Qu’est-ce qu’il y a dans un virtualenv?

PyConFR 2024

Denis Viviès <legnonpi@gmail.com>

Moi, moi et moi

- Denis Viviès (il/lui)
- Github: https://github.com/Gnonpi
- Repo de la présentation:
- J'aime: 
- Je travaille actuellement à Foodles

Merci à Pierre Verkest

Merci à Foodles

Merci à tech@Foodles

Précisions

- Focalisé sur Python 3.6+
- Focalisé sur CPython
- Focalisé sur Unix
- Focalisé sur venv+pip
- Je ne suis pas un génie,
  il peut y avoir des erreurs

Sommaire

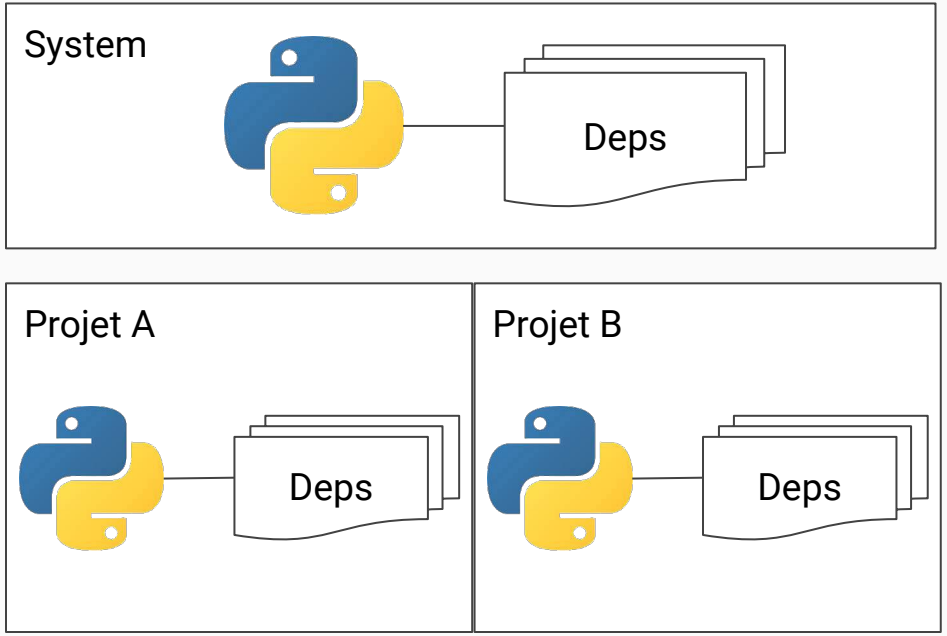
1. Comment on utilise un venv?
2. Histoire de $PATH
3. Structure de dedans
4. Le processus de création
5. C’est quoi “installer un package”?
6. Les outils pour gérer

1. Comment on utilise un venv?

Comment on utilise un venv

Au fait, pourquoi on utilise un venv?

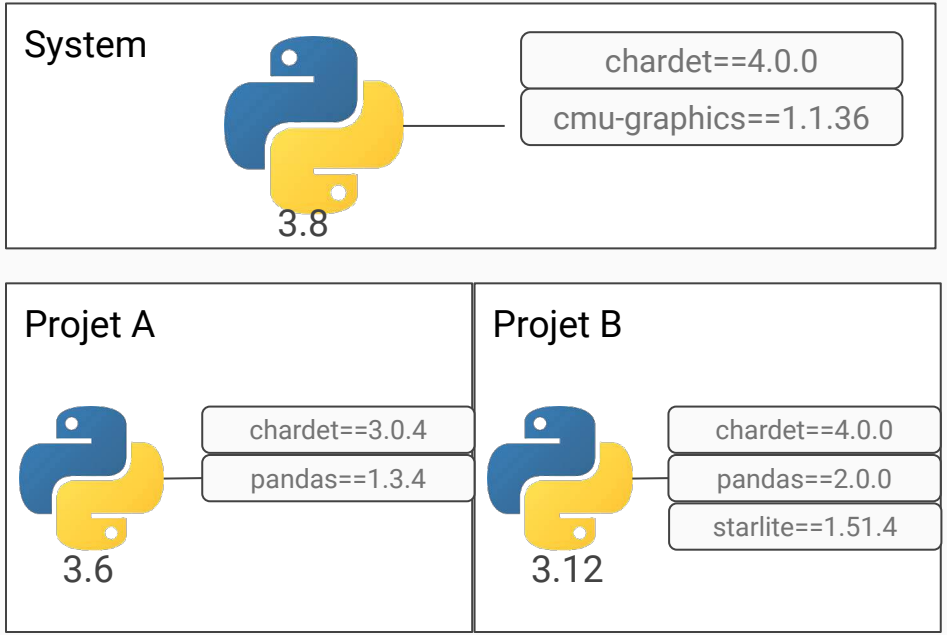
- Un projet = ses dépendances
- Isolation des dépendances
- Standardisé



Comment on utilise un venv

Au fait, pourquoi on utilise un venv?

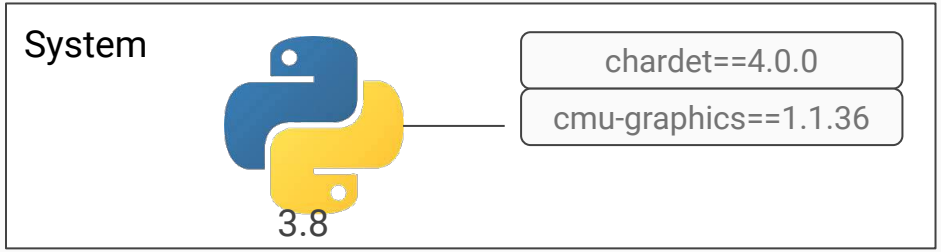
- Un projet = ses dépendances
- Isolation des dépendances
- Standardisé

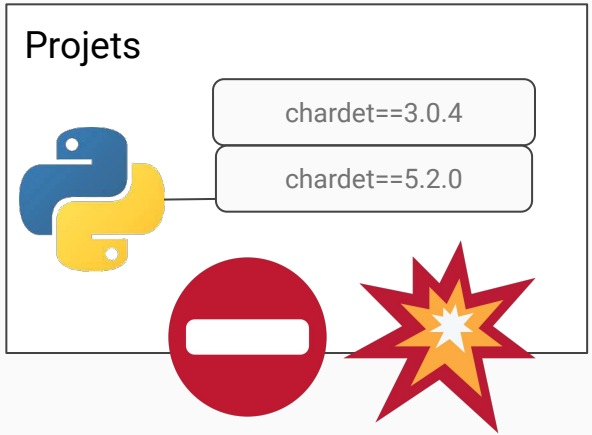


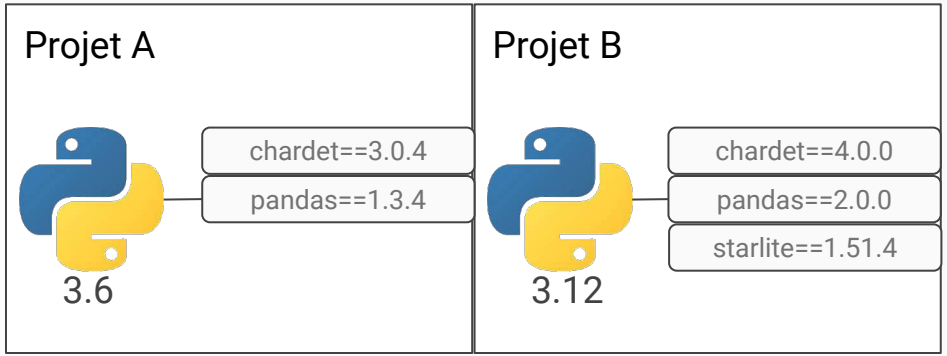
Comment on utilise un venv

Au fait, pourquoi on utilise un venv?

