

Herramientas para el desarrollador

Tomas Zulberti

tzulberti@gmail.com

¿Que es lo que queremos hacer?

Queremos hacer un programa en Python, que lo pueda usar cualquiera y que otros puedan contribuir escribiendo código

Por ejemplo: el tp de la facultad en grupo, una aplicación web, un juego para la PyWeek.

Por lo tanto vamos a necesitar lo siguiente:

- Una forma de compartir los archivos
- Escribir código de forma mas rápida
- Instalar librerías hechas por otras personas
- Probar el programa

Importante: En las cosas que voy a mostrar no voy a entrar en mucho detalle, sino que es para dar una idea de las herramientas que existen.

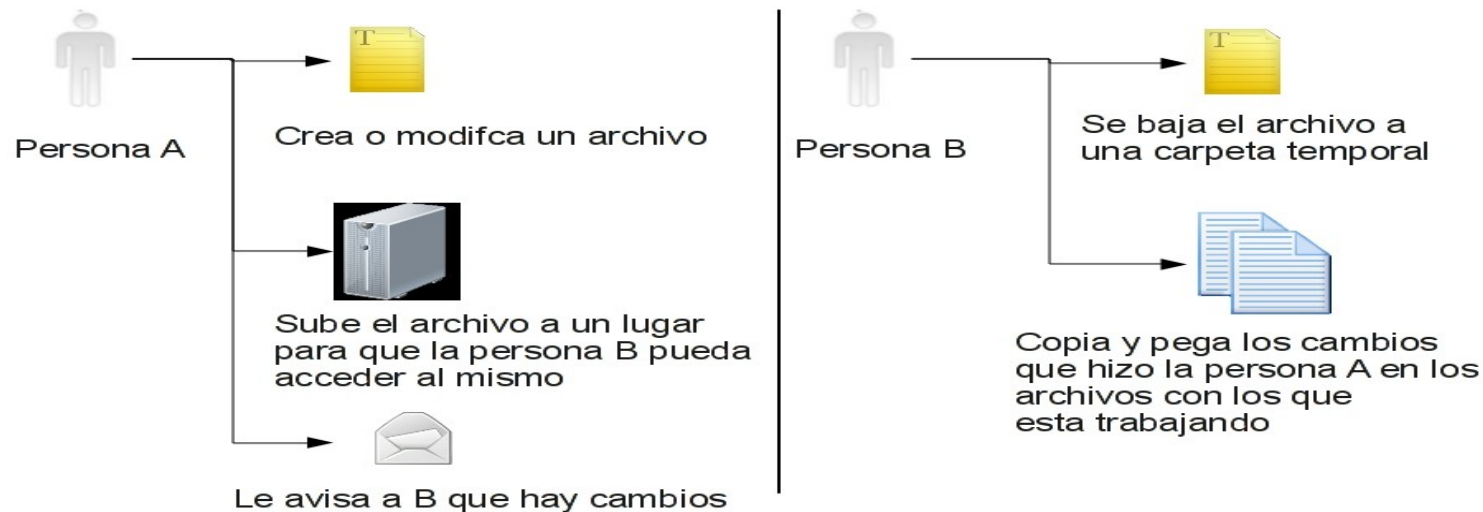
¿Que es mejor?



Cuando todos sabemos que el mejor es:



¿Como es la forma típica de compartir archivos?



La persona A sube el archivo en el que estuvo trabajando que puede ser nuevo o una modificación. Lo pone en algún lugar tal que B pueda acceder a esos cambios, y le avisa a la persona B. La persona B se baja el archivo y copia y pega los cambios que haya hecho la persona A.

Esto trae las siguientes preguntas:

- ¿Que pasa si la persona A se olvida de avisarle un cambio la persona B?
- ¿Que pasa si la persona B también esta trabajando sobre un archivo que trabajo la persona A?
- ¿Que pasa si la persona B copia y pega mal los archivos?

¿Comparto los archivos por mail?

- No esta muy en claro cual es la ultima versión Como cada uno trabaja en su maquina, termina habiendo varias versiones diferentes en el historial del mail.
- No es fácil aplicar los cambios que hizo el otro. Generalmente Pepe se baja la versión que hizo Juan, en una carpeta temporal, se fija que cambios hizo, y los aplica en su maquina.
- Si son mas de dos, entonces el problema es que todos siempre tienen que estar fijándose que hizo el anterior para actualizar su código
- Mas o menos se sabe que cambios hizo cada uno si es que lo escribe en el mail.

¿Comparto los archivos usando una carpeta compartida?

