Python Virtual Environments

or

How not to stuff things up

Why?

- Different projects, different dependencies
- Testing different library versions
- Isolate host system
- System libraries outdated

How?

- venv module

  https://docs.python.org/3/library/venv.html

- virtualenv

  https://github.com/pypa/virtualenv

- anaconda

  https://www.anaconda.com/products/individual

venv module

- since Python 3.5
- built-in module (“batteries included”), though separate package in Ubuntu (“python3-venv”)
- limited to Python version it is part of
- command-line

```
python3 -m venv <venv_path>
```

Example

```
fracpete@venv:~$ python3.7 -m venv venv37

fracpete@venv:~$ . ./venv37/bin/activate

(venv37) fracpete@venv:~$ pip freeze

pkg-resources==0.0.0

(venv37) fracpete@venv:~$ pip install numpy

Collecting numpy

  Using cached
https://files.pythonhosted.org/packages/65/b9/0b02ffd2689cbfa5d1da09a59378b626768386add3b654718d43d97e0ef1/
numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl

Installing collected packages: numpy

Successfully installed numpy-1.20.1

(venv37) fracpete@venv:~$ deactivate
```

virtualenv

- more powerful than venv module

  faster, extendable, can use any Python version, ...
- Debian: sudo apt-get install virtualenv
- Other OS: install via pipx (see last slide)
- Command-line:

```
virtualenv -p <python_bin> <venv_path>
```

Example

```
fracpete@venv:~$ virtualenv -p /usr/bin/python3.7 virtualenv37

fracpete@venv:~$ . ./virtualenv37/bin/activate

(virtualenv37) fracpete@venv:~$ pip freeze

pkg-resources==0.0.0

(virtualenv37) fracpete@venv:~$ pip install numpy

Collecting numpy

  Downloading numpy-1.20.1-cp37-cp37m-manylinux2010_x86_64.whl (15.3 MB)

     |████████████████████████████████| 15.3 MB 10.6 MB/s

Installing collected packages: numpy

Successfully installed numpy-1.20.1

(virtualenv37) fracpete@venv:~$ deactivate
```

anaconda

- Advertised as *data science toolkit* (Python/R)
- For long time best option under Windows (compilation is a pain!)
- But many unofficial binaries now available:

  https://www.lfd.uci.edu/~gohlke/pythonlibs/
- Massive download (and much larger installation): 400-500MB!
- Modifies $HOME/.bashrc
- Packages need to be installed through their channels, otherwise environments cannot be cloned
- Command-line tool “conda” manages environments

```
conda create -n <env_name> python=python3.X
```

Example

```
(base) fracpete@venv:~$ conda create -n conda37 python=3.7
Collecting package metadata (current_repodata.json): done
...
(base) fracpete@venv:~$ conda activate conda37
(conda37) fracpete@venv:~$ pip freeze
certifi==2020.12.5
(conda37) fracpete@venv:~$ pip install numpy  # or: conda install numpy
Collecting numpy
  Using cached numpy-1.20.1-cp37-cp37m-manylinux2010_x86_64.whl (15.3 MB)
Installing collected packages: numpy
Successfully installed numpy-1.20.1
(conda37) fracpete@venv:~$ pip freeze
certifi==2020.12.5
numpy==1.20.1
(conda37) fracpete@venv:~$ conda deactivate
```

Let’s give it a spin!

Other tools

- Python Poetry

  Poetry is a tool for dependency management and packaging in Python. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.

  https://python-poetry.org/

- pipx

  pipx is made specifically for application installation, as it adds isolation yet still makes the apps available in your shell: pipx creates an isolated environment for each application and its associated packages.

  https://pypi.org/project/pipx/