- Un projet = ses dépendances
- Isolation des dépendances
- Standardisé







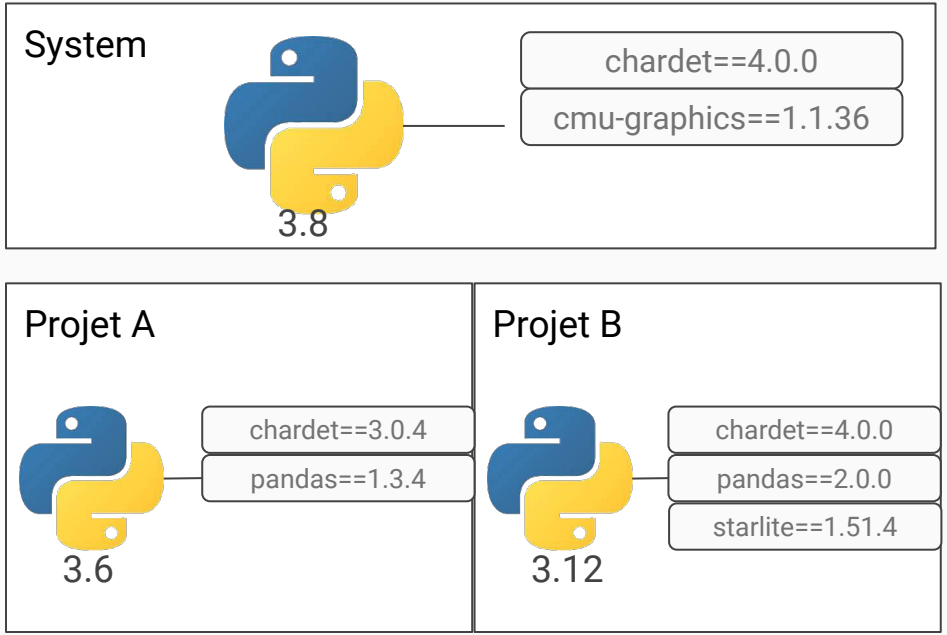
Comment on utilise un venv

Au fait, pourquoi on utilise un venv?

- Un projet = ses dépendances
- Isolation des dépendances
- Standardisé

PEP 405 – Python Virtual Environments

Author: Carl Meyer <carl at oddbird.net>
BDFL-Delegate: Alyssa Coghlan
Status: Final
Type: Standards Track
Topic: Packaging
Created: 13-Jun-2011
Python-Version: 3.3

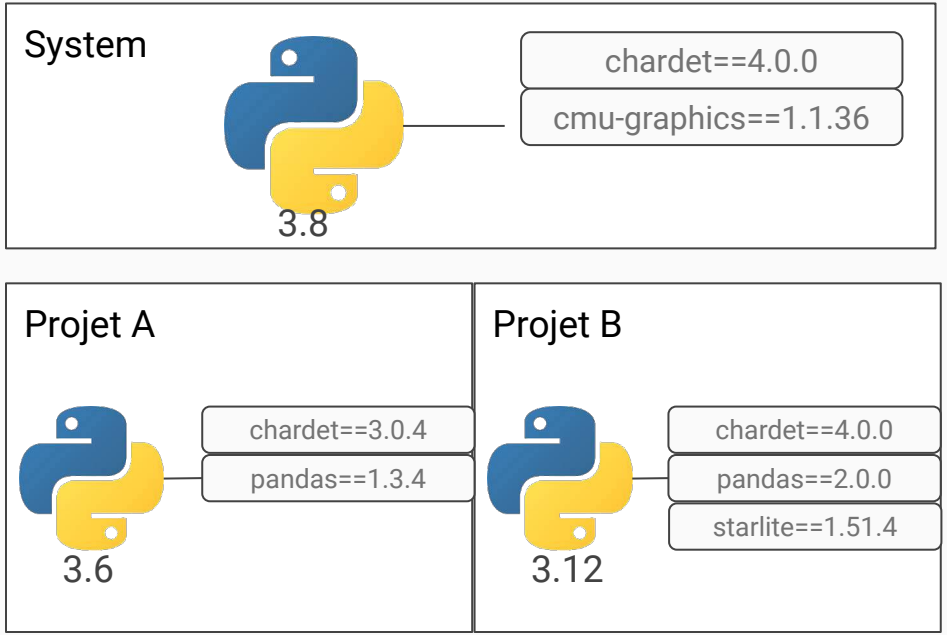


Comment on utilise un venv

Au fait, pourquoi on utilise un venv?

- Un projet = ses dépendances
- Isolation des dépendances
- Standardisé

Beaucoup d'outils modernes utilisent un virtualenv sous le capot



Comment on utilise un venv

Les commandes types:

```
➜  scripts cd project
requirements.txt
➜  project python3 -m venv venv
➜  project source venv/bin/activate
(venv) ➜  project pip install -r requirements.txt
Collecting pandas (from -r requirements.txt (line 1))
  Downloading pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_64.manylinux
 (89 kB)
                                                    89.9/89.9 kB 3.9 MB/s et
Collecting numpy>=1.26.0 (from pandas->-r requirements.txt (line 1))
Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, tzdata, six, numpy, python-dateuti
Successfully installed numpy-2.1.2 pandas-2.2.3 python-dateutil-2.9.0.p
16.0 tzdata-2024.2
(venv) ➜  project python3 -c "import pandas"
(venv) ➜  project python3 -c "import pandas; print(pandas.__version__)"
2.2.3
(venv) ➜  project
```

Comment on utilise un venv

Ce qui commence à être intéressant:

```
➜  scripts cd project
requirements.txt
➜  project which python3
/usr/bin/python3
➜  project which pip3
/usr/bin/pip3
➜  project python3 -m venv venv
sourc#
➜  project source venv/bin/activate
(venv) ➜  project which python3
/tmp/scripts/project/venv/bin/python3
(venv) ➜  project which pip3
/tmp/scripts/project/venv/bin/pip3
(venv) ➜  project pip install -r requirements.txt
Collecting pandas (from -r requirements.txt (line 1))
  Downloading pandas-2.2.3-cp312-cp312-manylinux_2_17_x86_
 (89 kB)
                                                89.9/89.9 kB
Collecting numpy>=1.26.0 (from pandas->-r requirements.txt
```

Comment on utilise un venv

Vérifier que l'on a bien installé les dépendances:

```
➜  project pip freeze
setuptools==68.1.2
wheel==0.42.0
➜  project source venv/bin/activate
(venv) ➜  project pip freeze
numpy==2.1.2
pandas==2.2.3
python-dateutil==2.9.0.post0
pytz==2024.2
six==1.16.0
tzdata==2024.2
(venv) ➜  project deactivate
➜  project venv/bin/pip freeze
numpy==2.1.2
pandas==2.2.3
python-dateutil==2.9.0.post0
pytz==2024.2
six==1.16.0
tzdata==2024.2
```

Comment on utilise un venv

Est-ce que ça marche si je copie-colle mon dossier ailleurs?

```
➜  project mv venv renamedvenv
➜  project renamedvenv/bin/pip freeze
zsh: renamedvenv/bin/pip: bad interpreter: /tmp/scripts/project/venv/bin/python3: no such file
 or directory
➜  project
```

2. Histoire de $PATH

Histoire de $PATH

Quand on installe certains programmes:
“add this to your $PATH”

Ou des scripts bash avec:
#!/bin/bash

If `pipenv` isn’t available in your shell after installation, you’ll need to add the user site-packages binary directory to your `PATH`.

