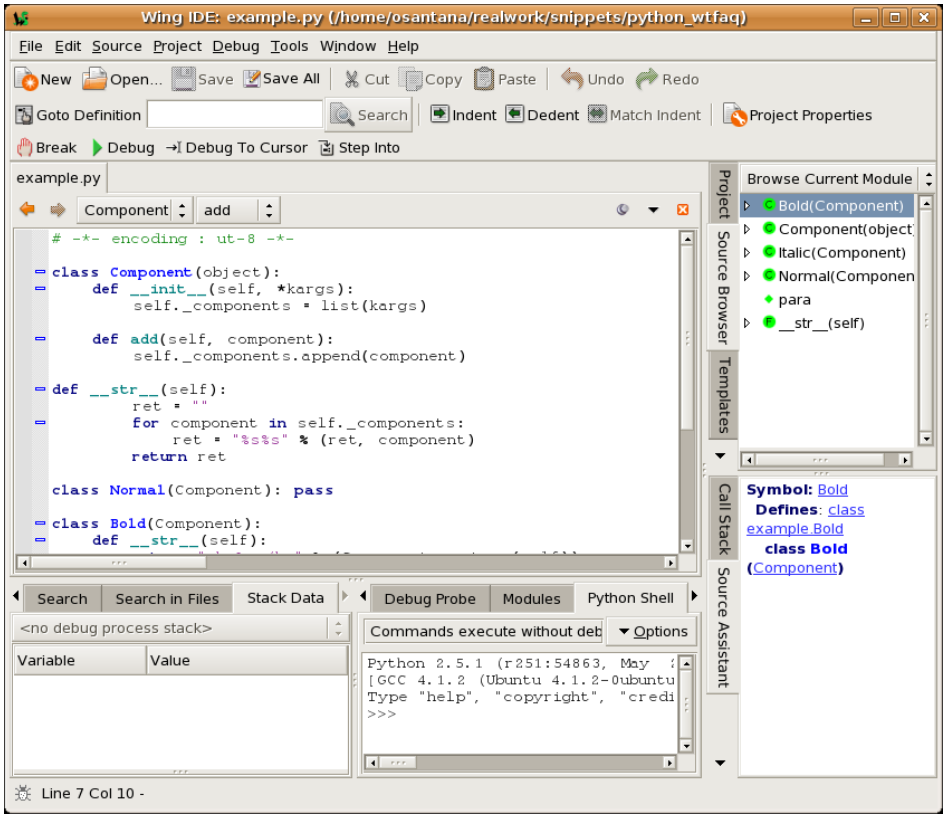


- **Boa-Constructor**
- Características:
  - Autocompletion
  - Syntax Highlight
  - Navegação em código
  - Integração com editor de tela
  - Integração com Depurador

`http://boa-constructor.sf.net/`

O foco principal dessa IDE são as aplicações gráficas. Ela propositalmente tenta imitar o visual do Delphi (apesar de algumas pequenas diferenças).

