Python virtual environments

for Python 2.7 ***for Python 3.7***

Software installation

`pip install virtualenv` Nothing to do: it's built into Python3

Creating a virtual environment

`virtualenv <`*`subdir`*`>` `python3 -m venv <`*`subdir`*`>`

This creates a subdirectory with the given name and installs a basic Python environment into it. It's common to name the subdirectory `venv` but if you can name it whatever you want.

Activating your virtual environment in `bash`

`source <`*`subdir`*`>/bin/activate`

This will run a script that modifies your `$PATH` variable so that all of your Python tooling runs out of this subdirectory, instead of the usual places. You will notice that your prompt is modified to mention the virtual-environment subdirectory.

From here on, you can use `pip` and other Python tools as you usually would. Any installations go only into this subdirectory and don't affect any other Python software on your computer.

This instruction is for `bash`. There are ways to set it up for other shells, but you're on your own there.

Deactivating your virtual environment

`deactivate`

This will run a script to restore your `$PATH` variable to forget about this virtual-environment subdirectory and to reset your prompt. Now if you try to use Python tools, you'll get the system-wide ones again.

Basic versioning

Installing packages with `pip`

Make sure you've activated your environment and type

`pip install <package>`

If you need a specific version

`pip install <package>==<version>`

Saving your version configuration

`pip freeze > <requirements-file>`

It's common to name the requirements file `requirements.txt`.

Reinstalling the same versions

`pip install -r <requirements-file>`

Python packaging

The tools for bundling up Python code for deliveries to others has evolved crazily over the years, from

- just sending a tarfile of source, to
- the `distutils` package and a `setup.py` file to install, to
- the `setuptools` package with the Egg format and the `easy_install` command.
- `setuptools` later added the Wheel format.
- `setuptools` was forked as `distribute`, but its improvements were folded back into `setuptools` and `distribute` was then abandoned.
- Work began on `distutils2` which stores metadata in a `setup.cfg` file, it was renamed as `packaging`, then abandoned.
- A package called `distlib` which includes some goodies from `packaging` *might* become standard in the future, but it's now.

So trying to read up on packaging is terribly confusing.

For now, `setuptools` is the way to go, so if you have to google for a packaging question, ignore stuff relating to other tools.

Installing `setuptools`

```
pip install setuptools
```

Supplying package metadata for `setuptools`

You can supply metadata (name, version number, licence, dependencies, etc.) either in Python code (`setup.py`) or in plain text (`setup.cfg`). Both these files go in the top directory of your package. Both ways have advantages. Text is usually easier to enter and can be automated, but Python code can calculate some of the values.

To supply metadata in code, the simplest version would be:

```
from setuptools import setup, find_packages

setup(
    name="<package-name>",
    version="<package-version>",
    packages=find_packages(),
    ...and any other metadata needed
)
```

Note: `find_packages()` is a function call to automate finding which packages exist by walking the directory tree.

To supply metadata in text, your `setup.py` file should look like this:

```
from distutils.core import setup

setup()
```

and your `setup.cfg` will contain

```
[metadata]
name = <package-name>
version = <package-version>

[options]
packages = find:
```

You can find descriptions of the format and useful metadata at:

https://setuptools.readthedocs.io/en/latest/setuptools.html

To install directly from a GitHub repository

```
pip install git+https://github.com/nedervold/testomundo.git
```

If the repository contains a `setup.py` (and possibly `setup.cfg`), it can be installed directly.

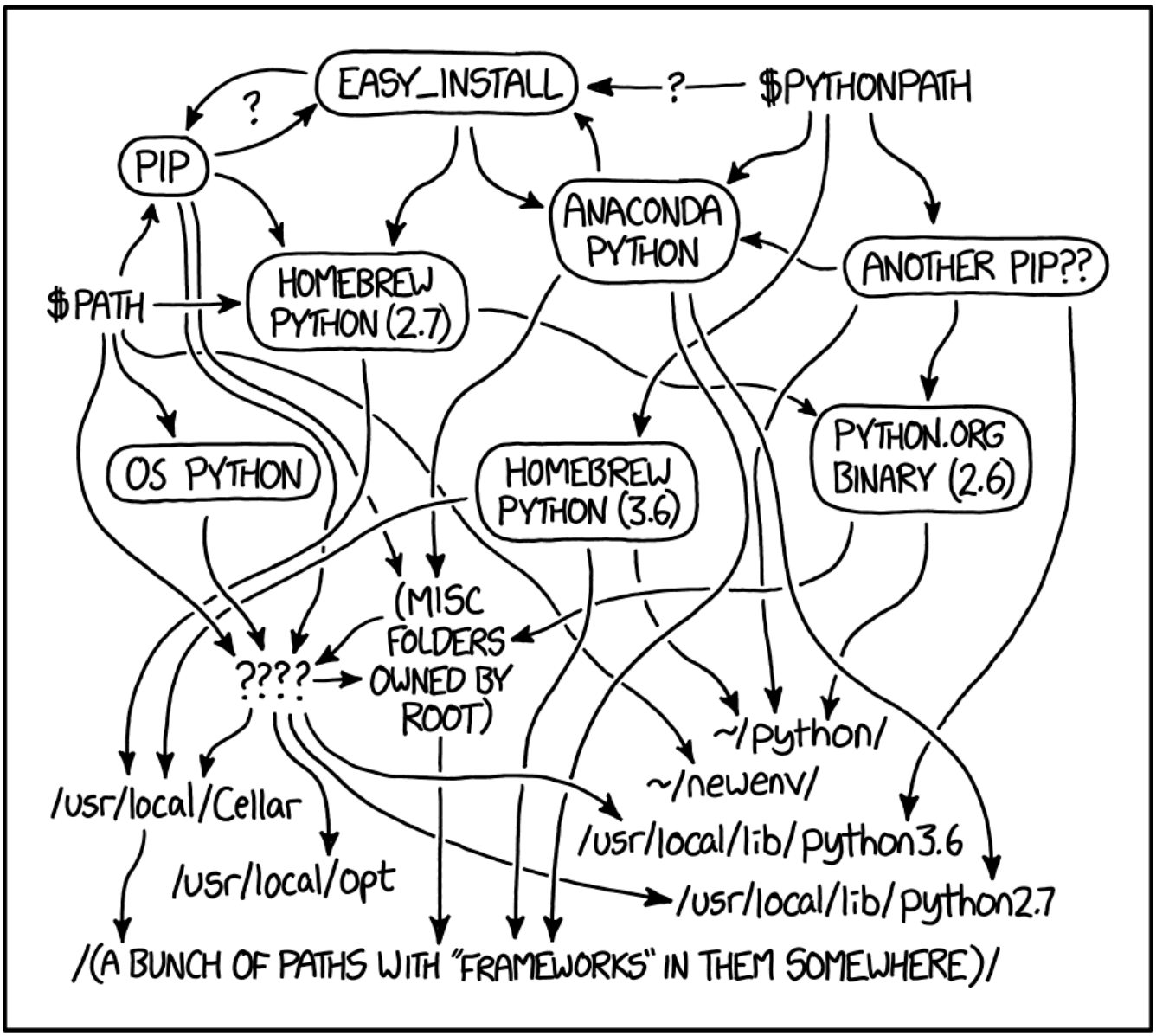
To build a source distribution file

```
python setup.py sdist
```

More information

See:

- The Python packaging user guide https://packaging.python.org
- How to package your Python code https://python-packaging.readthedocs.io/en/latest/



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.