virtualenv

Python Virtual Environment Setup

To have a good control over pre-requisite library modules for our software, we run in a python virtual environment.

That means that whenever you want to use or maintain the software you must first activate that virtual environment like this: (replace 'generic' with an existing project environment name)

source \${HOME}/venv_generic/bin/activate

Once active you can deactivate it like this:

deactivate

Installation

We are going to install any non pip packages we rely on, if any, first, then install virtualenv. Then create a virtual environment. Then activate it.

Install virtualenv itself.

```
sudo apt update
sudo apt install virtualenv
```

Install a .rst to .pdf converter.

snap install rst2pdf

Create a virtual environment for the project

virtualenv --prompt venv_generic \${HOME}/venv_generic

Then activate the created environment

source \${HOME}/venv_generic/bin/activate

Now we install modules our project will need, first pip itself and the tools it needs.

```
python3 -m pip install -U pip
pip install wheel
pip install setuptools
pip install twine
```

We are using flit for building and installling our software.

pip install flit

Saving prerequisites

We keep our dependencies under version control, so each time we install more components we need to refresh our requirements file.

pip freeze >\${HOME}/allrepos/generic/requirements.txt

Reinstalling

Later, when we are migrating to a different workstation, or for some reason need to restablish our project from scratch, we can install all our prerequisites in one shot instead of installing items one at a time like we did above.

```
pip install -r ${HOME}/allrepos/generic/requirements.txt
```

Automatic activation

In many cases we want to activate a project whenver we log in, so lets do that automatically.

Append this snippet to the end of our "\${HOME}/.bashrc" file.

```
#
# Added for automatic vitual environment activation
#
if [ -f "${HOME}/venv_generic/bin/activate" ] ; then
    source ${HOME}/venv_generic/bin/activate
fi
```

Good ignoring keeps flit happy

Flit works nicely with git, but it is quite strict, so we must ignore all irrelevant files properly or flit will reject builds.

Below is a good start for a project ".gitignore" file:

~ \#\# dist/* pdf_docs/* html_docs/*