IDEs Proprietárias

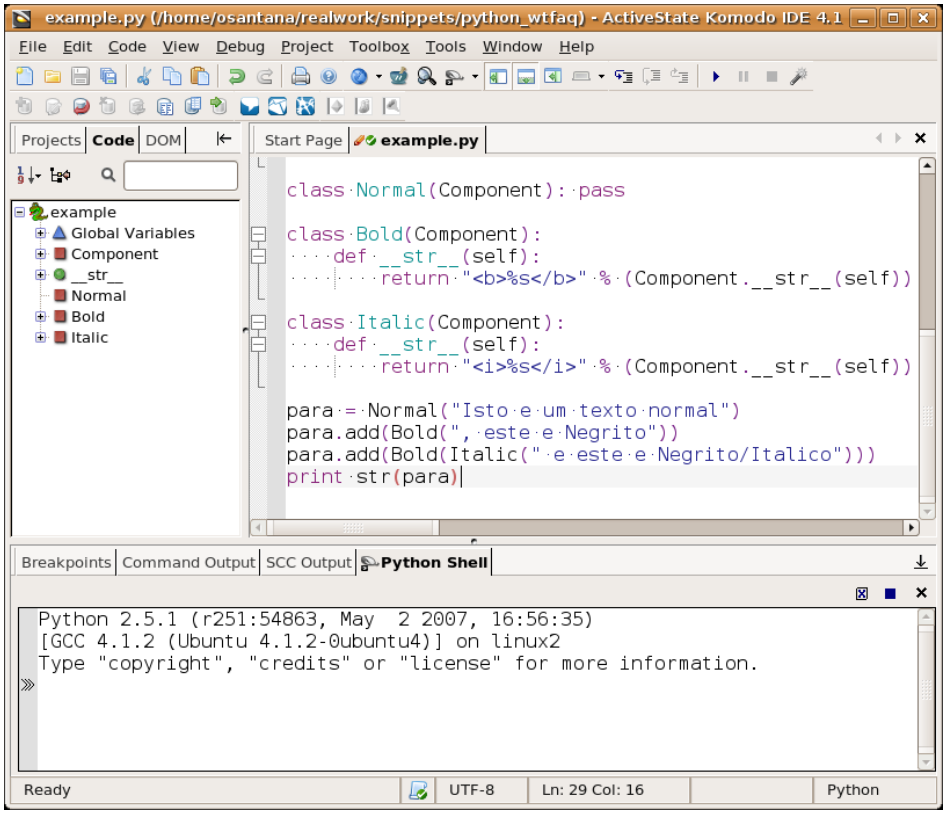


- **Wing IDE**
- $179 (Pro) / $35 (Personal)
- Características:
  - Autocompletion
  - Syntax Highlight
  - Navegação em código
  - Integração com Depurador
  - Integração com VCSs

`http://wingide.com/`

Essa IDE é muito conhecida entre os programadores Python pois foi uma das primeiras disponíveis para se trabalhar com essa linguagem. Os desenvolvedores Zope a utilizam principalmente por sua ferramenta de depuração integrada.

IDEs Proprietárias



- **Komodo IDE**
- $295
- Características:
  - Autocompletion
  - Syntax Highlight
  - Navegação em código
  - Verificação de código
  - Integração com Depurador
  - Integração com VCSs

`http://activestate.com/`

Essa IDE é ideal para desenvolvedores que trabalham com aplicações Web. Ela é projetada especialmente para esse tipo de desenvolvimento.

Editores de Texto Livres

- **Vi** - `http://vim.org` - Editor muito conhecido no universo dos Unix. Curva de aprendizado íngreme mas extremamente poderoso.
- **Emacs** - `http://gnu.org/software/emacs` - Editor muito conhecido no universo dos Unix. Curva de aprendizado menos íngreme que a do Vi. O Emacs também é extremamente poderoso e é um dos principais rivais do Vi.
- **jEdit** - `http://jedit.org/` - Editor de textos multiplataforma tem uma riquíssima biblioteca de plugins extremamente úteis para desenvolvedores.
- Outros - SciTE, jExt, Scribes[1], Gedit, Kate, ...

[1] `http://scribes.sf.net/`

Editores de Texto Proprietários

- **Komodo Edit** - `http://activestate.com` - Grátis - A versão reduzida da IDE Komodo mencionada anteriormente conta apenas com as funcionalidades de editor de textos.
- **TextMate** (OS X) - `http://macromates.com` - €39 - O editor de textos mais badalado do universo Apple. É extremamente fácil de usar e poderoso e pode ser personalizado através de *scripts*.
- **UltraEdit** (Win) - `http://ultraedit.com/` - $49.95 - Velho conhecido dos programadores Windows esse editor de textos ainda "bate um bolão" entre eles.

Primeira grande questão

Que IDE/Editor de textos usar?