On Linux and macOS you can find the user base binary directory by running `python -m site --user-base` and appending `bin` to the end. For example, this will typically print `~/.local` (with `~` expanded to the absolute path to your home directory), so you’ll need to add `~/.local/bin` to your `PATH`. You can set your `PATH` permanently by modifying ~/.profile.

```
#!/bin/bash
set -e
python3 -m venv --upgrade-deps .venv
. .venv/bin/activate
pip install -r requirements/dev.txt
pip install -e .
pre-commit install --install-hooks
```

Histoire de $PATH

Quand on installe certains programmes:
“add this to your $PATH”

Ou de vieux scripts python avec un “shebang”:
`#!/usr/bin/env python3`

To have access to Yarn’s executables globally, you will need to set up the `PATH` environment variable in your terminal. To do this, add `export PATH="$PATH:`yarn global bin`"` to your profile, or if you use Fish shell, simply run the command `set -U fish_user_paths (yarn global bin) $fish_user_paths`

main pandas / generate_version.py

twoertwein TYP: simple return types from ruff (#56568)

Code Blame 69 lines (54 loc) · 1.67 KB

```
#!/usr/bin/env python3

# Note: This file has to live next to setup.py or versioneer will not work
import argparse
import os
import sys

import versioneer

sys.path.insert(0, "")
```

⇒ Recherche et résolution de l’exécutable à lancer

Histoire de $PATH

```
➜  scripts echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Lecture de gauche à droite, séparés par :

Si un exécutable est dans le dossier,
il est sélectionné

Sinon, on passe au dossier à droite

Histoire de $PATH

Résolution suivant le $PATH:

```
➜  fooo mkdir -p aaa bbb ccc
➜  fooo echo -e '#!/bin/bash\necho "aaa"' > aaa/foo
➜  fooo echo -e '#!/bin/bash\necho "bbb"' > bbb/foo
➜  fooo echo -e '#!/bin/bash\necho "ccc"' > ccc/foo
➜  fooo chmod u+x aaa/foo bbb/foo ccc/foo
➜  fooo PATH=./ccc:./aaa:$PATH foo
ccc
➜  fooo PATH=./aaa:./bbb:$PATH foo
aaa
```

Histoire de $PATH

Résolution suivant le $PATH:

Si on regarde dans `venv/bin/activate`:

cpython / Lib / venv / scripts / common / **activate**

Code | Blame | 76 lines (65 loc) · 2.12 KB ·

```
39  # on windows, a path can contain colons and backslashes and
40  case "$(uname)" in
41      CYGWIN*|MSYS*|MINGW*)
42          # transform D:\path\to\venv to /d/path/to/venv on MS
43          # and to /cygdrive/d/path/to/venv on Cygwin
44          VIRTUAL_ENV=$(cygpath __VENV_DIR__)
45          export VIRTUAL_ENV
46          ;;
47      *)
48          # use the path as-is
49          export VIRTUAL_ENV=__VENV_DIR__
50          ;;
51  esac
52
53  _OLD_VIRTUAL_PATH="$PATH"
54  PATH="$VIRTUAL_ENV/"__VENV_BIN_NAME__":$PATH"
55  export PATH
56
```

Histoire de $PATH

Mais en fait, ça va plus loin:

Le shebang des bin/ installés

Permet de faire si on n'est pas sûr:
`venv/bin/pip install -r requirements.txt`

```
➜  scripts cat venv/bin/pip
#!/tmp/scripts/venv/bin/python3
# -*- coding: utf-8 -*-
import re
import sys
from pip._internal.cli.main import main
if __name__ == '__main__':
```

C’est quoi le $**PYTHONPATH** ?

C’est un $PATH pour résoudre les modules Python

C’est `sys.path`

Voilà

```
➜  scripts python3 -m venv venv-a
➜  scripts python3 -m venv venv-b
➜  scripts venv-a/bin/pip install -q pandas
➜  scripts venv-b/bin/pip install -q pydantic
p#
➜  scripts python3
Python 3.12.3 (main, Sep 11 2024, 14:17:37) [GCC 13.2.0] on
Type "help", "copyright", "credits" or "license" for more in
>>> import sys
>>> import pandas
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'pandas'
>>> sys.path.append("venv-a/lib/python3.12/site-packages")
>>> sys.path.append("venv-b/lib/python3.12/site-packages")
>>> import pandas
>>> import pydantic
>>>
```

3. La structure de dedans

La structure de dedans

Faisons un petit ls:

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

La structure de dedans

Faisons un petit ls:

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

Ça on connaît

Ça, ça ressemble à du C

Ça vous l'utilisez tout le temps

Ça on veut que ça disparaisse

Un fichier de config?

La structure de dedans - bin

On connaît déjà 3 trucs dans `venv/bin/`:

- activate
- pip
- python

La structure de dedans - bin

On connaît déjà 3 trucs dans `venv/bin/`:

```
➜  scripts ls -l venv/bin
total 36
-rw-r--r-- 1 root root 2022 Oct 24 19:28 activate
-rw-r--r-- 1 root root  911 Oct 24 19:28 activate.csh
-rw-r--r-- 1 root root 2190 Oct 24 19:28 activate.fish
-rw-r--r-- 1 root root 9033 Oct 24 19:28 Activate.ps1
-rwxr-xr-x 1 root root  234 Oct 24 19:28 pip
-rwxr-xr-x 1 root root  234 Oct 24 19:28 pip3
-rwxr-xr-x 1 root root  234 Oct 24 19:28 pip3.12
lrwxrwxrwx 1 root root    7 Oct 24 19:28 python -> python3
lrwxrwxrwx 1 root root   16 Oct 24 19:28 python3 -> /usr/bin/python3
lrwxrwxrwx 1 root root    7 Oct 24 19:28 python3.12 -> python3
```

La structure de dedans - bin

Comment un package donne un exécutable:

```
➜  scripts ls venv/bin
activate      activate.fish  f2py          pip   pip3.12  python3
activate.csh  Activate.ps1   numpy-config  pip3  python   python3.12
➜  scripts venv/bin/numpy-config
usage: numpy-config [-h] [--version] [--cflags] [--pkgconfigdir]

options:
  -h, --help      show this help message and exit
  --version       Print the version and exit.
  --cflags        Compile flag needed when using the NumPy headers.
  --pkgconfigdir  Print the pkgconfig directory in which `numpy.pc` i
                  setting $PKG_CONFIG_PATH).
➜  scripts cat venv/bin/numpy-config
#!/tmp/scripts/venv/bin/python3
# -*- coding: utf-8 -*-
import re
import sys
from numpy._configtool import main
if __name__ == '__main__':
    sys.argv[0] = re.sub(r'(-script\.pyw|\.exe)?$', '', sys.argv[0])
    sys.exit(main())
```

La structure de dedans

Faisons un petit ls:

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

La structure de dedans

Faisons un petit ls:

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

Ça c'est quoi?

La structure de dedans - pyvenv.cfg

Petit fichier de config:

- trace de la commande initiale
- key-value des options
- commande vers l'exécutable python

Métadonnées du venv

```
❯ cat venv/pyvenv.cfg
home = /home/denis-foodles/.pyenv/versions/3.11.4/bin
include-system-site-packages = false
version = 3.11.4
executable = /home/denis-foodles/.pyenv/versions/3.11.4/bin/python3.11
command = /home/denis-foodles/.pyenv/versions/3.11.4/bin/python -m venv /tmp/project/venv
```

La structure de dedans - pyvenv.cfg

Petit fichier de config:

- trace de la commande initiale
- key-value des options
- commande vers l’exécutable python

Métadonnées du venv

Ajout automatique à `sys.prefix` et `sys.exec_prefix`

La structure de dedans - pyvenv.cfg

Petit fichier de config:

- trace de la commande initiale
- key-value des options
- commande vers l'exécutable python

Métadonnées du venv

Ajout automatique à `sys.prefix` et `sys.exec_prefix`
pendant l'import automatique de `site`

```
❯ venv/bin/python3 -m site
sys.path = [
    '/tmp',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python311.zip',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python3.11',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python3.11/lib-dynload',
    '/tmp/venv/lib/python3.11/site-packages',
]
USER_BASE: '/home/denis-foodles/.local' (exists)
USER_SITE: '/home/denis-foodles/.local/lib/python3.11/site-packages' (doesn't exist)
ENABLE_USER_SITE: False
❯ python3 -m site
sys.path = [
    '/tmp',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python311.zip',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python3.11',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python3.11/lib-dynload',
    '/home/denis-foodles/.pyenv/versions/3.11.4/lib/python3.11/site-packages',
]
USER_BASE: '/home/denis-foodles/.local' (exists)
USER_SITE: '/home/denis-foodles/.local/lib/python3.11/site-packages' (doesn't exist)
ENABLE_USER_SITE: True
```

La structure de dedans

Faisons un petit ls:

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

La structure de dedans

Faisons un petit ls:

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

Ça vous l'utilisez tout le temps

La structure de dedans - lib et lib64

`lib` c'est pour 'library' → ensemble des packages installés

- **lib/** ⇒ dossier où l'on installe les packages
- **lib64/** ⇒ symlink vers lib/ pour rétrocompatibilité
- **include/** ⇒ headers C pour les packages qui utilisent l'API Python/C

```
❯ ls venv
bin  include  lib  lib64  pyvenv.cfg
```

La structure de dedans - lib et lib64

`lib/` ⇒ dossier où l'on installe les packages

```
❯ tree -L 3 venv/lib
venv/lib
└── python3.11
    └── site-packages
        ├── dateutil
        ├── _distutils_hack
        ├── distutils-precedence.pth
        ├── numpy
        ├── numpy-2.1.1.dist-info
        ├── numpy.libs
        ├── pandas
        ├── pandas-2.2.2.dist-info
        ├── pip
        ├── pip-23.1.2.dist-info
        ├── pkg_resources
        ├── __pycache__
        ├── python_dateutil-2.9.0.post0.dist-info
        ├── pytz
        ├── pytz-2024.2.dist-info
        ├── setuptools
        ├── setuptools-65.5.0.dist-info
        ├── six-1.16.0.dist-info
        ├── six.py
        ├── tzdata
        └── tzdata-2024.1.dist-info

22 directories, 2 files
```

La structure de dedans - lib et lib64

lib/ ⇒ dossier où l'on installe les packages

Structure standardisée

`import` va venir chercher les librairies ici

```
❯ tree -L 3 venv/lib
venv/lib
└── python3.11
    └── site-packages
        ├── dateutil
        ├── _distutils_hack
        ├── distutils-precedence.pth
        ├── numpy
        ├── numpy-2.1.1.dist-info
        ├── numpy.libs
        ├── pandas
        ├── pandas-2.2.2.dist-info
        ├── pip
        ├── pip-23.1.2.dist-info
        ├── pkg_resources
        ├── __pycache__
        ├── python_dateutil-2.9.0.post0.dist-info
        ├── pytz
        ├── pytz-2024.2.dist-info
        ├── setuptools
        ├── setuptools-65.5.0.dist-info
        ├── six-1.16.0.dist-info
        ├── six.py
        ├── tzdata
        └── tzdata-2024.1.dist-info

22 directories, 2 files
```

Aparté sur `import`

https://docs.python.org/3/reference/import.html#importsystem

`import` =

1. Trouver un module, le charger
2. Charger les noms dans le namespace

Aparté sur `import`

https://docs.python.org/3/reference/import.html#importsystem

`import` =

1. Trouver un module, le charger
2. Charger les noms dans le namespace

“All packages are modules"

1. sys.modules
2. finder: built-in
3. finder: frozen modules
4. finder: import path -> sys.path

Aparté sur `import`

https://docs.python.org/3/reference/import.html#importsystem

`import` =

1. Trouver un module, le charger
2. Charger les noms dans le namespace

“All packages are modules"

- sys.modules
- finder: built-in
- finder: frozen modules
- finder: import path -> sys.path

```
➜  scripts venv/bin/python3
Python 3.12.3 (main, Sep 11 2024, 14:17:37) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more informati
on.
>>> import sys
>>> sys.path
['', '/usr/lib/python312.zip', '/usr/lib/python3.12', '/usr/lib/pyt
hon3.12/lib-dynload', '/tmp/scripts/venv/lib/python3.12/site-packag
es']
>>>
```

La structure de dedans - lib et lib64

lib/ ⇒ dossier où l'on installe les packages

- dist-info
- <package>/
- random <package>.py

```
❯ tree -L 3 venv/lib
venv/lib
└── python3.11
    └── site-packages
        ├── dateutil
        ├── _distutils_hack
        ├── distutils-precedence.pth
        ├── numpy
        ├── numpy-2.1.1.dist-info
        ├── numpy.libs
        ├── pandas
        ├── pandas-2.2.2.dist-info
        ├── pip
        ├── pip-23.1.2.dist-info
        ├── pkg_resources
        ├── __pycache__
        ├── python_dateutil-2.9.0.post0.dist-info
        ├── pytz
        ├── pytz-2024.2.dist-info
        ├── setuptools
        ├── setuptools-65.5.0.dist-info
        ├── six-1.16.0.dist-info
        ├── six.py
        ├── tzdata
        └── tzdata-2024.1.dist-info

22 directories, 2 files
```

La structure de dedans - lib et lib64

lib/ ⇒ dossier où l'on installe les packages

- dist-info
- <package>/
- random <package>.py

1 module = 1 version = 1 artefact

```
❯ tree -L 3 venv/lib
venv/lib
└── python3.11
    └── site-packages
        ├── dateutil
        ├── _distutils_hack
        ├── distutils-precedence.pth
        ├── numpy
        ├── numpy-2.1.1.dist-info
        ├── numpy.libs
        ├── pandas
        ├── pandas-2.2.2.dist-info
        ├── pip
        ├── pip-23.1.2.dist-info
        ├── pkg_resources
        ├── __pycache__
        ├── python_dateutil-2.9.0.post0.dist-info
        ├── pytz
        ├── pytz-2024.2.dist-info
        ├── setuptools
        ├── setuptools-65.5.0.dist-info
        ├── six-1.16.0.dist-info
        ├── six.py
        ├── tzdata
        └── tzdata-2024.1.dist-info

22 directories, 2 files
```

La structure de dedans - lib et lib64

lib/ ⇒ dossier où l'on installe les packages

- dist-info
- <package>/
- wheel?
- random <package>.py

1 module = 1 version = 1 artefact

```
packageA
    requires packageC==1.0.0
packageB
    requires packageC==2.0.0
```

1 package = 1 version

npm aliasing:
yarn add is-even@1.0.0 myotheriseven@npm:is-even@0.1.0

```
❯ yarn add is-even@1.0.0 myotheriseven@npm:is-even@0.1.0
yarn add v1.22.22
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Saved lockfile.
success Saved 6 new dependencies.
info Direct dependencies
├─ is-even@1.0.0
└─ myotheriseven@0.1.0
info All dependencies
├─ is-buffer@1.1.6
├─ is-even@1.0.0
├─ is-number@1.1.2
├─ is-odd@0.1.2
├─ kind-of@3.2.2
└─ myotheriseven@0.1.0
Done in 1.13s.
❯ ls node_modules
is-buffer  is-even  is-number  is-odd  kind-of  myotheriseven
```

1 module = 1 version = 1 artefact

1 package = 1 version

cargo multiple versions:

If multiple packages have a common dependency with semver-incompatible versions, then Cargo will allow this, but will build two separate copies of the dependency. For example:

```
# Package A
[dependencies]
rand = "0.7"

# Package B
[dependencies]
rand = "0.6"
```

The above will result in Package A using the greatest `0.7` release (`0.7.3` at the time of this writing) and Package B will use the greatest `0.6` release (`0.6.5` for example). This can lead to potential problems, see the Version-incompatibility hazards section for more details.

1 module = 1 version = 1 artefact

La structure de dedans - lib et lib64

lib/ ⇒ dossier où l'on installe les packages

- dist-info
- <package>/
- random <package>.py

1 module = 1 version = 1 artefact

⇒ on y revient partie 5