- A diferencia del mail se sabe cual es la ultima versión, y esa es la que este en la carpeta compartida.
- No se tiene una lista con los cambios que hizo cada uno.
- Las carpetas compartidas solo se pueden usar si están físicamente en el mismo lugar.
- Los backups se tienen que hacer a mano. Es decir, si alguien pone en la carpeta algo que no compila, entonces a nadie le va a compilar el proyecto, y tienen que usar un backup.
- De nuevo, uno tiene esa carpeta y además en la que esta trabajando... Por lo que se tiene que copiar archivos de una carpeta a la otra con mucho cuidado.
- Es peligroso si dos personas pisan archivos casi al mismo tiempo...

En resumen, mas o menos los mismos problemas que compartir los archivos por mail, con la diferencia que la ultima versión es la que esta en la carpeta compartida

Finalmente...

[Algoritmos II] Entrega Tp2

Inbox | X

Otros/Alumnos | X



Pablo Rodriguez to me

[show details](#) 9:24 PM (25 minutes ago)

Reply



Hola. Les informo que su entrega del TP2 no compila y por lo tanto no aprueban este tp.

Saludos,
Pablo Rodriguez

[Reply](#)

[Reply to all](#)

[Forward](#)

¿Que cosas nos importan cuando compartimos los archivos?

- Que sea fácil pasar archivos de una maquina a la otra
- Que siempre sepamos cual es la ultima versión
- Que tengamos un log con los cambios que hizo cada uno
- Que yo no tenga que tener una carpeta en donde esta la ultima versión, y además mi versión
- Que se puedan volver para atrás a una versión anterior en cualquier momento de una forma fácil

Y lo mas importante:

- que todo eso sea FACIL de hacer. Es decir, correr un comando o algo similar

Sistema de revisiones de archivos

¿Que son?

Básicamente es un programa que permite subir archivos a un servidor, y que hace todo lo que queremos. Es decir, siempre se puede volver para atrás, agregar un archivo, actualiza los cambios hechos por otra persona, etc...

Todo esto lo hace corriendo comandos MUY simples.

Algunos programas conocidos son:

- svn (subversion)
- git
- mercurial
- bazaar

Todos estos funcionan en Windows, Linux, Mac....



¿Como estos sistemas resuelven los problemas?

Los ejemplos a continuación los muestro con SVN, pero no es muy diferente a otros programas.

Cada vez que uno sube un cambio al servidor se le pide que detalle cuales fueron los cambios hechos. Además, se lista los archivos que se van a subir y la razón (es nuevo, fue modificado, Etc).

```
svn commit
```

Actualizar a la última versión es cosa de un solo comando. Esto además resuelve las diferencias entre el archivo de la máquina local y el del servidor.

```
svn update
```

Se puede hacer un listado de los archivos modificados y darse cuenta si uno agregó uno nuevo

```
svn st
```

Se puede ver los cambios que se les hizo a un archivo.

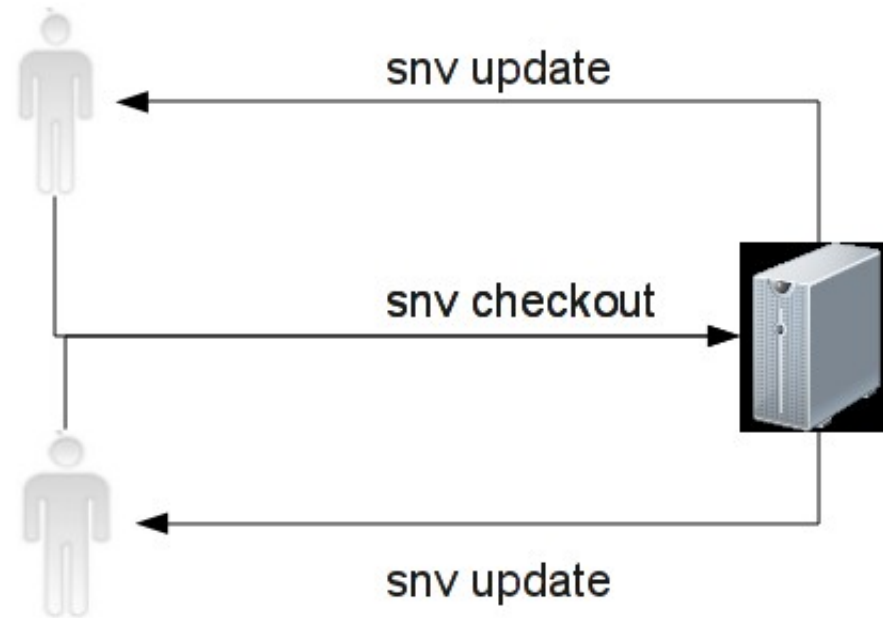
```
svn log nombearchivo.py
```

Siempre se puede volver a una versión anterior del archivo.

```
svn revert -rNroRevision nombearchivo.py
```

MUCHAS MAS COSAS

Sistema de revisiones de archivos > ¿Como se trabaja?



