Virtual Environments

A Virtual Environment, put simply, is an isolated working copy of Python which allows you to work on a specific project without worry of affecting other projects.

For example, you can work on a project which requires Django 1.3 while also maintaining a project which requires Django 1.0.

virtualenv

virtualenv is a tool to create isolated Python environments.

Install it via pip:

```
$ pip install virtualenv
```

Basic Usage

1. Create a virtual environment:

```
$ virtualenv venv
```

This creates a copy of Python in whichever directory you ran the command in, placing it in a folder named `venv`.

2. To begin using the virtual environment, it needs to be activated:

```
$ source venv/bin/activate
```

You can then begin installing any new modules without affecting the system default Python or other virtual environments.

3. If you are done working in the virtual environment for the moment, you can deactivate it:

```
$ deactivate
```

This puts you back to the system's default Python interpreter with all its installed libraries.

To delete a virtual environment, just delete its folder.

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.

virtualenvwrapper

virtualenvwrapper 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):

```
$ pip install virtualenvwrapper
$ export WORKON_HOME=~/Envs
$ source /usr/local/bin/virtualenvwrapper.sh
```

(Full virtualenvwrapper install instructions.)

For Windows, you can use the virtualenvwrapper-powershell clone.

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

```
PS> pip install virtualenvwrapper-powershell
PS> $env:WORKON_HOME="~/Envs"
PS> mkdir $env:WORKON_HOME
PS> import-module virtualenvwrapper
```

Basic Usage

1. Create a virtual environment:

```
$ mkvirtualenv venv
```

This creates the `venv` folder inside `~/Envs`.

2. Work on a virtual environment:

```
$ workon venv
```

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.

3. Deactivating is still the same:

```
$ deactivate
```

4. To delete:

```
$ rmvirtualenv venv
```

Other useful commands

`lsvirtualenv`
: List all of the environments.

`cdvirtualenv`
: Navigate into the directory of the currently activated virtual environment, so you can browse its `site-packages`, for example.

`cdsitepackages`
: Like the above, but directly into `site-packages` directory.

`lssitepackages`
: Shows contents of `site-packages` directory.

Full list of virtualenvwrapper commands.

autoenv

When you `cd` into a directory containing a `.env` autoenv automagically activates the environment.

Install it on Mac OS X using `brew`:

```
$ brew install autoenv
```

And on Linux:

```
$ git clone git://github.com/kennethreitz/autoenv.git ~/.autoenv
$ echo 'source ~/.autoenv/activate.sh' >> ~/.bashrc
```