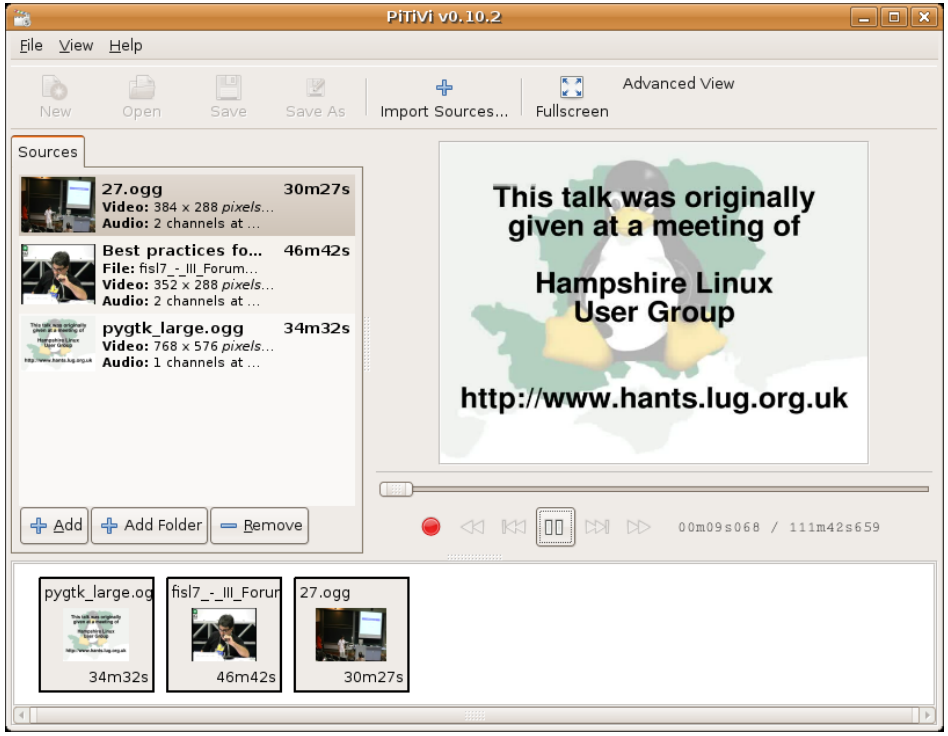
- **Menos é mais** - Prefira IDEs mais simples de serem usadas. Seu foco tem que ser "aprender Python"e não a IDE. Nessa linha de raciocínio prefira editores de textos à IDEs.
- **Eu uso...** - Vi[2]. Tive que aprender a usar Vi para trabalhar com Linux e desde então ele supre as minhas necessidades totalmente.
  - Sim, é difícil de aprender a usá-lo.
  - Mas uma vez aprendido ele é o melhor. :)
- **E se eu usasse uma IDE?**
  - Seria Komodo com dinheiro, ...
  - ...Eclipse sem dinheiro ou...
  - ...Pida (com Vi) depois de pronto. :)

[2] O Scribes também parece interessante

Toolkits Gráficos

- Critérios para a escolha de um Toolkit gráfico
  - **Licenciamento** - Eu posso usar o toolkit livremente? Se minha aplicação não for livre, tenho que pagar?
  - **Aparência** - A minha aplicação vai ter a aparência padrão da plataforma? Vai ficar mais feia? Mais bonita?
  - **Ferramentas** - Eu vou ter que desenhar minhas telas no meu programa ou posso usar um editor para isso?
  - **Dependência** - Vou ter que adicionar mais uma dependência para minha aplicação?
  - **Plataforma** - O toolkit escolhido funciona em várias plataformas?
- Toolkits disponíveis: Tkinter, PyGtk, PyQt, wxPython, PyFLTK, PXPy, Anygui, PySWT, PyUi, ...
- Não irei comentar sobre todos os toolkits disponíveis
- `http://pythonbrasil.com.br/moin.cgi/ComparacaoDeGUIs`

Toolkit Gráfico

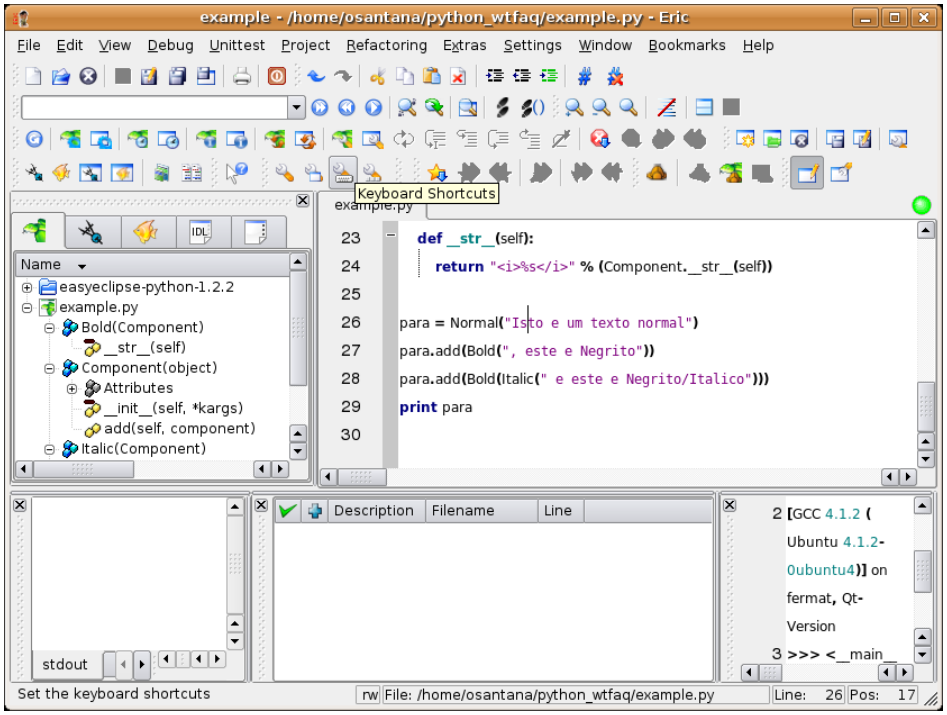


- **PyGtk**
  - **Licença:** LGPL
  - **Aparência:** Gtk+
  - **Dependências:** Gtk+
  - **Plataformas:** Win, OS X, Linux

`http://pygtk.org/`

- **Editores de telas:** Gazpacho, Glade
- **Bibliotecas:** Eagle, PyMVC
- **Framework:** Kiwi (MVC)
- **Aplicações:** `http://www.pygtk.org/applications.html`

Toolkit Gráfico

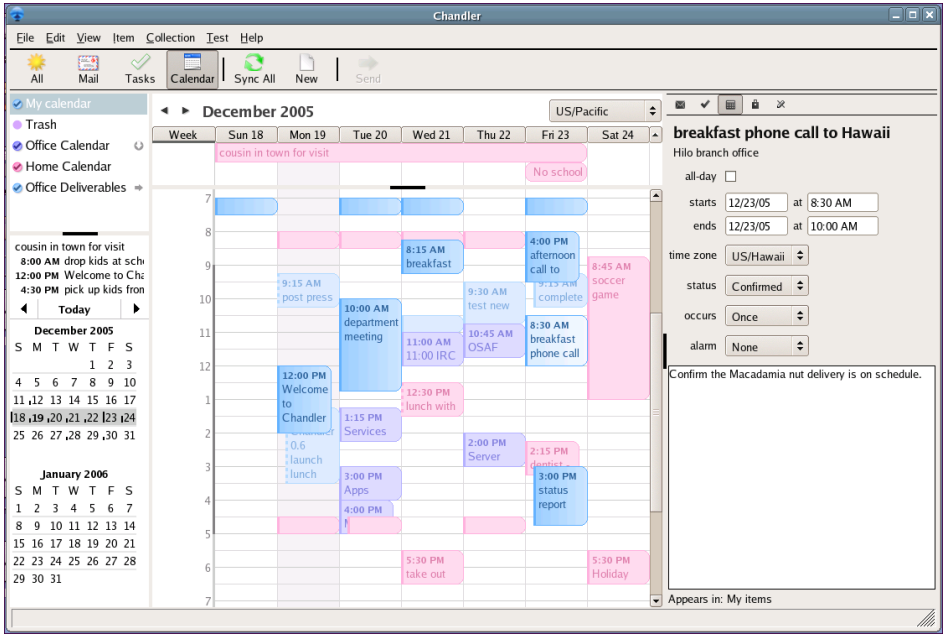


- **PyQt**
  - **Licença:** GPL (ou Qt Licence)
  - **Aparência:** Qt/Nativa
  - **Dependências:** Qt
  - **Plataformas:** Win, OS X, Linux

`riverbankcomputing.co.uk/pyqt`

- **Editores de telas:** Qt Designer
- **Bibliotecas:** PyKDE
- **Aplicações:** Várias aplicações KDE

Toolkit Gráfico



- **wxPython**
  - **Licença:** wxWidgets (BSD)
  - **Aparência:** Nativa
  - **Dependências:** wxWidgets
  - **Plataformas:** Win, OS X, Linux

`http://wxpython.org/`

- **Editores de telas:** wxGlade, wxDesigner (proprietário)
- **Aplicações:** Chandler, Juice, PySoulSeek, ...

