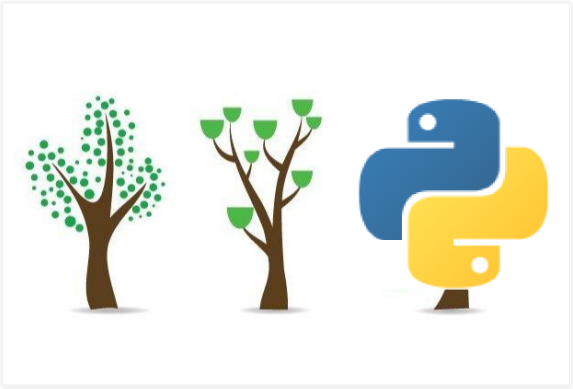
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

lunes, 23 de febrero de 2015

Entornos virtuales con Python



Introducción

Un entorno virtual Python es un espacio de trabajo creado a partir de una instalación de Python existente en un sistema; al que es posible agregar o quitar paquetes y módulos sin riesgo de dañar la instalación principal. Utilizar este recurso es muy apropiado cuando se están desarrollando varios proyectos al mismo tiempo que utilizan módulos distintos o versiones diferentes de un mismo módulo.

Tan extendido está su uso entre los desarrolladores que desde de la versión 3.3 de Python se incluye con el intérprete el módulo ***venv*** (y el script ***pyvenv.py***) para permitir gestionar los entornos virtuales de un sistema sin necesidad de instalar ningún software adicional. Para versiones anteriores de Python se puede instalar el popular módulo ***virtualenv*** que se utiliza de forma parecida al módulo *venv*.

A continuación, mostraremos la forma de crear y usar entornos virtuales y comentaremos los pasos a seguir para configurar el editor **Geany** para trabajar con distintos proyectos al mismo tiempo, que utilicen entornos virtuales diferentes con interpretes Python y módulos distintos.

Tanto si se usa ***venv*** como ***virtualenv*** los desarrolladores podrán gestionar sus propios entornos virtuales sin necesidad de contar con privilegios de administración en los sistemas.

Versiones de Python y rutas de instalación

Antes de proceder a la creación de un entorno virtual conviene conocer qué versiones de Python tenemos instaladas en el sistema:

$ python2 -V
Python 2.7.6

$ python3 -V
Python 3.4.0

También, si vamos a utilizar el módulo ***virtualenv*** será necesario identificar las rutas de instalación de los intérpretes:

- En sistemas Linux:

$ which python2
/usr/bin/python2

$ which python3
/usr/bin/python3

- En sistemas Windows la ruta de un intérprete Python la podemos consultar en la variable de entorno **PATH** o con el método **sys.executable**:

C:\> echo %PATH%

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

C:\windows\system32;C:\Python27;C:\Python27\Script;C:\Python34;C:\Python34\Scripts;C:\Program Files\Geany\bin

o desde el entorno interactivo Python

```
>>> import sys
>>> sys.executable
'C:\\Python34\\python3.exe'
```

Preparando el sistema

Tanto en Linux como en Windows para trabajar con entorno virtuales si la versión de Python que tenemos instalada es 3.3 o posterior utilizaremos el módulo ***venv***, que ya viene incluido con el intérprete. Si la versión de Python es anterior a 3.3 tendremos que instalar el módulo ***virtualenv***:

$ sudo pip3 install virtualenv

Si nuestro acceso a Internet utiliza un *proxy* que requiere autenticación, agregaremos el siguiente parámetro al comando anterior:

--proxy http://usuario:pass@servidor:puerto

Por supuesto, no hay inconveniente, hasta el momento, en instalar el módulo ***virtualenv*** en sistemas con Python 3.3 o posterior.

También, para gestionar entornos virtuales con Python 2.x instalaremos el módulo ***virtualenv***:

$ sudo pip install virtualenv

Creando y usando un entorno virtual con venv

- En Linux:

```
$ python3 -m venv virtual
$ cd virtual
$ source bin/activate
```

- En Windows:

```
C:\> python3 -m venv virtual
C:\> cd virtual
C:\virtual> Scripts\activate.bat
```

También, para crear un entorno virtual podemos utilizar el script ***pyvenv.py***:

- En Linux:

```
$ pyvenv-3.4 virtual
$ cd virtual
$ source bin/activate
```

- En Windows:

```
C:\>python3 c:\Python34\Tools\Scripts\pyvenv.py virtual
C:\> cd virtual
C:\virtual> Scripts\activate
```

En este punto ya hemos creado y activado el entorno virtual. Si al intentar crear el entorno se ha producido un error recomendamos la lectura del apartado final: "*Observaciones*".

En cambio si todo ha ido bien, realizaremos algunas operaciones en dicho entorno (Los comandos serán los mismos tanto para el sistema Linux como para Windows).

***Para consultar los módulos del entorno virtual*:**

```
(virtual) $ pip3 list
pip (6.0.8)
setuptools (12.0.5)
```

***Para añadir nuevos módulos al entorno virtual (p.e.: ipython, easygui)*:**

```
(virtual) $ pip3 install easygui
(virtual) $ pip3 install ipython
```

***Para consultar documentación de alguno de los módulos instalados*:**

```
$ python3 -m pydoc easygui
```

simultáneamente varias operaciones en el mismo espacio de proceso se...

