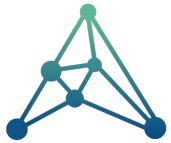
PYOPP

virtualenv & co

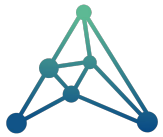
- What is virtualenv?
- Why is it needed?
- How to use it?
- How does it work?
- Variants & beyond



ERUM
DATA HUB

Quickstart for pip

- python package manager
- can manage:
  - system python packages
    ➜ **DON'T:** you'll mess things up
  - per-user python packages
    ➜ **avoid it:**
    - explicitly enabled via `--user`
  - virtual environment
    ➜ **use this:** more on next slides
- interfaces with online package repositories
  - e.g. https://pypi.org/
- install/remove package
  - `pip install package-name`
  - `pip remove package-name`
- update package
  - `pip install -U package-name`
- list available versions
  - `pip install package-name==`
- install specific version
  - `pip install package-name==1.2`
- list installed packages
  - `pip freeze`



What is virtualenv?

- **generic principle:** isolated python environments
  - user-created directory
    - no need for admin/root
  - can be located anywhere
    - i.e. directly in project folder
  - contains python packages
    - allows installing incompatible packages on same computer
  - can be (de)activated
- **specific implementation:** virtualenv *(details later)*

```
/home/username/projects/
    my-awesome-project/
        do-stuff.py
        config-files
        env/
            …
```