Segunda grande questão

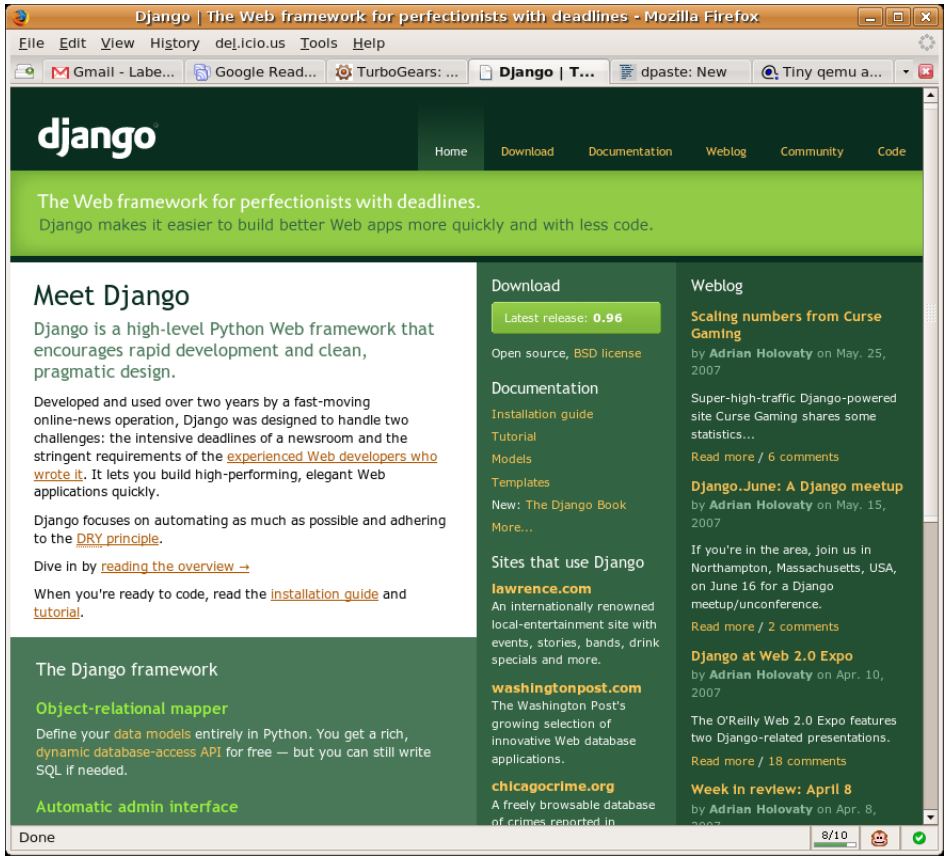
Que toolkit gráfico usar?

- Para escolher o melhor, dê pesos aos items que devem ser avaliados em um toolkit gráfico, dê notas à esses quesitos para cada um dos toolkits e calcule a média ponderada.
- Experimente os dois melhores colocados desenvolvendo uma mini-aplicação.
- **Eu uso...** - PyGtk. Usei o método descrito acima.
  - Sim, eu tive que usar isso no trabalho
  - Não gosto de dois componentes GTK: TreeView e TextView.
  - Dê uma olhada no Eagle e no Kiwi
- Eu não gosto de desenvolver aplicações *standalone*. Prefiro desenvolvimento Web. Empresas como o Google estão mostrando que o futuro está dentro do navegador.