Archivo

febrero 2015 (2)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

$ python3 -m pydoc -w ipython

***Para conocer la ruta del intérprete que estamos utilizando*:**

(virtual) $ which python3
/home/usuario/virtual/bin/python3

Para ejecutar un programa o script python desde la línea de comandos:

(virtual) $ python3 programa.py

***Para desactivar el entorno virtual*:**

- En Linux:

(virtual) $ deactivate

- En Windows:

(virtual) C:\virtual> Scripts\deactivate.bat

Creando y usando un entorno virtual con virtualenv

El módulo ***virtualenv***, como comentamos al principio, tenemos que instalarlo si la versión de Python instalada es anterior a 3.3. o si queremos utilizarlo con alguna versión 2.x.

El primer paso, en este caso, para crear un entorno virtual sería:

$ virtualenv --python=/usr/bin/python3 virtual

El argumento ***--python*** sirve para indicar a ***virtualenv*** la ruta del intérprete Python de nuestra instalación principal.

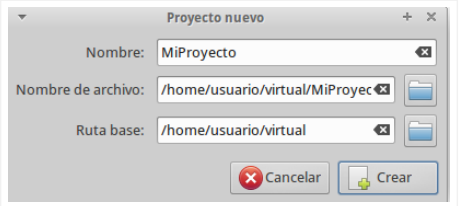
Una vez creado el entorno lo activaremos con:

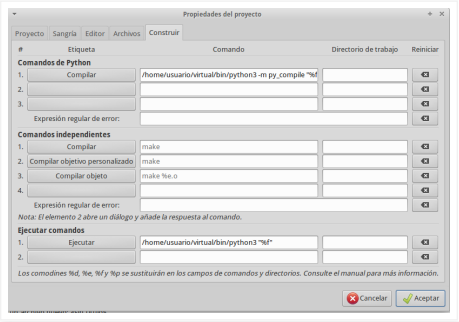
$ cd virtual
$ source bin/activate

A partir de ahora podemos utilizarlo de la misma forma que un entorno creado con el módulo ***venv***.

Configurar el editor Geany para entornos virtuales

Los usuarios del editor ***Geany*** pueden configurar sus proyectos para que utilicen el intérprete Python de cualquier entorno virtual existente en el sistema. Para ello, en ***Geany***, después de crear un nuevo proyecto *(Menú Proyecto, Nuevo)* y un primer archivo Python *(Menú Archivo, Nuevo, desde plantilla, main.py)* han de cambiarse las rutas del intérprete a utilizar, por defecto, para ejecutar o compilar el código *(Menú Construir, Establecer comandos de ejecución)*.





De esta forma cada proyecto en ***Geany*** puede utilizar el intérprete de un determinado entorno virtual y todos los módulos disponibles en el mismo.

Para conocer una ruta concreta, podemos activar antes el entorno virtual a utilizar y a continuación ejecutar el siguiente comando:

(virtual) $ which python3
/home/usuario/virtual/bin/python3

Una vez obtenida la ruta de intérprete tendríamos que utilizarla para construir los comandos de ejecución y compilación *(Menú Construir, Establecer comandos de ejecución)* para un proyecto determinado:

Compilar: **/home/usuario/virtual/bin/python3 -m py_compile "%f"**
Ejecutar: **/home/usuario/virtual/bin/python3 "%f"**

Si estamos desarrollando varios proyectos al mismo tiempo, al abrir uno, se utilizará el intérprete Python que tenga configurado.

Por último, para conocer desde un programa qué módulos tenemos disponibles en nuestro entorno virtual, ejecutar el siguiente código:

```
import pip
modulosinstalados = pip.get_installed_distributions()
for modulo in modulosinstalados:
    print(modulo)
```

Observaciones

Hay distribuciones GNU/Linux (como Ubuntu 14.04 y derivadas) que al intentar crear un entorno virtual con el módulo ***venv*** dan el siguiente error:

Error: Command '['/home/usuario/virtual/bin/python3.4', '-Im', 'ensurepip', '--upgrade', '--default-pip']' returned non-zero exit status

Este problema sucede por una mala implementación del módulo ***pip***. Si se encuentra en esta situación tiene al menos dos opciones para solucionarlo:

1.- Crear el entorno virtual omitiendo ***pip*** al principio y después instalar ***pip*** y ***setuptools***.

$ python3 -m venv --without-pip virtual
$ source ./virtual/bin/activate
$ wget https://pypi.python.org/packages/source/s/setuptools/setuptools-3.4.4.tar.gz
$ tar -vzxf setuptools-3.4.4.tar.gz
$ cd setuptools-3.4.4
$ python setup.py install
$ cd ..
$ wget https://pypi.python.org/packages/source/p/pip/pip-1.5.6.tar.gz
$ tar -vzxf pip-1.5.6.tar.gz
$ cd pip-1.5.6
$ python setup.py install
$ cd ..
$ deactivate
$ source ./virtual/bin/activate

2.- Actualizar la versión de Python

El siguiente procedimiento, bastante más lento que el anterior, actualizará Python a la versión 3.4.1 en un sistema basado en Debian (Ubuntu, Linux Mint, etc). Acceder al sistema como ***root*** y ejecutar el siguiente procedimiento:

$ sudo apt-get install libssl-dev openssl
$ wget https://www.python.org/ftp/python/3.4.1/Python-3.4.1.tgz
$ tar -xvf Python-3.4.1.tgz
$ cd Python-3.4.1/
$ ./configure
$ make
$ make install
$ python3

Para versiones más recientes acceder al sitio oficial de descargas de Python (https://www.python.org/downloads/) para conocer la ruta desde donde realizar la descarga y el nombre del archivo .tgz a descargar, o simplemente, realizar la descarga desde allí y después instalar.

Ir al índice del tutorial de Python

Publicado por Pherkad en 15:47

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua