# Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)



martes, 28 de julio de 2015

## Empaquetado y distribución de proyectos Python (I)

### PyPI, pip y PyPA

**PyPI** es la web con el repositorio oficial para los proyectos de la comunidad Python mantenida por la **Python Software Foundation**. PyPI está abierto a todos los desarrolladores de Python que deseen compartir sus proyectos, en forma de paquetes de distribución, con otras personas.

Normalmente, un paquete se puede descargar manualmente desde el repositorio y después instalar (**python setup.py install**); pero lo habitual es utilizar la herramienta **pip** que permite desde la consola instalar, desinstalar, actualizar, listar, buscar y realizar otras operaciones relacionadas con los proyectos publicados (en **PyPI** o en otros repositorios basados en la misma tecnología).

Cuando **pip** instala un paquete hará lo propio con sus dependencias, es decir, con aquellos paquetes de los que depende su funcionamiento. En la actualidad **Pip** es la herramienta para gestionar paquetes aconsejada por la **PyPA** (Python Packaging Authority), la principal autoridad en recomendaciones sobre empaquetado y distribución de aplicaciones Python.

A continuación, se hace un recorrido por los conceptos esenciales para introducirse en este mundo de los paquetes de distribución Python y, también, explicamos el uso de **pip**. Y en la siguiente entrada recogemos un caso práctico con todos los pasos que hay que seguir para empaquetar y distribuir un proyecto.

### Paquete de distribución

Un ***paquete de distribución*** es un archivo comprimido, identificado con un nombre y un número de versión, que contiene paquetes y módulos Python y otros archivos que se utilizan para compartir un lanzamiento.

(En este contexto un paquete de distribución o simplemente "paquete" no debe confundirse con un paquete que se importa en un programa Python).

Un paquete de distribución es el archivo, llamado comúnmente huevo (egg), que un usuario final descarga de Internet e instala en un sistema con Python.

En **PyPI** en la actualidad hay decenas de miles de paquetes y éstos pueden ser, principalmente, de dos tipos:

***Paquetes source o sdist***

Estos paquetes requieren ser compilados en la parte del usuario final. Este proceso es imprescindible cuando se incluyen extensiones escritas en lenguaje C que deben ser compiladas.

Hay casos en los que es necesario utilizar **easy_install** para instalar distribuciones construidas siguiendo el formato de los huevos de **setuptools**. **Easy_install** es un módulo que se incluyó con las herramientas **setuptools**, una biblioteca que aporta mejoras a las herramientas de distribución estándar **distutils** -utilidades de distribución- (de la biblioteca de Python) para construir y distribuir los paquetes con mayor facilidad, especialmente, aquellos con

**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

dependencias.

***Paquetes wheel***

Los paquetes **wheel** tienen un formato de distribución pre-built (de tipo binario) que ofrecen una instalación más rápida en comparación con el formato del paquete **source** o **sdist** (distribución de código fuente).

**Wheel** es un formato de paquete integrado para Python, que nace a partir de la recomendación PEP-376, con el objetivo de proporcionar una infraestructura estándar para gestionar los paquetes de distribución Python; para que las distintas herramientas que se utilizan actualmente en la instalación y desinstalación de proyectos sean interoperables.

Un archivo **wheel** es un archivo con el formato ZIP que utiliza un nombre de archivo con una nomenclatura que incluye el nombre del proyecto, su versión y que tiene la extensión **.whl**.

En este formato muchos paquetes se instalan correctamente en el momento de ser desempaquetados ganándose velocidad en el proceso.

A partir de la versión 1.5 **pip** prefiere el formato **wheel** sobre **sdist**, aunque por razones de compatibilidad gestiona los dos formatos de paquetes sin problemas. En definitiva, el formato **wheel** pretende con el tiempo sustituir al que fuera precursor de los huevos (Eggs).

## Requerimientos para instalar paquetes de distribución

A partir de Python 3.4 se incluye **pip** como gestor de paquetes por defecto y no es necesario instalar absolutamente nada, excepto en algunos casos muy concretos. También, se incluyen las herramientas **setuptools**.

***Instalar pip y setuptools***

**- Con Python 3.4 o posterior:**

Normalmente, en una instalación Python 3.4 o posterior las herramientas **pip** y **setuptools** están listas para su uso. En determinadas circunstancias (porque se haya excluido previamente de la instalación, se haya desinstalado en un momento dado o se trate de un entorno virtual) es posible que no estén disponibles. En ese caso si la instalación cuenta con el paquete **ensurepip**, instalar con:

**$ python3 -m ensurepip --upgrade**

El paquete **ensurepip** proporciona soporte para poner a punto **pip** en una instalación o en un entorno virtual Python. Es importante tener en cuenta que **pip** es un proyecto independiente que tiene su propio ciclo de lanzamientos.

Para conocer la ruta de instalación de **pip** en un sistema con GNU/Linux:

**$ which pip3**

/usr/bin/pip3

En un sistema **Windows pip** se instala en el directorio C:\Python*\Scripts\ Esta ruta tendrá que agregarse a la variable de entorno PATH para que pip se pueda invocar desde cualquier ubicación.

