

Python Virtual Environments

What and Why

A Virtual Environment is a tool to keep the dependencies required by different projects in separate places, by creating virtual Python environments for them. It solves the “Project X depends on version 1.x but, Project Y needs 4.x” dilemma, and keeps your global site-packages directory clean and manageable. For example, you can work on a project which requires Django 1.3 while also maintaining a project which requires Django 1.0. ¹⁾

Using Virtual Environments

The python virtual environment package is installed by default on most CS machines. To make sure you can do the following:

```
user@computer:~/projects$ which virtualenv
/usr/bin/virtualenv
```

Creating a new project

```
user@hester:~/projects$ virtualenv --no-site-packages exampleproject
The --no-site-packages flag is deprecated; it is now the default behavior.
New python executable in exampleproject/bin/python
Installing
distribute.....
.....done.
Installing pip.....done.
```

```
user@computer:~/projects$ ls -l exampleproject/
total 4
drwxrwxr-x 2 user group 4096 Nov 25 10:16 bin
drwxrwxr-x 2 user group  30 Nov 25 10:16 include
drwxrwxr-x 3 user group  30 Nov 25 10:16 lib
drwxrwxr-x 2 user group  56 Nov 25 10:16 local
```

```
user@computer:~/projects$ cd exampleproject/
```

Activate

Now you need to activate your virtual environment. This will setup some path variables to make the environments bin and lib directory to be the default.

```
user@computer:~/projects/exampleproject$ source bin/activate
(exampleproject)user@computer:~/projects/exampleproject$
```

Deactivate

To deactivate or stop working on your environment use the function that gets sourced when you activate your environment:

```
(exampleproject)user@computer:~/projects/exampleproject$ deactivate
user@computer:~/projects/exampleproject$
```

virtualenvwrapper

virtualenvwrappers goal is to make virtualenv easier to use... sort of like when iTunes automatically organizes your iTunes library.

Basic Usage

Create a virtual environment

This creates the exampleproject folder inside ~/Envs.

```
user@computer:~/ $ mkvirtualenv venv
```

Work on a virtual environment

'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.

```
user@computer:~/ $ workon venv
```

Deactivating is still the same:

```
user@computer:~/ $ deactivate
```

To delete:

```
user@computer:~/ $ 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.
ls sitepackages
Shows contents of site-packages directory.
```

1)
<http://docs.python-guide.org/en/latest/dev/virtualenvs>

From:
<https://howto.cs.uchicago.edu/> - **How do I?**

Permanent link:
https://howto.cs.uchicago.edu/python:virtual_environments?rev=1416946440

Last update: **2014/11/25 14:14**

