Python Virtual Environment

User Level Development

Python based developing and testing can easily be done of new or experimental packages in a python virtual environment which is separate from the system python and therefore any failures or bugs will not affect other users.

Installing python virtual environment

- Install python virtual environment

```
> sudo aptitude install python-virtualenv
> sudo aptitude install virtualenvwrapper
```

- Create a directory for using to house virtual environments

```
> mkdir testdevels
> cd testdevels
```

There are multiple ways of using virtual environments. In the following section I shall describe the two methods I use depending on needs.

Using the python virtual environment

Quick environment

For running and testing example code that will be discarded (such as workshops and conferences)

Setting up a python virtual environment

- Create your working directory with the virtual environment

```
> mkdir devel
> cd devel
```

- Initiate the environment and include python system `site-packages` so you do not need to install large packages such as numpy for every environment.

```
> virtualenv --system-site-packages venv
```

The main difference between this method of starting the environment and the method below, is that the entire environment is in the **`venv`** directory (`bin/, lib/, include/, local/`).

You need to take care that when you are addressing installs to this environment, that you address the pip and python, etc in the `venv/bin/` directory.

Using a python virtual environment

You can either enable and disable the environment as needed, or you can address the virtual install options directly.

- Active a virtual environment

```
> source venv/bin/activate
```

- Do some stuff
- When done, deactivate the virtual environment to return to system python

```
> deactivate
```

Things to do:

- Check that you use virtual installs

```
> which pip
```

- Else, simply address the binaries from the virtual environment

```
> venv/bin/pip install docopt
> ls /venv/lib/python2.7/site-packages
```

- Ensure the executable of your script is set to the correct python

```
> head test.py
```

- Or, verify your python when the environment is active

```
> which python
```

To remove the virtual environment, simple delete the virtual directory to remove from the system.

Development environment

For software package and system development

Setting up a python virtual environment

- Add the following lines to your shell startup file (.bashrc, .profile, etc.) to set the location where the virtual environments should live, the location of your development project directories, and the location of the script installed with this package:

```
export WORKON_HOME=$HOME/.virtualenvs
possible_scripts='/usr/local/bin/virtualenvwrapper.sh /etc/bash_completion.d/virtualenvwrapper'
for script in $possible_scripts; do
  [[ -f $script ]] && source $script
done
```

- After editing it, reload the startup file

```
> source ~/.bashrc
```

- Create your working directory

```
> mkdir testenv
> cd testenv
```

- Start the virtual environment, but include python system `site-packages` so you do not need to install large packages such as numpy for every environment.

```
> mkvirtualenv --system-site-packages testenv
> setvirtualenvproject
```

Using a python virtual environment

You can either enable and disable the environment as needed, or you can address the virtual install options directly.

```
> cd testenv
```

- To start working on the virtual environment

```
> source venv/bin/activate
```

- To stop working in the virtual environment

```
> deactivate
```

When done, remember to deactivate the virtual environment to return to system python

Any system packages can be imported directly, and for local installations use easy_install or pip install options.

```
> cd <your_package>
> pip install .
```

or for development installations

```
> pip install --editable .
```

To remove the virtual environment, simple delete the virtual directory to remove from the system.