```
❯ tree -L 3 venv/lib
venv/lib
└── python3.11
    └── site-packages
        ├── dateutil
        ├── _distutils_hack
        ├── distutils-precedence.pth
        ├── numpy
        ├── numpy-2.1.1.dist-info
        ├── numpy.libs
        ├── pandas
        ├── pandas-2.2.2.dist-info
        ├── pip
        ├── pip-23.1.2.dist-info
        ├── pkg_resources
        ├── __pycache__
        ├── python_dateutil-2.9.0.post0.dist-info
        ├── pytz
        ├── pytz-2024.2.dist-info
        ├── setuptools
        ├── setuptools-65.5.0.dist-info
        ├── six-1.16.0.dist-info
        ├── six.py
        ├── tzdata
        └── tzdata-2024.1.dist-info

22 directories, 2 files
```

4. Le processus de création

EnvBuilder

https://github.com/python/cpython/tree/main/Lib/venv

Suivre `EnvBuilder.create()`

- trouver les site-packages et python existants
- créer la configuration (donc `pyvenv.cfg`)
- préparer python (copier **python** dans `bin/`, ajouter symlinks)
- préparer pip (appeler `ensurepip` avec des args)
- préparer les scripts (activate et autres)

```
class EnvBuilder:
    """
    This class exists to allow virtual environment creat
    customized. The constructor parameters determine the
    behaviour when called upon to create a virtual envir

    By default, the builder makes the system (global) si
    *un*available to the created environment.

    If invoked using the Python -m option, the default i
    on Windows platforms but symlinks elsewhere. If inst
    other way, the default is to *not* use symlinks.

    :param system_site_packages: If True, the system (gl
                                 dir is available to cre
    :param clear: If True, delete the contents of the en
                  it already exists, before environment
    :param symlinks: If True, attempt to symlink rather
                     virtual environment.
    :param upgrade: If True, upgrade an existing virtual
```

À la main?

Ça devrait être facile à refaire en bash non?

Lancer un vote à main levée

-A-	-B-	-C-
Oui, en 20mn	Oui, en 2-3 soirées	Trop compliqué

À la main?

Ça devrait être facile de le refaire en bash non?

Réponse A: environ 20mn, une 30n de lignes

Limites (non-exhaustives):

- pas de activate
- python déjà bootstrappé
- on suppose un système linux
- des strings hardcodés
- …

$ create-venv.sh

```bash
#!/bin/bash

echo ">Creating virtualenv"

VENVNAME=myvenv
CURRENT_PYTHON=$(pyenv which python)
CURRENT_PYTHON_BIN=$(dirname ${CURRENT_PYTHON})

echo ">> Cleanup"
rm -rf ${VENVNAME}

mkdir ${VENVNAME}

echo ">> Adding pyvenv.cfg"
# envsubst to replace values
# todo: clean it up
PYVENV_PATTERN="
home = ${CURRENT_PYTHON_BIN}
include-system-site-packages = false
version = 3.12.3
executable = ${CURRENT_PYTHON}
command = ${CURRENT_PYTHON} -m venv /tmp/proj/myvenv
"
echo ${PYVENV_PATTERN} > /tmp/pattern
envsubst < /tmp/pattern > ${VENVNAME}/pyvenv.cfg

echo ">> Adding top level dirs"
mkdir -p ${VENVNAME}/bin ${VENVNAME}/include ${VENVNAME}/lib
ln -s lib ${VENVNAME}/lib64

echo ">> Adding python links"
ln -s python ${VENVNAME}/bin/python3.12
ln -s python ${VENVNAME}/bin/python3
ln -s ${CURRENT_PYTHON} ${VENVNAME}/bin/python

echo ">> Adding pip"
${VENVNAME}/bin/python -m ensurepip --default-pip

echo "> Installing a package, using it"
${VENVNAME}/bin/pip install pandas
${VENVNAME}/bin/python -c 'import pandas as pd; print(pd.DataFrame())'

echo -e "\n\n"

# ----------------------------
tree -L 5 myvenv
```

À la main?

Attraper le Python existant

Créer le dossier

Remplir le pyvenv.cfg

Ajouter les dossiers internes

Créer les symlinks Python

Appeler `ensurepip`

Profit

$ create-venv.sh

```bash
#!/bin/bash

echo ">Creating virtualenv"

VENVNAME=myvenv
CURRENT_PYTHON=$(pyenv which python)
CURRENT_PYTHON_BIN=$(dirname ${CURRENT_PYTHON})

echo ">> Cleanup"
rm -rf ${VENVNAME}

mkdir ${VENVNAME}

echo ">> Adding pyvenv.cfg"
# envsubst to replace values
# todo: clean it up
PYVENV_PATTERN="
home = ${CURRENT_PYTHON_BIN}
include-system-site-packages = false
version = 3.12.3
executable = ${CURRENT_PYTHON}
command = ${CURRENT_PYTHON} -m venv /tmp/proj/myvenv
"
echo ${PYVENV_PATTERN} > /tmp/pattern
envsubst < /tmp/pattern > ${VENVNAME}/pyvenv.cfg

echo ">> Adding top level dirs"
mkdir -p ${VENVNAME}/bin ${VENVNAME}/include ${VENVNAME}/lib
ln -s lib ${VENVNAME}/lib64

echo ">> Adding python links"
ln -s python ${VENVNAME}/bin/python3.12
ln -s python ${VENVNAME}/bin/python3
ln -s ${CURRENT_PYTHON} ${VENVNAME}/bin/python

echo ">> Adding pip"
${VENVNAME}/bin/python -m ensurepip --default-pip

echo "> Installing a package, using it"
${VENVNAME}/bin/pip install pandas
${VENVNAME}/bin/python -c 'import pandas as pd; print(pd.DataFrame())'

echo -e "\n\n"

# ----------------------------
tree -L 5 myvenv
```

À la main?

Attraper le Python existant

er

.cfg

nternes

Python

pip`

Profit

```
→ scripts ./create-venv.sh
>Creating virtualenv
>> Cleanup
>> Adding pyvenv.cfg
>> Adding top level dirs
>> Adding python links
>> Adding pip
> Installing a package, using it
Defaulting to user installation because normal site-packages is not writeable
Collecting chardet
  Downloading chardet-5.2.0-py3-none-any.whl.metadata (3.4 kB)
Downloading chardet-5.2.0-py3-none-any.whl (199 kB)
Installing collected packages: chardet
Successfully installed chardet-5.2.0
<module 'chardet' from '/home/pyconfruser/.local/lib/python3.13/site-packages/chardet/__i
nit__.py'>


myvenv
├── bin
│   ├── pip -> pip3
│   ├── pip3
│   ├── python -> /usr/local/bin/python3
│   ├── python3 -> python
│   └── python3.12 -> python
├── include
├── lib
├── lib64 -> lib
└── pyvenv.cfg

5 directories, 6 files
→ scripts
```

$ create-venv.sh

```
#!/bin/bash

echo ">Creating virtualenv"

VENVNAME=myvenv
CURRENT_PYTHON=$(pyenv which python)
CURRENT_PYTHON_BIN=$(dirname ${CURRENT_PYTHON})

echo ">> Cleanup"
rm -rf ${VENVNAME}

mkdir ${VENVNAME}

echo ">> Adding pyvenv.cfg"
# envsubst to replace values
# todo: clean it up
PYVENV_PATTERN="
home = ${CURRENT_PYTHON_BIN}
include-system-site-packages = false
version = 3.12.3
executable = ${CURRENT_PYTHON}
command = ${CURRENT_PYTHON} -m venv /tmp/proj/myvenv
"
echo ${PYVENV_PATTERN} > /tmp/pattern
envsubst < /tmp/pattern > ${VENVNAME}/pyvenv.cfg

echo ">> Adding top level dirs"
mkdir -p ${VENVNAME}/bin ${VENVNAME}/include ${VENVNAME}/lib
ln -s lib ${VENVNAME}/lib64

echo ">> Adding python links"
ln -s python ${VENVNAME}/bin/python3.12
ln -s python ${VENVNAME}/bin/python3
ln -s ${CURRENT_PYTHON} ${VENVNAME}/bin/python

echo ">> Adding pip"
${VENVNAME}/bin/python -m ensurepip --default-pip

echo "> Installing a package, using it"
${VENVNAME}/bin/pip install pandas
${VENVNAME}/bin/python -c 'import pandas as pd; print(pd.DataFrame())'

echo -e "\n\n"

# ----------------------------
tree -L 5 myvenv
```

À la main?

Attraper le Python existant

Créer le dossier

Remplir le pyvenv.cfg

Ajouter les dossiers internes

Créer les symlinks Python

Appeler `ensurepip`

Profit

$ create-venv.sh

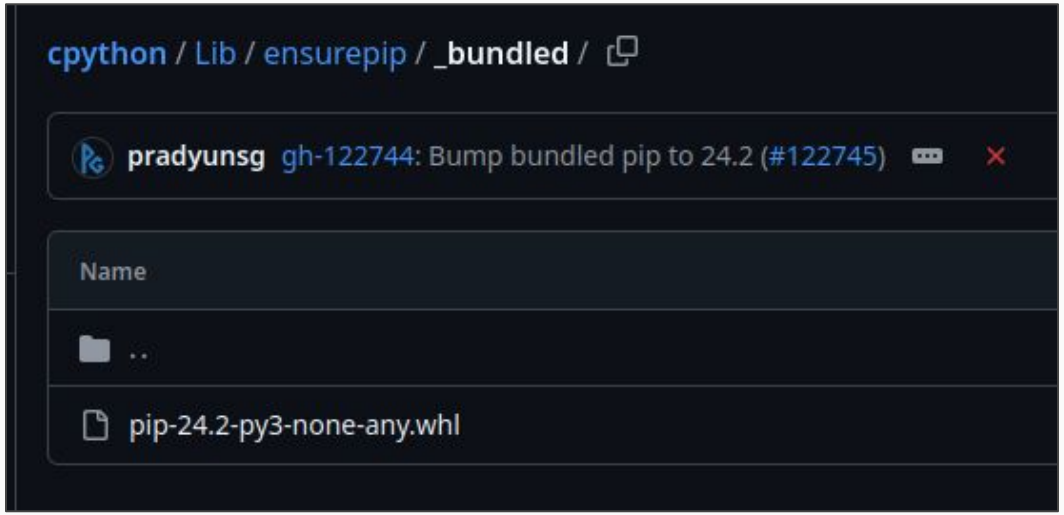
```bash
#!/bin/bash

echo ">Creating virtualenv"

VENVNAME=myvenv
CURRENT_PYTHON=$(pyenv which python)
CURRENT_PYTHON_BIN=$(dirname ${CURRENT_PYTHON})

echo ">> Cleanup"
rm -rf ${VENVNAME}

mkdir ${VENVNAME}

echo ">> Adding pyvenv.cfg"
# envsubst to replace values
# todo: clean it up
PYVENV_PATTERN="
home = ${CURRENT_PYTHON_BIN}
include-system-site-packages = false
version = 3.12.3
executable = ${CURRENT_PYTHON}
command = ${CURRENT_PYTHON} -m venv /tmp/proj/myvenv
"
echo ${PYVENV_PATTERN} > /tmp/pattern
envsubst < /tmp/pattern > ${VENVNAME}/pyvenv.cfg

echo ">> Adding top level dirs"
mkdir -p ${VENVNAME}/bin ${VENVNAME}/include ${VENVNAME}/lib
ln -s lib ${VENVNAME}/lib64

echo ">> Adding python links"
ln -s python ${VENVNAME}/bin/python3.12
ln -s python ${VENVNAME}/bin/python3
ln -s ${CURRENT_PYTHON} ${VENVNAME}/bin/python

echo ">> Adding pip"
${VENVNAME}/bin/python -m ensurepip --default-pip

echo "> Installing a package, using it"
${VENVNAME}/bin/pip install pandas
${VENVNAME}/bin/python -c 'import pandas as pd; print(pd.DataFrame())'

echo -e "\n\n"

# ----------------------------
tree -L 5 myvenv
```

C’est quoi `ensurepip`?

https://docs.python.org/3/library/ensurepip.html#module-ensurepip

Module même sans accès internet - pip bundled avec CPython

Juste installer pip dans le virtualenv ou system libs



5. C’est quoi “installer un package”?

Installer dans un venv

Qu’est-ce qui se passe quand je fais ?

```
$ venv/bin/pip install pandas
```

Installer dans un venv

Qu’est-ce qui se passe quand je fais ?

```
$ venv/bin/pip install pandas
```

Y’a des histoires de wheel, des eggs,
des fois il faut faire `apt get <lib>`?

Installer dans un venv

Qu’est-ce qui se passe quand je fais ?

```
$ venv/bin/pip install pandas
```

Y’a des histoires de wheel, des eggs,
des fois il faut faire `apt get <lib>`?

Installer dans un venv

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

```
/tmp/proj  ls venv/lib/python3.10/site-packages/                                  ✔  8s  22:53:26
dateutil                    pip-24.2.virtualenv                    six.py
_distutils_hack             pkg_resources                          tzdata
distutils-precedence.pth    __pycache__                            tzdata-2024.2.dist-info
numpy                       python_dateutil-2.9.0.post0.dist-info  _virtualenv.pth
numpy-2.1.2.dist-info       pytz                                   _virtualenv.py
numpy.libs                  pytz-2024.2.dist-info                  wheel
pandas                      setuptools                             wheel-0.44.0.dist-info
pandas-2.2.3.dist-info      setuptools-75.1.0.dist-info            wheel-0.44.0.virtualenv
pip                         setuptools-75.1.0.virtualenv
pip-24.2.dist-info          six-1.16.0.dist-info
```

Installer dans un venv

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

Est-ce que je suis capable de l'implémenter en bash ?

Installer dans un venv - simple py

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

Est-ce que je suis capable de l'implémenter en bash ?

```
root@69b8601d4821:/tmp/scripts# cd ..
root@69b8601d4821:/tmp# cat <<EOF > mymodule.py
def say_hello():
    print("hello")
EOF
root@69b8601d4821:/tmp# python3 -m venv venv
root@69b8601d4821:/tmp# mv mymodule.py venv/lib/python3.12/site-packages/
root@69b8601d4821:/tmp# venv/bin/python -c "from mymodule import say_hello; say_hello()"
hello
```

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

```
➜  scripts python3 -m venv --no-pip venv
usage: venv [-h] [--system-site-packages] [--symlinks | --copies] [--clear] [--upgrade]
            [--without-pip] [--prompt PROMPT] [--upgrade-deps]
            [--without-scm-ignore-files]
            ENV_DIR [ENV_DIR ...]
venv: error: unrecognized arguments: --no-pip
➜  scripts python3 -m venv --without-pip venvnopip
➜  scripts ls -lh venvnopip/bin
total 24K
-rw-r--r-- 1 pyconfruser pyconfruser 2.1K Oct 24 21:01 activate
-rw-r--r-- 1 pyconfruser pyconfruser  919 Oct 24 21:01 activate.csh
-rw-r--r-- 1 pyconfruser pyconfruser 2.2K Oct 24 21:01 activate.fish
-rw-r--r-- 1 pyconfruser pyconfruser 8.9K Oct 19 04:16 Activate.ps1
lrwxrwxrwx 1 pyconfruser pyconfruser    7 Oct 24 21:01 python -> python3
lrwxrwxrwx 1 pyconfruser pyconfruser   22 Oct 24 21:01 python3 -> /usr/local/bin/python3
lrwxrwxrwx 1 pyconfruser pyconfruser    7 Oct 24 21:01 python3.13 -> python3
```

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

itsdangerous 2.2.0

```
pip install itsdangerous
```

Latest version

Released: Apr 16, 2024

Safely pass data to untrusted environments and back.

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Owner

Pallets Projects

Download files

Download the file for your platform. If you're not sure which to choose, learn more about installing packages.

Source Distribution

itsdangerous-2.2.0.tar.gz (54.4 kB view hashes)

Uploaded Apr 16, 2024 `Source`

Built Distribution

itsdangerous-2.2.0-py3-none-any.whl (16.2 kB view hashes)

Uploaded Apr 16, 2024 `Python 3`

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.h

Installer = ajouter à `lib/pythonx.y/site-p`

itsdangerous 2.2.0

```
pip install itsdangerous
```

Safely pass data to untrusted environments and back.

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Owner

Pallets Projects

Download files

Download the file for your platfor

Source Distribution

itsdangerous-2.2.0.tar.gz (54.4

Uploaded Apr 16, 2024 `Source`

Built Distribution

itsdangerous-2.2.0-py3-none-

Uploaded Apr 16, 2024 `Python 3`

https://pypi.org/simple/itsdangerous/

Links for itsdangerous

itsdangerous-0.9.tar.gz
itsdangerous-0.9.1.tar.gz
itsdangerous-0.10.tar.gz
itsdangerous-0.11.tar.gz
itsdangerous-0.12.tar.gz
itsdangerous-0.13.tar.gz
itsdangerous-0.14.tar.gz
itsdangerous-0.15.tar.gz
itsdangerous-0.16.tar.gz
itsdangerous-0.17.tar.gz
itsdangerous-0.18.tar.gz
itsdangerous-0.19.tar.gz
itsdangerous-0.20.tar.gz
itsdangerous-0.21.tar.gz
itsdangerous-0.22.tar.gz
itsdangerous-0.23.tar.gz
itsdangerous-0.24.tar.gz
itsdangerous-1.1.0-py2.py3-none-any.whl
itsdangerous-1.1.0.tar.gz
itsdangerous-2.0.0a1-py3-none-any.whl
itsdangerous-2.0.0a1.tar.gz
itsdangerous-2.0.0rc1-py3-none-any.whl
itsdangerous-2.0.0rc1.tar.gz
itsdangerous-2.0.0rc2-py3-none-any.whl
itsdangerous-2.0.0rc2.tar.gz

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

```
PKG_NAME="itsdangerous-2.2.0"
PKG_TAR_NAME=$PKG_NAME".tar.gz"
PKG_TAR_URL="https://files.pythonhosted.org/packages/9c/cb/8ac0172223a
fbccb63986cc25049b154ecfb5e85932587206f42317be31d/itsdangerous-2.2.0.t
ar.gz"
wget $PKG_TAR_URL
tar -xzf $PKG_TAR_NAME
cp -R $PKG_NAME/src/itsdangerous
venvnopip/lib/python3.10/site-packages/itsdangerous
tree -L 5 venvnopip

venvnopip/bin/python -c "import itsdangerous;
print(dir(itsdangerous))"
```

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packa`

```
PKG_NAME="itsdangerous-2.2.0"
PKG_TAR_NAME=$PKG_NAME".tar.gz"
PKG_TAR_URL="https://files.pythonhosted.org/packages/9c/cb/8ac01
fbccb63986cc25049b154ecfb5e85932587206f42317be31d/itsdangerous-2
ar.gz"
wget $PKG_TAR_URL
tar -xzf $PKG_TAR_NAME
cp -R $PKG_NAME/src/itsdangerous
venvnopip/lib/python3.10/site-packages/itsdangerous
tree -L 5 venvnopip

venvnopip/bin/python -c "import itsdangerous;
print(dir(itsdangerous))"
```

```
itsdangerous-2.2.0.tar. 100%[===========================>]  53,13K  --.-KB/s    in 0,01s

2024-10-16 00:06:32 (4,85 MB/s) - 'itsdangerous-2.2.0.tar.gz.2' saved [54410/54410]

+ tar -xzf itsdangerous-2.2.0.tar.gz
+ cp -R itsdangerous-2.2.0/src/itsdangerous venvnopip/lib/python3.10/site-packages/itsdangerou
s
+ tree -L 5 venvnopip
venvnopip
├── bin
│   ├── activate
│   ├── activate.csh
│   ├── activate.fish
│   ├── Activate.ps1
│   ├── python -> /home/denis/.pyenv/versions/3.10.6/bin/python
│   ├── python3 -> python
│   └── python3.10 -> python
├── include
├── lib
│   └── python3.10
│       └── site-packages
│           └── itsdangerous
│               ├── encoding.py
│               ├── exc.py
│               ├── __init__.py
│               ├── _json.py
│               ├── py.typed
│               ├── serializer.py
│               ├── signer.py
│               ├── timed.py
│               └── url_safe.py
├── lib64 -> lib
└── pyvenv.cfg

8 directories, 17 files
+ venvnopip/bin/python -c 'import itsdangerous; print(dir(itsdangerous))'
['BadData', 'BadHeader', 'BadPayload', 'BadSignature', 'BadTimeSignature', 'HMACAlgorithm', 'N
oneAlgorithm', 'Serializer', 'SignatureExpired', 'Signer', 'TimedSerializer', 'TimestampSigner
', 'URLSafeSerializer', 'URLSafeTimedSerializer', '__builtins__', '__cached__', '__doc__', '__
file__', '__getattr__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '_jso
n', 'annotations', 'base64_decode', 'base64_encode', 'encoding', 'exc', 'serializer', 'signer'
, 't', 'timed', 'url_safe', 'want_bytes']
```

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

```
HTTPX_NAME="httpx-0.27.2"
HTTPX_TAR_NAME=$HTTPX_NAME".tar.gz"
HTTPX_TAR_URL="https://files.pythonhosted.org/packages/78/82/08f8c936781f67d9e6b9eeb8a0c8b4e40613
6ea4c3d1f89a5db71d42e0e6/httpx-0.27.2.tar.gz"
wget $HTTPX_TAR_URL
tar -xzf $HTTPX_TAR_NAME
mv $HTTPX_NAME/httpx venvnopip/lib/python3.10/site-packages/httpx
tree -L 5 venvnopip

venvnopip/bin/python -c "import httpx; print(dir(httpx))"
```

Installer dans un venv - tarball

https://docs.python.org/3/installing/index.html

Installer = ajouter à `lib/pythonx.y/site-packages`

```
HTTPX_NAME="httpx-0.27.2"
HTTPX_TAR_NAME=$HTTPX_NAME".tar.gz"
HTTPX_TAR_URL="https://files.pythonhosted.org/packages/78/82/08f8c936781f67d9e6b9eeb8a0c8b4e40613
6ea4c3d1f89a5db71d42e0e6/httpx-0.27.2.tar.gz"
wget $HTTPX_TAR_URL
tar -xzf $HTTPX_TAR_NAME
mv $HTTPX_NAME/httpx venvnopip/lib/python3.10/site-packages/httpx
tree -L 5 venvnopip

venvnopip/bin/python -c "import httpx; print(dir(httpx))"
```

HTTPX is a fully featured HTTP client for Python 3, which provides sync and async APIs, and support for both HTTP/1.1 and HTTP/2.

Installer dans un venv - tarball

https://docs.python.org/3/installin

Installer = ajouter à `lib/pythonx.y`

