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- System libraries outdated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/distribution

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
- Windows: pip install virtualenv (but tied to Python version)
- Other OS: install via pipx (see "Other tools")
- 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 "world’s most popular open-source Python distribution platform"
- For long time best option under Windows (compilation is a pain!)
- Download >500MB, modifies $HOME/.bashrc
- Packages to be installed through their channels, otherwise envs cannot be cloned
- Command-line tool “conda” manages environments (C:\ProgramData\anaconda3\Library\bin)

```
conda create -n <env_name> python=3.X
```

- Many libraries upload platform-specific .whl files to PyPI nowadays
- Unofficial binaries available for Windows (563 as of 11/4/22):

  https://www.lfd.uci.edu/~gohlke/pythonlibs/
- Or use WSL2 and a proper Linux distro?

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
```

Python on Windows

- Install Microsoft Build Tools
  - 2017: https://aka.ms/vs/15/release/vs_buildtools.exe
  - 2019: https://aka.ms/vs/16/release/vs_buildtools.exe
  - 2022: https://aka.ms/vs/17/release/vs_buildtools.exe
- Download Python from python.org
- No pip?
  - Download it: https://bootstrap.pypa.io/get-pip.py
  - Install it: python get-pip.py
- But: you may need other dependencies for compiling as well...

Let’s give it a spin!

Managing dependencies?

- Installing via "pip install ..." may work now...
- Use requirements.txt to store "name==version"
- Use: pip freeze > requirements.txt
- Store requirements.txt under version control
- Recreate with: pip install -r requirements.txt

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/

Other tools (2)

- pipenv

  a tool that aims to bring the best of all packaging worlds (bundler, composer, npm, cargo, yarn, etc.) to the Python world. Windows is a first-class citizen, in our world.

  https://pipenv.pypa.io/

But...

- What about vulnerabilities?

  safety

  checks your installed dependencies for known security vulnerabilities.

  By default it uses the open Python vulnerability database Safety DB, but can be upgraded to use pyup.io's Safety API using the --key option.

  https://github.com/pyupio/safety

Safety

- create empty virtual environment
- install old (and unsafe) django version

  ./venv/bin/pip install "django<2.0"
- install safety

  ./venv/bin/pip install safety
- check environment

  ./venv/bin/safety

Safety output

```
+==============================================================================+
|                                                                              |
|                              /$$$$$$            /$$                          |
|                             /$$__  $$          | $$                          |
|           /$$$$$$$  /$$$$$$ | $$  \__//$$$$$$  /$$$$$$   /$$   /$$          |
|          /$$_____/ |____  $$| $$$$   /$$__  $$|_  $$_/  | $$  | $$          |
|         |  $$$$$$   /$$$$$$$| $$_/  | $$$$$$$$  | $$    | $$  | $$          |
|          \____  $$ /$$__  $$| $$    | $$_____/  | $$ /$$| $$  | $$          |
|          /$$$$$$$/|  $$$$$$$| $$    |  $$$$$$$  |  $$$$/|  $$$$$$$          |
|         |_______/  \_______/|__/     \_______/   \___/   \____  $$          |
|                                                              /$$  | $$          |
|                                                             |  $$$$$$/          |
|  by pyup.io                                                  \______/          |
|                                                                              |
+==============================================================================+
| REPORT                                                                       |
| checked 17 packages, using free DB (updated once a month)                    |
+============================+===========+========================+==========+
| package                    | installed | affected               | ID       |
+============================+===========+========================+==========+
| django                     | 1.11.29   | <2.2.24                | 40637    |
| django                     | 1.11.29   | <2.2.25                | 43041    |
| django                     | 1.11.29   | <2.2.26                | 44426    |
| django                     | 1.11.29   | <2.2.26                | 44423    |
| django                     | 1.11.29   | <2.2.26                | 44427    |
| django                     | 1.11.29   | <2.2.27                | 44741    |
| django                     | 1.11.29   | <2.2.27                | 44742    |
+==============================================================================+
```

Safety output (2)

```
+==============================================================================+
| REPORT                                                                       |
| checked 17 packages, using free DB (updated once a month)                    |
+============================+===========+=========================+==========+
| package                    | installed | affected                | ID       |
+============================+===========+=========================+==========+
| django                     | 1.11.29   | <2.2.24                 | 40637    |
+==============================================================================+
| Django before 2.2.24, 3.x before 3.1.12, and 3.2.x before 3.2.4 has a        |
| potential directory traversal via django.contrib.admindocs. Staff members    |
| could use the TemplateDetailView view to check the existence of arbitrary    |
| files. Additionally, if (and only if) the default admindocs templates have   |
| been customized by application developers to also show file contents, then   |
| not only the existence but also the file contents would have been exposed.   |
| In other words, there is directory traversal outside of the template root    |
| directories.                                                                 |
| https://www.djangoproject.com/weblog/2021/jun/02/security-releases           |
+==============================================================================+
...
```