Lección 15: Gestor de paquetes y entornos virtuales

Índice

Anterior

Gestor de paquetes PIP

Python pip es el gestor de paquetes de python y se utiliza para instalar/desinstalar fácilmente paquetes en una instalación de python o en un entorno virtual. No es el único gestor de paquetes de Python, pero sí el mas extendido.

Listar paquetes instalados

Para listar los paquetes instalados se utiliza *pip list*.

```
$ pip list
Package           Version
----------------- -------
Babel             2.13.0
docxcompose       1.4.0
docxtpl           0.16.7
Jinja2            3.1.2
lxml              4.9.3
MarkupSafe        2.1.3
pip               22.2.2
python-docx       1.0.1
setuptools        63.2.0
six               1.16.0
typing_extensions 4.8.0
```

Instalar paquetes

Generalmente *pip* vendrá instalado en cualquier distribución de python. Para comprobar si *pip* esta instalado se puede ejecutar:

```
$ pip --version
pip 23.2.1 from /usr/local/python/3.10.8/lib/python3.10/site-
packages/pip (python 3.10)
```

Instalar un paquete es muy sencillo, por ejemplo vamos a instalar el paquete *docxtpl*, utilizado para trabajar plantillas de documentos en docx.

```
$ pip install docxtpl
Collecting docxtpl
  Downloading docxtpl-0.16.7-py2.py3-none-any.whl (28 kB)
Collecting jinja2
```

```
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
                             133.1/133.1 kB 6.2 MB/s eta 0:00:00
Collecting python-docx
  Downloading python_docx-1.0.1-py3-none-any.whl (237 kB)
                             237.4/237.4 kB 12.3 MB/s eta 0:00:00
Collecting lxml
  Downloading lxml-4.9.3-cp310-cp310-manylinux_2_28_x86_64.whl (7.9 MB)
                             7.9/7.9 MB 66.6 MB/s eta 0:00:00
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting docxcompose
  Downloading docxcompose-1.4.0.tar.gz (20 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: setuptools in ./venv/lib/python3.10/site-
packages (from docxcompose->docxtpl) (63.2.0)
Collecting babel
  Downloading Babel-2.13.0-py3-none-any.whl (10.1 MB)
                             10.1/10.1 MB 56.1 MB/s eta 0:00:00
Collecting typing-extensions
  Downloading typing_extensions-4.8.0-py3-none-any.whl (31 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2
Using legacy 'setup.py install' for docxcompose, since package 'wheel' is not installed.
Installing collected packages: typing-extensions, six, MarkupSafe, lxml, babel, python-
docx, jinja2, docxcompose, docxtpl
  Running setup.py install for docxcompose ... done
Successfully installed MarkupSafe-2.1.3 babel-2.13.0 docxcompose-
1.4.0 docxtpl-0.16.7 jinja2-3.1.2 lxml-4.9.3 python-docx-1.0.1 six-
1.16.0 typing-extensions-4.8.0
```

Obtener información sobre un paquete

Se obtiene información sobre un paquete con *pip show*.

```
$ pip show Jinja2
Name: Jinja2
Version: 3.1.2
Summary: A very fast and expressive template engine.
Home-page: https://palletsprojects.com/p/jinja/
Author: Armin Ronacher
Author-email: armin.ronacher@active-4.com
License: BSD-3-Clause
Location: /workspaces/curso_python/15_EntornosVirtuales/venv/lib/python3.10/site-
packages
Requires: MarkupSafe
Required-by: docxtpl
```

Desinstalar paquetes

```
$ pip uninstall docxtpl
```

Entornos virtuales

Los entornos virtuales permiten crear una instalación de python aislada de la instalación principal en nuestro sistema e instalar paquetes de forma independientes. Esto facilita la gestión de dependencias cuando se esta desarrollando un proyecto, especialmente si se tiene pensado distribuir la aplicación a otros usuarios.

Ventajas: - Tener varios entornos, con varios conjuntos de paquetes, sin conflictos entre ellos. De esta manera, los requisitos de diferentes proyectos se pueden satisfacer al mismo tiempo. - Lanzar fácilmente proyectos con sus propios módulos dependientes. - Permite gestionar los paquetes necesarios para el proyecto independientemente de otros proyectos y de la instalación principal de python del sistema. - Distribuir las aplicaciones con el mínimo de paquetes necesarios para su funcionamiento y en las versiones correctas.

Crear un entorno virtual y activarlo

Una buena práctica al iniciar un proyecto, es crear un entorno virtual en la carpeta del proyecto.

```
$ cd proyecto1
$ python -m venv my_venv
```

Se creará una carpeta *my_venv* que contiene el entorno virtual (una instalación de python independiente de otros entornos y de la instalación principal de python)

Despues de crear el entorno virtual es necesario activarlo para trabajar en el entorno virtual.

En linux:

```
$ source my_venv\bin\activate
(my_venv) $
```

En Windows PowerShell:

```
C:\Users\User\Documents\user1\proyecto1> my_venv\Scripts\activate
(my_venv) C:\Users\User\Documents\user1\proyecto1>
```

En Windows Cmd:

```
C:\Users\User\Documents\user1\proyecto1> my_venv\Scripts\Activate.bat
(my_venv) C:\Users\User\Documents\user1\proyecto1>
```

Una vez creado y activado, se pueden instalar los paquetes necesarios para el proyecto que se está desarrollando con *pip install nombrepaquete*.

Fichero requirements.txt

En general los entornos virtuales ocupan bastante espacio y tienen muchos ficheros, por lo que no suelen distribuirse con el código de las aplicaciones o cuando se publican en GitHub. En su lugar, lo que se incluye es un fichero *requirements.txt* con la lista de paquetes necesarios y su versión.

Una vez activado el entorno virtual e instalados los paquete necesarios para el proyecto, se puede crear el fichero *requirements.txt* con *pip freeze*.

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

Si el fichero *requirements.txt* está disponible, se pueden instalar todos los paquetes necesarios de una sola vez con el siguiente comando:

```
(my_venv) C:\Users\User\Documents\user1\proyecto1> pip install -
r requirements.txt
```

Siguiente