Python pip

What is pip?

`pip` is the standard package manager for Python. We can use `pip` to install additional packages that are not available in the Python standard library. For example,

```
pip install numpy
```

If we had installed `pip` on our system, this command would have installed the `numpy` library.

How to install pip?

`pip` comes pre-installed on the Python versions 3.4 or older. We can check if `pip` is installed by using the following command in the console:

```
pip --version
```

If `pip` is already available in the system, the respective `pip` version is displayed, like:

```
pip 19.3.1 from C:\Python37\lib\site-packages\pip (python 3.7)
```

If we are using an older version of Python or do not have `pip` installed for some other reason, follow the steps as described in this link: pip installation

---

Using pip

`pip` is a command-line program. After its installation, a `pip` command is added which can be used with the command prompt.

The basic syntax of pip is:

```
pip <pip arguments>
```

Installing Packages with pip

Apart from the standard Python library, the Python community contributes to an extensive number of packages tailored for various development frameworks, tools, and libraries.

Most of these packages are officially hosted and published to the Python Package Index(PyPI). `pip` allows us to download and install these packages.

Basic Package Installation

The `install` command used to install packages using `pip`. Let's take an example:

Suppose we want to install `requests`, a popular HTTP library for Python. We can do it with the help of the following command.

```
pip install requests
```

Output

```
Collecting requests
  Using cached https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl
Collecting chardet<3.1.0,>=3.0.2
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Using cached https://files.pythonhosted.org/packages/b4/40/a9837291310ee1ccc242ceb6ebfd9eb21539649f193a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
Collecting idna<2.9,>=2.5
  Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
Collecting certifi>=2017.4.17
  Downloading https://files.pythonhosted.org/packages/b9/63/df50cac98ea0d5b006c55a399c3bf1db9da7b5a24de7890bc9cfd5dd9e99/certifi-2019.11.28-py2.py3-none-any.whl (156kB)
Installing collected packages: chardet, urllib3, idna, certifi, requests
Successfully installed certifi-2019.11.28 chardet-3.0.4 idna-2.8 requests-2.22.0 urllib3-1.25.7
```

Here, we can see that the `pip` has been used with the `install` command followed by the name of the package we want to install (`requests`).

All other dependencies like `chardet`, `urllib3` and `certifi` required for this package are also installed by `pip`.

Specifying Package Version

When `pip install` is used in its minimal form, `pip` downloads the most recent version of the package.

Sometimes, only a specific version is compatible with other programs. So, we can define the version of the package in the following way:

```
pip install requests==2.21.0
```

Here, we have installed the *2.11.0* version of the `requests` library.

Listing Installed Packages with pip

The `pip list` command can be used to list all the available packages in the current Python environment.

```
pip list
```

Output

```
Package    Version
---------- ----------
certifi    2019.11.28
chardet    3.0.4
idna       2.8
pip        19.3.1
requests   2.22.0
setuptools 45.0.0
urllib3    1.25.7
wheel      0.33.6
```

Package Information with pip show

The `pip show` command displays information about one or more installed packages. Let's look at an example:

```
pip show requests
```

Output

```
Name: requests
```

```
Version: 2.22.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: [emailÂ protected]
License: Apache 2.0
Location: c:\users\dell\desktop\venv\lib\site-packages
Requires: certifi, chardet, urllib3, idna
Required-by:
```

Here, the `show` command displays information about the `requests` library. Notice the **Requires** and **Required-by** column in the above output.

Requires column shows which dependencies the `requests` library requires. And, **Required-by** column shows the packages that require `requests`.

Uninstalling a Package with pip

We can uninstall a package by using `pip` with the `pip uninstall` command.

Suppose we want to remove the `requests` library from our current Python environment. We can do it in the following way:

```
pip uninstall requests
```

Output

```
Uninstalling requests-2.22.0:
  Would remove:
    C:\Python37\lib\site-packages\requests-2.22.0.dist-info\*
    C:\Python37\lib\site-packages\requests\*
Proceed (y/n)? y
  Successfully uninstalled requests-2.22.0
```

As we can see, the `requests` package is removed after the final prompt.

Note: Even though the specified package is removed, the packages that were installed as dependencies are not removed. In this case, the dependencies (`chardet`, `urllib3`, and `certifi)` of the `requests` library aren't uninstalled.

If we need to remove the dependencies of a package as well, we can use the `pip show` command to view installed packages and remove them manually.

Using Requirement Files

A file containing all the package names can also be used to install Python packages in batches.

Let's take a look at an example:

Suppose we have a file **requirements.txt** which has the following entries:

```
numpy
Pillow
pygame
```

We can install all these packages and their dependencies by using a single command in `pip`.

```
pip install -r requirements.txt
```

Output

```
Collecting numpy
  Using cached https://files.pythonhosted.org/packages/a9/38/f6d6d8635d496d6b4ed5d8ca4b9f193d0edc59999c3a63779cbc38aa650f/numpy-1.18.1-cp37-cp37m-win_amd64.whl
Collecting Pillow
  Using cached https://files.pythonhosted.org/packages/88/6b/66f502b5ea615f69433ae1e23ec786b2cdadbe41a5cfb1e1fabb4f9c6ce9/Pillow-7.0.0-cp37-cp37m-win_amd64.whl
Collecting pygame
  Using cached https://files.pythonhosted.org/packages/ed/56/b63ab3724acff69f4080e54c4bc5f55d1fbdeeb19b92b70acf45e88a5908/pygame-1.9.6-cp37-cp37m-win_amd64.whl
Installing collected packages: numpy, Pillow, pygame
Successfully installed Pillow-7.0.0 numpy-1.18.1 pygame-1.9.6
```

Here, we have used the same `install` command with `pip`.

However, the additional argument `-r` specifies `pip` that we are passing a requirements file rather than a package name.

Creating Requirements File

As an alternative to manually creating the requirements file, `pip` offers the `freeze` command. Let's look at how to use this command.

Suppose our current Python environment has the following packages. It can be displayed using `pip list`.

```
Package    Version
---------- -------
numpy      1.17.0
Pillow     6.1.0
pip        19.3.1
pygame     1.9.6
setuptools 45.0.0
wheel      0.33.6
```

The packages that don't come preinstalled will Python are listed using the `freeze` command.

```
pip freeze
```

Output

```
numpy==1.17.0
Pillow==6.1.0
```

```
pygame==1.9.6
```

The `pip freeze` command displays the packages and their version in the format of the requirements file.

So this output can be redirected to create a requirements file using the following command:

```
pip freeze > requirements.txt
```

A new **requirements.txt** file is created in the working directory. It can later be used in other Python environments to install specific versions of packages.

Search packages in pip

The `search` command is used to search for packages in the command prompt. Let's look at an example:

```
pip search pygame
```

Output

```
pygame-anisprite (1.0.0)              - Animated sprites for PyGame!
pygame-ai (0.1.2)                     - Videogame AI package for PyGame
pygame-engine (0.0.6)                 - Simple pygame game engine.
pygame-assets (0.1)                   - Assets manager for Pygame apps
pygame-gui (0.4.2)                    - A GUI module for pygame 2
pygame-spritesheet (0.2.0)            - Python pygame extension that provides SpriteSheet class.
pygame-minesweeper (1.0)              - Minesweeper game implemented in python using pygame
pygame-menu (2.1.0)                   - A menu for pygame, simple, lightweight and easy to use
pygame-plot (0.1)                     - Quick visualization of data using pygame with a matplotlib style
pygame (1.9.6)                        - Python Game Development
...
```

Here, we have searched for a library called `pygame`. All other packages that match the keyword are displayed. This command is helpful for finding related packages.

To learn more about `pip`, visit: Python pip (official documentation)

Table of Contents