Persona A, llega y hace un **svn update** para traerse los últimos cambios. Hace los cambios que quiere hacer y hace un **svn checkout**. La persona B, también llega y hace un **svn update**, y automáticamente tiene en su carpeta de trabajo los cambios hechos por A.

Notar que acá no es necesario avisar los cambios entre A y B.

Sistemas distribuidos

El svn tiene algunas limitaciones:

- se tiene que tener conexión a internet para poder trabajar.
- todos los cambios se hacen a la misma maquina.

Para solucionar estos problemas, se crearon los sistemas distribuidos. Algunos conocidos son:

- mercurial
- git
- bazaar

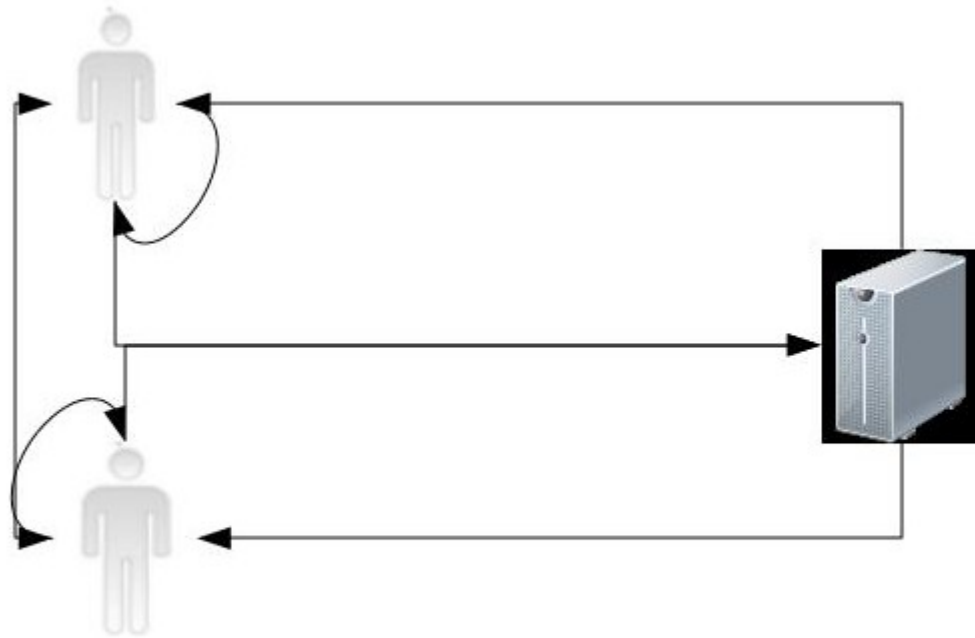
Una lista completa de las diferencias:

http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

Sistemas distribuidos

En este tipo de sistemas, toda maquina es valida:

- se aplican los cambios contra mi propia maquina
- puedo o bajarme los cambios hechos por las persona B
- puedo directamente subir la info al repositorio principal



Hosting gratuitos

SVN	http://code.google.com/hosting/
Git	http://github.com/
Mercurial	http://bitbucket.org/
Bazaar	https://launchpad.net/

No me importa cual usan, pero usen alguno

Una lista mas completa en:

http://en.wikipedia.org/wiki/Comparison_of_free_software_hosting_facilities

La mejor introducción a mercurial que haya visto en mi vida:

<http://hginit.com/index.html>

Hora de escribir código

```
from os import path

class Clase1(object):
    """ Esto es un texto de documentacion.
    """

    def esta_es_una_funcion(self):
        # comentario
        pass
```

```
from os import path

class Clase1(object):
    """ Esto es un texto de documentacion.
    """

    def esta_es_una_funcion(self):
        # comentario
        pass
```

¿Cual de los dos les resulta mas fácil de leer?

Justamente para eso están los entornos de desarrollos (IDEs)

¿Que son?

Básicamente son editores de texto que traen algunas ventajas sobre los editores de texto comunes:

- Coloreo de sintaxis
- Autocompletado de código
- Navegador de archivos
- Ayuda por errores de sintaxis.

Los que voy a nombrar acá son:

- Vim/Emacs
- Eclipse + PyDev
- Komodo/PyCharm

Una lista completa de IDEs para Python: <http://wiki.python.org/moin/PythonEditors>

Vim/Emacs

Son editores de texto bastante conocidos, a los cuales se les puede agregar miles de plugins

Se empieza con algo así:

```
from os import path

class Clase1(object):
    """ Esto es un texto de documentacion.
    """

    def esta_es_una_funcion(self):
        # comentario
        pass
```

Se puede configurar a llegar a algo así:

```
from os import path

class Clase1(object):
    """ Esto es un texto de documentacion.
    """

    def esta_es_una_funcion(self):
        # comentario
        pass
```

Vim/Emacs

Ventajas

- Funciona en todos los sistemas operativos
- Casi no usa memoria.
- Se lo puede usar para programar en cualquier lenguaje
- Funciona tanto en consola como en la parte gráfica

Desventajas:

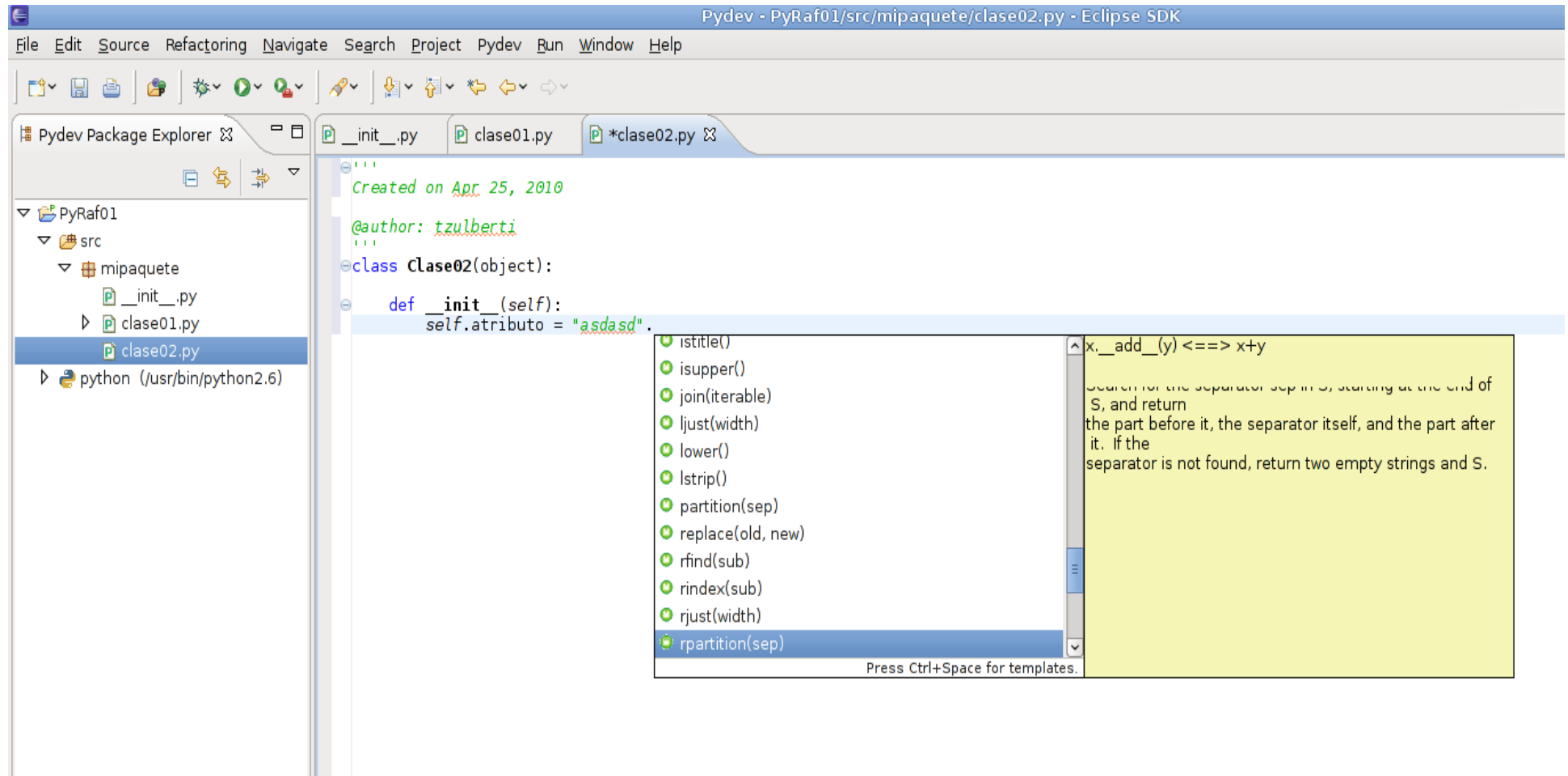
- Difícil de usar al principio. Por ejemplo, para guardar un archivo se escribe “:w”
- Se tarda en configurarlo como uno quiere.

Links de como configurarlo:

- <http://blog.dispatched.ch/2009/05/24/vim-as-python-ide/>
- <http://lucumr.pocoo.org/2007/4/2/vim-as-development-environment>
- <http://dancingpenguinsoflight.com/2009/02/python-and-vim-make-your-own-ide/>

Eclipse + Pydev

El eclipse un IDE que se lo puede usar para varios lenguajes (C, Java, Php, etc..), y que tiene un plugin para Python (Pydev)



Eclipse + Pydev

Ventajas

- Funciona en todos los sistemas operativos
- Se lo puede usar para programar en cualquier lenguaje
- Tiene plugins para todo
- Trabaja con django.
- Funciona también para otros lenguajes (javascript, html, etc)

Desventajas:

- Usa bastante memoria (no corre con menos de 1gb)
- Esta pensado mas para el mundo de Java que el de Python
- Toma un ratito configurarlo.
- Al tener bastantes plugins toma un rato acostumbrarse...

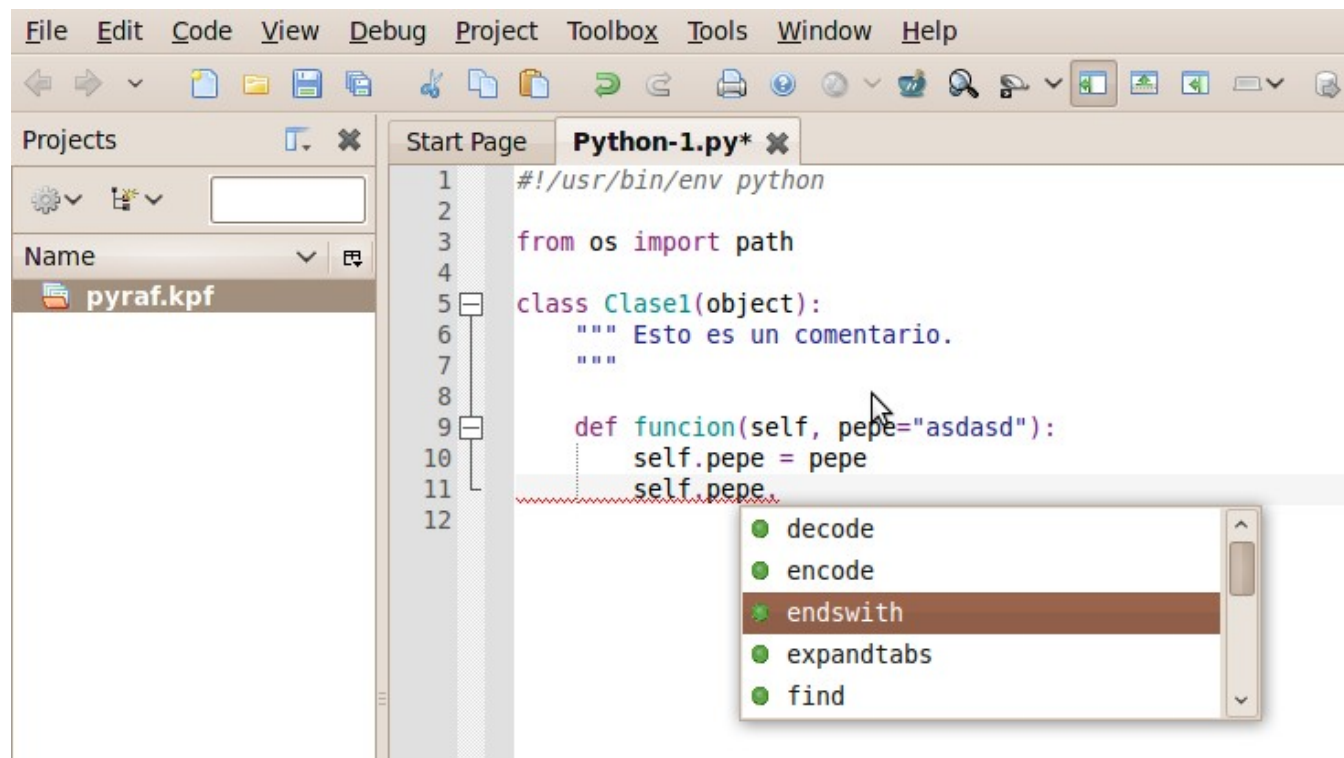
Como configurarlo:

- <http://pydev.org/download.html> (a la derecha de la pagina).
- http://pydev.org/manual_101_interpreter.html (como configurar el interprete)

Komodo / PyCharm

Básicamente son IDEs pensados para Python desde un principio, pero aunque existe una versión gratuita por ahora, puede que tengan que usar una limitada.

Komodo tiene dos versiones: paga y gratuita, pero PyCharm solo tiene una que es gratuita por ahora (no se sabe por cuanto tiempo va a seguir así).



Komodo / PyCharm

Ventajas

- Funciona en todos los sistemas operativos
- Están pensados para ser IDEs en python y/o lenguajes interpretados (php, ruby, etc)
- Una vez instalados están listo para ser usados.

Desventajas:

- Aunque por ahora son gratuitos, no se sabe si va a tener una versión paga. Para el Komodo existe una versión gratuita que esta limitada, y una versión Paga.
- Solo funciona en la parte gráfica
- Usan bastante RAM

Librerías / Frameworks

Necesito una librería que:

- se conecte a una base de datos (postgres, mysql, oracle, etc..)
- parsee archivos de formato xml,yaml, json, etc...
- necesito django
- quiero una librería que haga mi trabajo....

Python viene con baterías incluidas pero a veces no les da suficiente voltaje.



Librerías / Frameworks

Hay miles de librerías que se pueden bajar desde:

<http://pypi.python.org/>

Están ordenadas por categorías..



» Package Index

PACKAGE INDEX »

[Browse packages](#)
[Package submission](#)
[List trove classifiers](#)
[List packages](#)
[RSS \(last 40 updates\)](#)
[Python 3 packages](#)
[Tutorial](#)
[Get help](#)
[Bug reports](#)
[Comments](#)
[Developers](#)

ABOUT »

NEWS »

DOCUMENTATION »

DOWNLOAD »

COMMUNITY »

FOUNDATION »

CORE DEVELOPMENT »

LINKS »

PyPI

The Python Package Index is a repository of software for the Python programming language. There are currently **9828** packages here. To contact the PyPI admins, please use the [Get help](#) or [Bug reports](#) links.

To submit a package use "[python setup.py upload](#)" and to use a package from this index either "[pip install package](#)" or download, unpack and "[python setup.py install](#)" it.

- [Browse the tree of packages](#)
- [Submit package information](#) (note that you must [register to submit](#))

There now is an [RPM repository](#) that provides RPMs for most of the packages available here on PyPI.

Updated	Package	Description
2010-05-08	c2c.recipe.jsconfig 0.1	Generate a javascript configuration file based on buildout configuration
2010-05-08	stxnext.staticdeployment 0.5.8	Deploy Plone site to static files.
2010-05-08	django-webtest 1.0.3	Instant integration of Ian Bicking's WebTest (http://pythonpaste.org/webtest/) with django's testing framework.
2010-05-08	eventbeat 0.2.3	UNKNOWN
2010-05-08	monocle 0.2	An async programming framework with a blocking look-alike syntax
2010-05-08	Flickr.API 0.4.4	A Python interface to the Flickr API
2010-05-08	colander 0.6.2	A simple schema-based serialization and deserialization library

Not Logged In

[Login](#)
[Register](#)
[Lost Login?](#)
[Use OpenID](#) 
[ip](#)

Instalando librerías

Los frameworks/librerías se distribuyen en un archivo .egg



Básicamente es un archivo Zip con algunos archivos específicos y el código.

Para instalarlos hay dos formas:

- bajarlo e instalarlo a mano
- usar `easy_install`

Instalación manual

Generalmente en estos casos se hace lo siguiente:

- se baja un archivo comprimido en .zip o .tar o lo que sea
- se lo descomprime
- se corre `python setup.py install`

Un ejemplo de esto es la instalación de Django

Pero es un garrón tener que bajar el código a mano... estaría bueno que esa parte se haga automáticamente...

Usando easy_install

Para usarlo se tienen que instalar un paquete adicional:
- en debian/ubuntu es: python-setuptools

Ahora instalar un programa es tan simple como:
easy_install nombrePrograma

Por ejemplo:
easy_install django

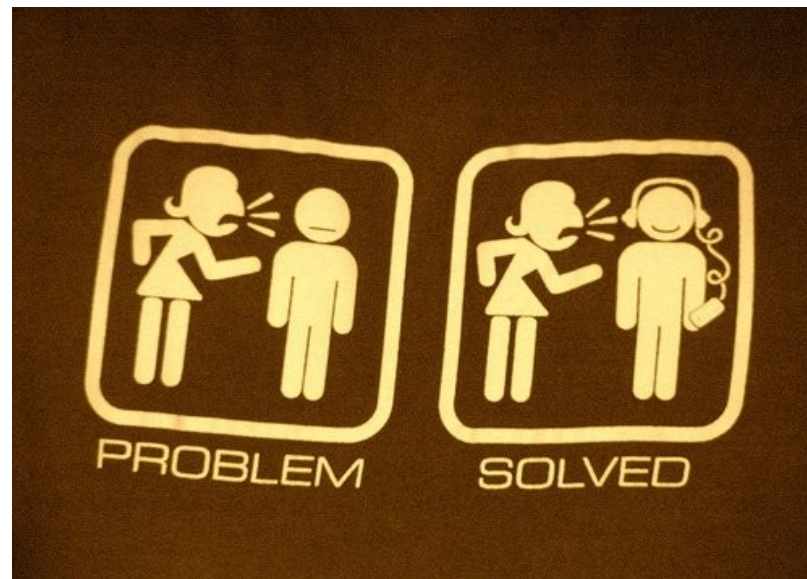
Esto se encarga de bajar el código, y correr el setup install y de bajar todas las dependencias.

Problemas

Se necesita ser admin para poder instalar los eggs

Se hace muy difícil trabajar con diferentes versiones de python (no se puede usar easy_install)

No se puede instalar dos versiones del mismo egg



Virtualenv al rescate

Permite crear un entorno virtual de python. Haciendo eso:

- se pueden instalar eggs sin ser administrador
- se se tiene mas de un virutalenv entonces:
 - se puede trabajar con diferentes versiones de python
 - se puede trabajar con diferentes versiones del egg

`easy_install virtualenv`

Esa es la ultima vez que van a necesitar derechos de admin.



Usando virtualenv

virtualenv nombreDelEntorno

source nombreDelEntorno/bin/activate

which python; **which** easy_install
/path/completo/al/entorno/bin/python
/path/completo/al/entorno/bin/easy_install

easy_install django

import sys; sys.path

Van a ver todas los eggs instalados hacen referencia a los eggs instalados del virtualenv

Mas info en:

- <http://tinyurl.com/virtualenv-cba2010>

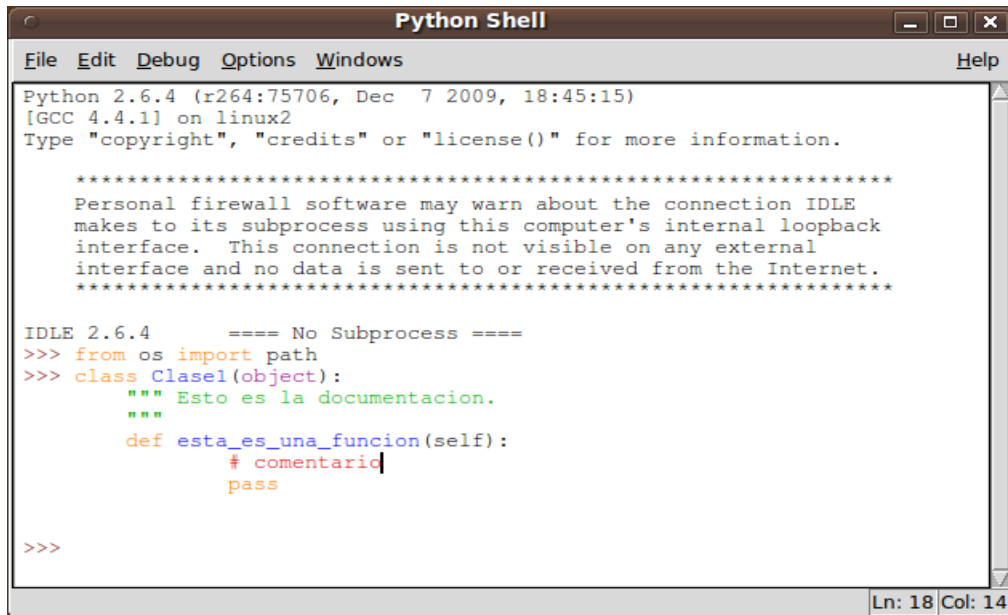
Otras consolas de python

Existen consolas gráficas para python.