**- Con versiones anteriores a Python 3.4:**

Descargar el archivo get-pip.py y ejecutar con Python:

**$ python3 get-pip.py**

La ejecución anterior sirve tanto para instalar como para actualizar **pip**. Además, instalará **setuptools** si no estuviera instalado.

Una vez instalado **pip**, si deseamos actualizar **setuptools** a la última versión disponible:

**$ pip3 install -U setuptools**

**- Pip con soporte para paquetes wheel:**

Para asegurar que se tiene instalada una versión reciente de **pip** con soporte para **wheel** (permite trabajar con archivos **.whl**):

**$ pip3 install --upgrade pip**

También, se puede instalar **wheel** con:

**$ pip3 install wheel**

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

**Archivo**

julio 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

**- Otras opciones para instalar pip:**

Para Instalar en un sistema con Debian/Ubuntu:

**$ sudo apt-get install python-pip3**

Para instalar mediante un archivo **.MSI** en un sistema con Windows:

Descargar e instalar el paquete .MSI creado por Christoph Gohlke desde el sitio (no oficial) de la Universidad de California:

Desde la misma página es posible descargar **setuptools** y otros muchos paquetes de distribución Python.

***pip en entornos virtuales***

Es muy recomendable trabajar con entornos virtuales Python porque permite instalar los paquetes de distribución en un espacio aislado sin poner en riesgo la instalación principal de Python. Además, para hacerlo no es necesario tener privilegios de administración.

Para trabajar con entornos virtuales Python utilizar **virtuaenv** o **pyenv**. Si quiere conocer cómo crear y utilizar un entorno virtual: Entornos Virtuales con Python.

## Uso de pip

***Instalar un paquete***

Para instalar paquetes de distribución desde el repositorio **PyPI**:

**$ pip3 install 'paquete'**

(Si en el sistema tenemos una instalación Python 2.x y otra 3.x utilizaremos **pip** para gestionar los paquetes de 2.x y **pip3** para los 3.x).

En un sistema GNU/Linux donde se requieran privilegios de administración para realizar instalaciones, utilizar **pip** con **sudo**:

**$ sudo pip3 install 'paquete'**

Si existen varios paquetes se instalará el que tenga la versión más reciente. Las comillas son opcionales:

**$ sudo pip3 install paquete**

Para instalar en una red con acceso a Internet que utiliza **proxy** con autenticación:

**$ pip3 install paquete --proxy http://usuario:password@servidor:puerto**

Para instalar una versión especifica de un paquete:

**$ pip3 install 'paquete == version'**

Para instalar un paquete descargado previamente desde **PyPI**:

**$ pip3 install paquete.tar.gz**

ó si se trata de un paquete **wheel**:

**$ pip3 install paquete.whl**

También es posible descomprimir el paquete descargado e instalarlo después con:

**$ pip3 setup.py install**

***Desinstalar un paquete***

Para desinstalar un paquete existente en el sistema:

**$ pip3 uninstall paquete**

***Actualizar un paquete***

Para actualizar un paquete existente en el sistema:

**$ pip3 install paquete --upgrade**

***Listar paquetes instalados***

Para listar todos los paquetes instalados en el sistema Python:

**$ pip3 list**

***Mostrar información de un paquete***

Para mostrar la versión, localización y dependencias de un paquete instalado:

**$ pip3 show paquete**

Name: paquete
Version: 0.0.1
Location: /usr/local/lib/python3.4/dist-packages
Requires:

***Buscar paquetes en PyPI***

Para buscar paquetes por su nombre o por palabras que aparecen en su descripción:

**$ pip3 search literal-de-busqueda**

Ejemplos:

**$ pip3 search 'pimen'**

pimento - Simple CLI Menus

**$ pip3 search 'Menus CLI'**

***Listar paquetes de una instalación en un momento dado (Congelar)***

Para listar los paquetes que serían necesarios para replicar la instalación actual en otro sistema:

**$ pip3 freeze >requirements.txt**

El archivo de requerimientos (requirements.txt) detalla las dependencias que tiene la instalación Python.

***Leer paquetes desde archivo e instalar***

Para leer una lista de paquetes de un archivo de requerimientos (requirements.txt) e instalarlos en la instalación actual:

**$ pip3 install -r requirements.txt**

***Mostrar la versión de pip***

**$ pip3 -V**

pip 7.1.0 from /usr/local/lib/python3.4/dist-packages (python 3.4)

***Actualizar pip a la última versión disponible***

**$ pip3 install --upgrade pip**

**Mostrar ayuda de pip**

**$ pip3 -h**

***Trabajar con una caché wheel***

Para construir una caché de **wheel** en un directorio local (incluye estructura piramidal y todas las dependencias):

**$ pip3 wheel --wheel-dir=/tmp/wh pyramid**

Para instalar desde una caché **wheel** (no se usará el repositorio **Pypi**):

**$ pip3 install --use-wheel --no-index --find-links=/tmp/wh pyramid**

Para Instalar desde una caché **wheel** remotamente

**$ pip3 install --use-wheel --no-index --find-links=https://wh.web.com/ pyramid**

Siguiente: caso práctico de empaquetado de un proyecto Python y posterior publicación en PyPI para su distribución

Ir al índice del tutorial de Python

Publicado por Pherkad en 14:45

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua