*VirtualEnv*

[virtualenv](http://pypi.python.org/pypi/virtualenv) is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executables to use the packages that a Python project would need. It can be used standalone, in place of Pipenv.

virtualenv my\_project will create a folder in the current directory which will contain the Python executable files, and a copy of the pip library which you can use to install other packages. The name of the virtual environment (in this case, it was my\_project) can be anything; omitting the name will place the files in the current directory instead.

This creates a copy of Python in whichever directory you ran the command in, placing it in a folder named my\_project.

To delete a virtual environment, just delete its folder. (In this case, it would be rm -rfmy\_project.)

After a while, though, you might end up with a lot of virtual environments littered across your system, and its possible you’ll forget their names or where they were placed.

Running virtualenv with the option --no-site-packages will not include the packages that are installed globally. This can be useful for keeping the package list clean in case it needs to be accessed later. [This is the default behavior for virtualenv 1.7 and later.]

In order to keep your environment consistent, it’s a good idea to “freeze” the current state of the environment packages.

virtualenvwrapper

[virtualenvwrapper](https://virtualenvwrapper.readthedocs.io/en/latest/index.html) provides a set of commands which makes working with virtual environments much more pleasant. It also places all your virtual environments in one place.

To install (make sure **virtualenv** is already installed):

**virtualenvwrapper** provides tab-completion on environment names. It really helps when you have a lot of environments and have trouble remembering their names.

workon also deactivates whatever environment you are currently in, so you can quickly switch between environments.