Framework Web

- Ao escolher um framework Web avalie os seguintes itens:
  - **Popularidade** - Os frameworks Web mais usados contam com um conjunto de módulos e *plug-ins* muito maior do que os outros.
  - **Atividade** - Veja se o desenvolvimento do framework é ativo.
  - **Documentação** - Exija documentação organizada, completa e de fácil entendimento. Conteúdos multimídia como *screencasts* também são importantes.
  - **Experimente** - Esse item é o mais importante. Você tem que experimentar os frameworks que mais te interessaram porque só assim você vai conseguir descobrir se ele é mais adequado à sua aplicação.
- Frameworks Web: Django, TurboGears, Pylons, Zope/Plone, Webware, CherryPy, Quixote, Twisted Woven, Spyce, ...
- Não irei comentar sobre todos os frameworks existentes
- `http://www.pythonbrasil.com.br/moin.cgi/PythonParaWeb`

Framework Web



- **Django**
  - **Documentação:** Muito completa e organizada
  - **Uso:** Extremamente fácil
  - **Template:** Django
  - **Persistência:** Django ORM

`http://djangoproject.com/`

O projeto Django nasceu na onda dos frameworks Web ágeis onde o Ruby on Rails foi pioneiro. O framework é muito bem implementado e a documentação merece menção honrosa. A comunidade também é muito ativa.

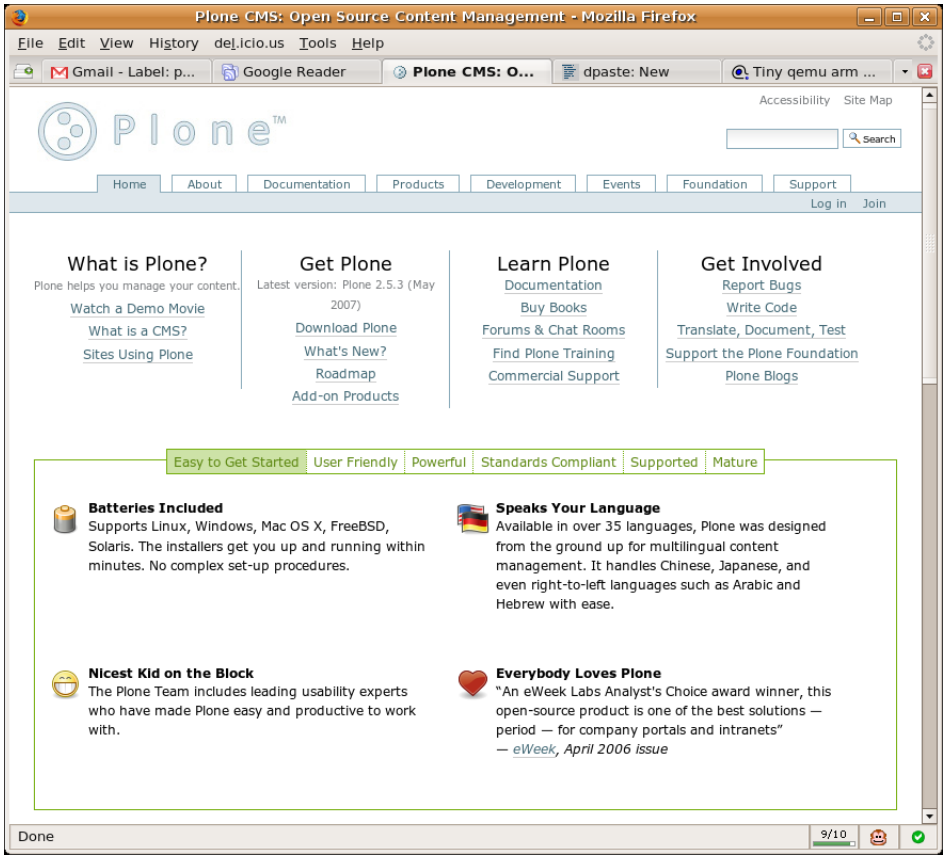
Framework Web

- **TurboGears**
  - **Documentação:** desorganizada
  - **Uso:** Extremamente fácil
  - **Template:** Kid, Genshi ou ZPT
  - **Persistência:** SQLObject ou SQLAlchemy

`http://turbogears.org/`

O Turbogears é o mais fácil de ser usado. O fato dele ser só um aglutinador de outros frameworks também o torna muito mais flexível. Mas o seu defeito é a documentação que é pouca, defasada e desorganizada.

Framework Web



- **Plone (Zope)**
  - **Documentação:** razoável
  - **Uso:** Complexo
  - **Template:** ZPT
  - **Persistência:** ZODB / BDs relacionais

`http://plone.org/`

O Plone não é exatamente um framework Web mas a sua flexibilidade é tão grande que o torna um concorrente destes. Ele é desenvolvido em Zope e esse sim é um software grande e complexo (mas extremamente poderoso).

Terceira grande questão

Que framework Web usar?

- Não existe fórmula mágica para facilitar a escolha. Tem que experimentar cada um deles
- O GvR escolheu o Django
- **Eu uso...** - TurboGears agora. Mas vou usar Django no futuro
  - Eu usei o TurboGears porque comprei o livro e queria justificar o investimento.
  - Eu gosto mais do sistema de mapeamento de URL do TG.
  - Eu ainda não experimentei o Pylons[3], mas ouvi boas histórias sobre ele
- Existe uma intenção dos desenvolvedores do Pylons e do TurboGears de trabalhar em conjunto.

[3] http://pylonshq.com/

Banco de Dados

- Evite escolher Bancos de Dados. Prefira usar frameworks que abstraiam o Banco de Dados.
  - Exceto se o Banco de Dados for Orientado à Objetos
- Frameworks: SQLObject, SQLAlchemy, ZODB, Durus...
- Bancos de Dados: SQLite, MySQL, PostgreSQL, Oracle, MsSQL, ODBC, ...
- Não irei comentar sobre todos os frameworks ou Bancos de Dados existentes
- `http://pythonbrasil.com.br/moin.cgi/BancosDeDadosSql`

Exemplo:

```
from sqlobject import *
__connection__ = "sqlite://:memory:"

class Permission(SQLObject):
  permission_name = UnicodeCol(length=16,
              alternateID=True,
              alternateMethodName='by_permission_name')
  description = UnicodeCol(length=255)
  groups = RelatedJoin('Group',
             intermediateTable='group_permission',
             joinColumn='permission_id',
             otherColumn='group_id')
```

O SQLObject permite fazer o mapeamento objeto-relacional de maneira muito simples e rápida. O banco de dados usado é definido através da variável `__connection__`. O acesso ao BD fica transparente para o desenvolvedor que irá lidar com objetos na maior parte do tempo.

Exemplo:

```
from sqlalchemy import *
from sqlalchemy.ext.assignmapper import assign_mapper

permissions_table = Table('permission', metadata,
  Column('permission_id', Integer, primary_key=True),
  Column('permission_name', Unicode(16), unique=True),
  Column('description', Unicode(255))
)

class Permission(object): pass

def assign(*args, **kw):
    return assign_mapper(session.context, *args, **kw)

assign(Permission, permissions_table,
  properties=dict(groups=relation(Group,
  secondary=group_permission_table, backref='permissions')))
```

O SQLAlchemy usa um outro sistema de trabalho que dá mais flexibilidade ao mapeamento objeto-relacional. Isso o torna ideal para o desenvolvimento de aplicações que precisarão usar bancos de dados legados.

Banco de Dados

Outras alternativas

- **ZODB** - Banco de Dados orientado à objetos, não utiliza o modelo relacional de banco de dados.
- **DB-API** - API Python padronizada para acesso à bancos de dados relacionais.
  - **SQLite** - Acompanha o Python 2.5
  - **MySQL** - mysqldb
  - **PostgreSQL** - psycopg2
  - **Oracle** - cx_oracle
- Exemplo com MySQL:

```
# Para conectar outros bancos, somente essa parte muda
import MySQLdb
con = MySQLdb.connect('servidor', 'usuario', 'senha')
con.select_db('banco de dados')
# Fim

cursor = con.cursor()
cursor.execute('ALGUM SQL')

con.close()
```

Quarta grande questão

Que Banco de Dados usar?

- A regra de outro aqui é: afaste-se o máximo possível do banco de dados.
- Se for usar um banco de dados relacional use um mapeador objeto-relacional
- **Eu uso...** - SQLAlchemy, mas sonho e poder usar só o ZODB
  - O SQLAlchemy parece ser melhor mantido que o SQLObject e sua documentação impressiona pela qualidade
  - O ZODB tem fama de não escalar bem e não é bem suportado por outros frameworks que não o Zope
- Como o SQLite acompanha o Python ele se mostra uma excelente alternativa para o desenvolvimento ou para a execução de testes (usando o banco de dados em memória).

Quinta grande questão

Qual o sentido da vida, do universo e de tudo mais?

Perguntas?

http://www.indt.org.br/