- IDLE
- Dreampie

Básicamente dreampie es una versión mejorada de IDLE.

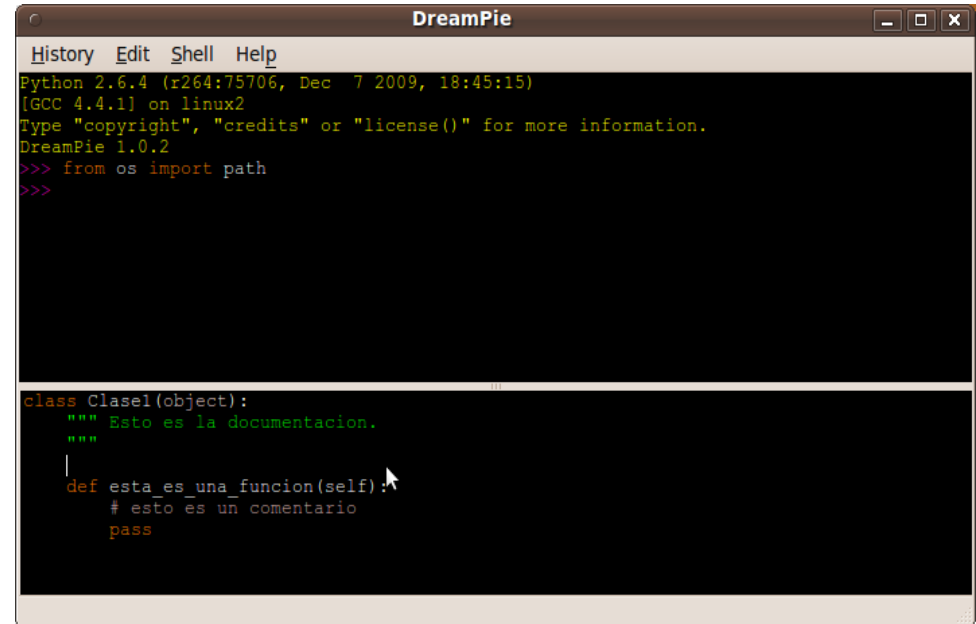
- Permite guardar el historial.
- Permite autocompletar



```
Python Shell
File Edit Debug Options Windows Help
Python 2.6.4 (r264:75706, Dec 7 2009, 18:45:15)
[GCC 4.4.1] on linux2
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 2.6.4      ==== No Subprocess ====
>>> from os import path
>>> class Clase1(object):
>>>     """ Esto es la documentacion.
>>>     """
>>>     def esta_es_una_funcion(self):
>>>         # comentario
>>>         pass
>>>
```



```
Dreampie
History Edit Shell Help
Python 2.6.4 (r264:75706, Dec 7 2009, 18:45:15)
[GCC 4.4.1] on linux2
Type "copyright", "credits" or "license()" for more information.
Dreampie 1.0.2
>>> from os import path
>>>

class Clase1(object):
    """ Esto es la documentacion.
    """
    def esta_es_una_funcion(self):
        # esto es un comentario
        pass
```

Otras consolas

Las dos consolas anteriores son alternativas gráficas a la consola de Python. Existe una alternativa que no es por consola que también es una alternativa.

Ipython

- Permite autocompletar
- Tiene el adicional “?” y “??” que muestra la documentación y el código de una función cualquiera.

easy_install ipython

```
In [1]: from sys import
_egginsert      bytearray      exc_type          getcheckinterval  gettrace          path              setcheckinterval  subversion
_plen           call_tracing   excepthook        getdefaultencoding hexversion         path_hooks        setdlopenflags    version
_clear_type_cache callstats     exec_prefix       getdlopenflags    ipcompleter       path_importer_cache setprofile         version_info
_current_frames  copyright    executable        getfilesystemencoding maxint            platform         setrecursionlimit warnoptions
_getframe        displayhook  exit              getprofile         maxsize           prefix            settrace
_api_version     dont_write_bytecode exitfunc          getrecursionlimit maxunicode        py3kwarning       stderr
_argv            exc_clear   flags             getrefcount       meta_path          pydebug           stdin
builtin_module_names exc_info     float_info        getsizeof          modules            real_prefix       stdout
```


Preguntas y Repuestas

tzulberti@gmail.com