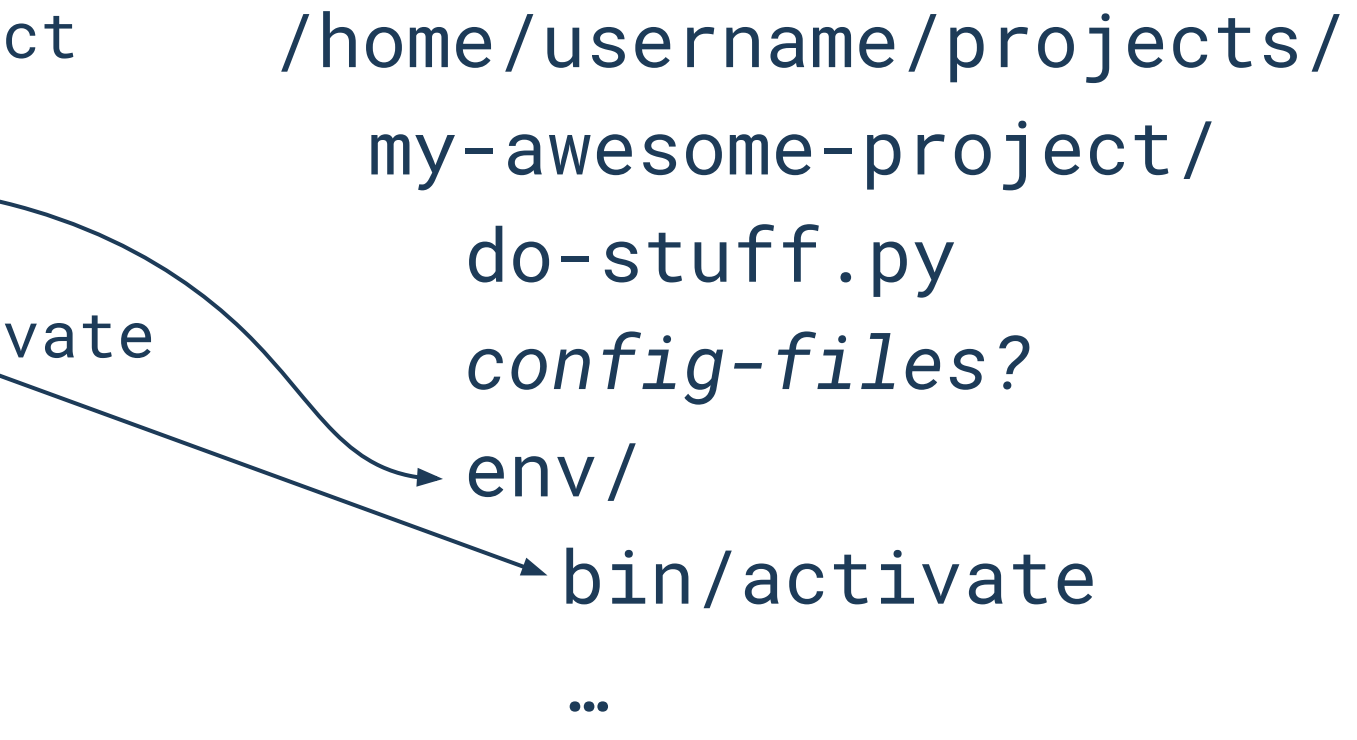

Why is it needed?

- allow installation/use of:
  - incompatible packages
  - different package versions
  - different python versions
- important: still separated
  - no simultaneous use
- for example:
  - **my-tool** needs
    - numpy < 2.0
  - **my-other-tool** needs
    - numpy ≥ 2.0

```
/home/username/projects/
  my-tool/
    env/
      …/numpy-1.26.4/…
  my-other-tool/
    env/
      …/numpy-2.3.0/…
```


How to use it?

- `cd my-awesome-project`
- create:
  - `virtualenv env`
- activate:
  - `source env/bin/activate`
- deactivate:
  - `deactivate`
- delete
  - `rm -rf env`



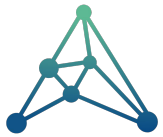
How to use it?

while environment is active:

- prompt has name as prefix
  - e.g. `(env)`
- python uses/sees packages only from this environment
- pip modifies (e.g. install, …) only this environment

- for example:
  - `pip install numpy`

```
/home/username/projects/
    my-awesome-project/
        do-stuff.py
        config-files?
        env/
            …/numpy-2.3.0/…
            …
```

Example

```
vispa-portal2:~ > cd my-awesome-project/
vispa-portal2:~/my-awesome-project > virtualenv env
created virtual environment CPython3.12.3.final.0-64 in 983ms
  creator CPython3Posix(dest=/home/BFischer/my-awesome-project/env, clear=False, no
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=/home/BFisc
    added seed packages: pip==25.1.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShel
vispa-portal2:~/my-awesome-project > source env/bin/activate
(env) vispa-portal2:~/my-awesome-project > pip install numpy
Collecting numpy
  Using cached numpy-2.3.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (62 kB)
Using cached numpy-2.3.0-cp312-cp312-manylinux_2_28_x86_64.whl (16.6 MB)
Installing collected packages: numpy
Successfully installed numpy-2.3.0
(env) vispa-portal2:~/my-awesome-project > deactivate
vispa-portal2:~/my-awesome-project > rm -rf env/
```


How does it work?

activate: to be “sourced”

- modifies environment:
  - only adds …/env/bin to PATH
  - does not change PYTHONPATH
- modifies shown prompt
  - can be turned off
- adds command: `deactivate`
  - reverts all changes of activate
  - not a file, just an ephemeral function in the current shell
- stay in same shell (i.e. history)

```
env/
   bin/
      activate
      python
      pip
   lib/python3.12/
      os.py
      site-packages/…
```

How does it work?

python executable

- copy/link of (system) python
- automatically detects virtual environment
  - via `os.py` (relative to `python`)
  - also for python tools (e.g. **pip**)
    - they explicitly use same python
  - usable without `activate`

packages

- installed into `site-packages`

```
env/
   bin/
      activate
      python
      pip
   lib/python3.12/
      os.py
      site-packages/…
```

Variants & beyond

Variants: Basic tools

- **virtualenv:** the original
  - pypi package
  - rich features
- **venv:** the python 3 built-in
  - always available (>3.3)
  - usage: `python -m venv …`
- **pipx:**
  - automatic virtual env for global tools (e.g. poetry)

Beyond: Power tools

➜ standalone executables

- **uv:** the full suite
  - Rust based ➜ very fast
  - (drop-in) replacement for pip, pipx, virtualenv, …
- **conda/mamba:**
  - “virtualenv” for any software
- **pixi:** the above, combined
  - also includes project tooling