Instalación de módulos

Python posse una activa comunidad de desarrolladores y usuarios que desarrollan toanto los módulos estándar de python, como módulos y paquetes desarolados por terceros.

PyPI y pip

- El *Python Package Index* o *PyPI*, es el repositorio de paquetes de software oficial para aplicaciones de terceros en el lenguaje de programación Python.
- `pip` : Sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python que se encuentran alojados en el repositorio *PyPI*.

Instalación de módulos python

Para instalar un nuevo paquete python tengo varias alternativas:

1. Utilizar el que este empaquetado en la distribución que estés usando. Podemos tener problemas si necesitamos una versión determinada.

   ```
   # apt-cache show python3-requests
   ...
   Version: 2.4.3-6
   ...
   ```

2. Instalar pip en nuestro equipo, y como superusuario instalar el paquete python que nos interesa. Esta solución nos puede dar muchos problemas, ya que podemos romper las dependencias entre las versiones de nuestros paquetes python instalados en el sistema y algún paquete puede dejar de funcionar.

   ```
   # pip search requests
   ...
   requests (2.13.0)      - Python HTTP for Humans.
   ...
   ```

3. Utilizar entornos virtuales: es un mecanismo que me permite gestionar programas y paquetes python sin tener permisos de administración, es decir, cualquier usuario sin privilegios puede tener uno o más "espacios aislados" (ya veremos más adelante que los entornos virtuales se guardan en directorios) donde poder instalar distintas versiones de programas y paquetes python. Para crear los entornos virtuales vamos a usar el programa `virtualenv` o el módulo `venv` .

Creando entornos virtuales con `virtualenv`

Podemos utilizar este software para trabajar con cualquier distribución de python, pero evidentemente es obligatorio si estamos trabajando con python 2.x o python 3.x (una versión anterior a la 3.3).

```
# apt-get install python-virtualenv
```

Si queremos crear un entorno virtual con python3:

```
$ virtualenv -p /usr/bin/python3 entorno2
```

La opción `-p` nos permite indicar el interprete que se va a utilizar en el entorno.

Para activar nuestro entorno virtual:

```
$ source entorno2/bin/activate
(entorno2)$
```

Y para desactivarlo:

```
(entorno2)$ deactivate
$
```

Creando entornos virtuales con `venv`

A partir de la versión 3.3 de python podemos utilizar el módulo `venv` para crear el entorno virtual.

Instalamos el siguiente paquete para instalar el módulos:

```
# apt-get install python3-venv
```

Ahora ya como un usuario sin privilegio podemos crear un entorno virtual con python3:

```
$ python3 -m venv entorno3
```

La opción `-m` del interprete nos permite ejecutar un módulo como si fuera un programa.

Para activar y desactivar el entono virtual:

```
$ source entorno3/bin/activate
(entorno3)$ deactivate
$
```

Instalando paquetes en nuestro entorno virtual

Independientemente del sistema utilizado para crear nuestro entorno virtual, una vez que lo tenemos activado podemos instalar paquetes python en él utilizando la herramienta `pip` (que la tenemos instalada automáticamente en nuestro entorno). Partiendo de un entorno activado, podemos, por ejemplo, instalar la última versión de django:

```
(entorno3)$ pip install django
```

Si queremos ver los paquetes que tenemos instalados con sus dependencias:

```
(entorno3)$ pip list
Django (1.10.5)
pip (1.5.6)
setuptools (5.5.1)
```

Si necesitamos buscar un paquete podemos utilizar la siguiente opción:

```
(entorno3)$ pip search requests
```

Si a continuación necesitamos instalar una versión determinada del paquete, que no sea la última, podemos hacerlo de la siguiente manera:

```
(entorno3)$ pip install requests=="2.12"
```

Si necesitamos borrar un paquete podemos ejecutar:

```
(entorno3)$ pip uninstall requests
```

Y, por supuesto para instalar la última versión, simplemente:

```
(entorno3)$ pip install requests
```

Para terminar de repasar la herramienta `pip`, vamos a explicar como podemos guardar en un fichero (que se suele llamar `requirements.txt`) la lista de paquetes instalados, que nos permite de manera sencilla crear otro entorno virtual en otra máquina con los mismos paquetes instalados. Para ello vamos a usar la siguiente opción de `pip`:

```
(entorno3)$ pip freeze
Django==1.10.5
requests==2.13.0
```

Y si queremos guardar esta información en un fichero que podamos distribuir:

```
(entorno3)$ pip freeze > requirements.txt
```

De tal manera que otro usuario, en otro entorno, teniendo este fichero pude reproducirlo e instalar los mismos paquetes de la siguiente manera:

```
(entorno4)$ pip install -r requirements.txt
```