```
HTTPX_NAME="httpx-0.27.2"
HTTPX_TAR_NAME=$HTTPX_NAME".tar.gz"
HTTPX_TAR_URL="https://files.pythonhos
6ea4c3d1f89a5db71d42e0e6/httpx-0.27.2
wget $HTTPX_TAR_URL
tar -xzf $HTTPX_TAR_NAME
mv $HTTPX_NAME/httpx venvnopip/lib/pyt
tree -L 5 venvnopip

venvnopip/bin/python -c "import httpx;
```

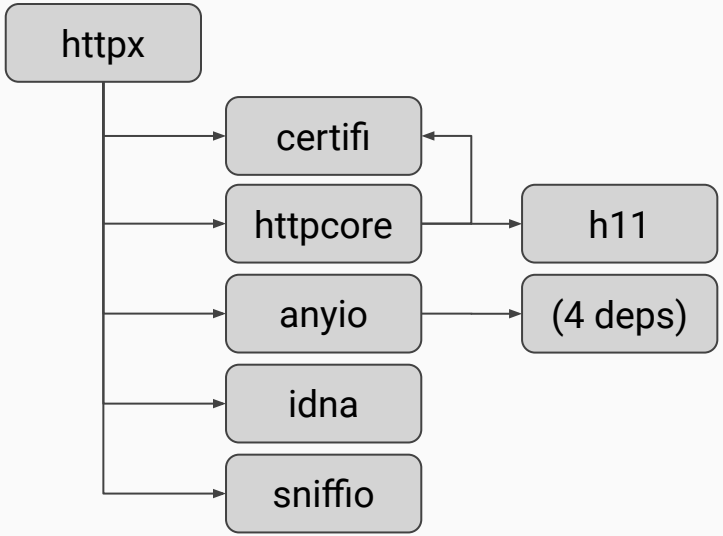
```
│   │   ├── _models.py
│   │   ├── _multipart.py
│   │   ├── py.typed
│   │   ├── _status_codes.py
│   │   ├── _transports
│   │   ├── _types.py
│   │   ├── _urlparse.py
│   │   ├── _urls.py
│   │   ├── _utils.py
│   │   └── __version__.py
├── lib64 -> lib
└── pyvenv.cfg

9 directories, 27 files
+ venvnopip/bin/python -c 'import httpx; print(dir(httpx))'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/__init__
.py", line 2, in <module>
    from ._api import *
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_api.py'
, line 6, in <module>
    from ._client import Client
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_client.
py", line 12, in <module>
    from ._auth import Auth, BasicAuth, FunctionAuth
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_auth.py
", line 12, in <module>
    from ._models import Cookies, Request, Response
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_models.
py", line 11, in <module>
    from ._content import ByteStream, UnattachedStream, encode_request, encode_response
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_content
.py", line 17, in <module>
    from ._multipart import MultipartStream
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_multipa
rt.py", line 16, in <module>
    from ._utils import (
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_utils.p
y", line 14, in <module>
    import sniffio
ModuleNotFoundError: No module named 'sniffio'
```

Installer dans un venv - tarball

httpx / pyproject.toml

Code | Blame | 133 lines (114 loc) · 3.59 KB

```
29    ]
30    dependencies = [
31        "certifi",
32        "httpcore==1.*",
33        "anyio",
34        "idna",
35        "sniffio",
```




```
│   │   ├── _models.py
│   │   ├── _multipart.py
│   │   ├── py.typed
│   │   ├── _status_codes.py
│   │   ├── _transports
│   │   ├── _types.py
│   │   ├── _urlparse.py
│   │   ├── _urls.py
│   │   ├── _utils.py
│   │   └── __version__.py
├── lib64 -> lib
└── pyvenv.cfg

9 directories, 27 files
+ venvnopip/bin/python -c 'import httpx; print(dir(httpx))'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/__init__
.py", line 2, in <module>
    from ._api import *
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_api.py
, line 6, in <module>
    from ._client import Client
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_client.
py", line 12, in <module>
    from ._auth import Auth, BasicAuth, FunctionAuth
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_auth.py
", line 12, in <module>
    from ._models import Cookies, Request, Response
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_models.
py", line 11, in <module>
    from ._content import ByteStream, UnattachedStream, encode_request, encode_response
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_content
.py", line 17, in <module>
    from ._multipart import MultipartStream
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_multipa
rt.py", line 16, in <module>
    from ._utils import (
  File "/home/denis/projects/pyconfr2024/venvnopip/lib/python3.10/site-packages/httpx/_utils.p
y", line 14, in <module>
    import sniffio
ModuleNotFoundError: No module named 'sniffio'
```

Installer dans un venv

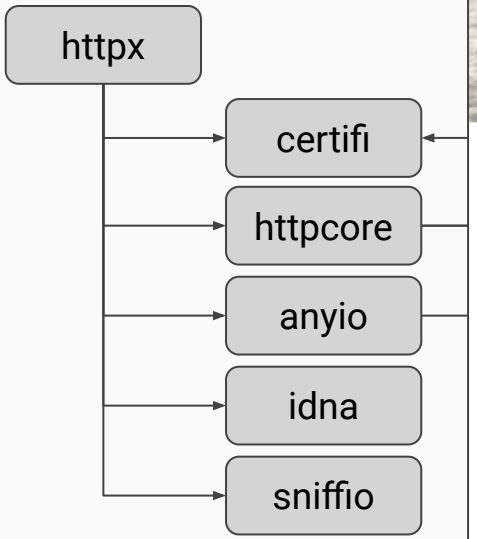
https://medium.com/@sdboyer/so-you-want-to-write-a-package-manager-4ae9c17d9527

So you want to write a package manager

sam boyer · Follow
50 min read · Feb 12, 2016

```
site-packages/httpx/__init__
site-packages/httpx/_api.py
site-packages/httpx/_client.
site-packages/httpx/_auth.py
site-packages/httpx/_models.
uest, encode_response
site-packages/httpx/_content
site-packages/httpx/_multipa
site-packages/httpx/_utils.p
```

```
httpx / pyproject.toml
Code  Blame  133 lines (114 loc)
29    ]
30    dependencies = [
31        "certifi",
32        "httpcore==1.*",
33        "anyio",
34        "idna",
35        "sniffio",
```



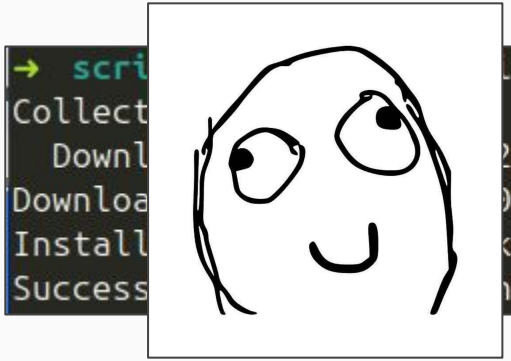
Installer dans un venv - wheel

Comment installer un fichier binaire ?

```
➜  scripts venv/bin/pip install --no-cache-dir chardet
Collecting chardet
  Downloading chardet-5.2.0-py3-none-any.whl.metadata (3.4 kB)
Downloading chardet-5.2.0-py3-none-any.whl (199 kB)
Installing collected packages: chardet
Successfully installed chardet-5.2.0
```

Installer dans un venv - wheel

Comment installer un fichier binaire ?



```
/tmp/tmp  mv ../chardet-5.2.0-py3-none-any.whl .
/tmp/tmp  ls
chardet-5.2.0-py3-none-any.whl
/tmp/tmp  unzip chardet-5.2.0-py3-none-any.whl
Archive:  chardet-5.2.0-py3-none-any.whl
  inflating: chardet/__init__.py
  inflating: chardet/__main__.py
  inflating: chardet/big5freq.py
  inflating: chardet/big5prober.py
  inflating: chardet/chardistribution.py
  inflating: chardet/charsetgroupprober.py
  inflating: chardet/charsetprober.py
  inflating: chardet/codingstatemachine.py
  inflating: chardet/codingstatemachinedict.py
  inflating: chardet/cp949prober.py
  inflating: chardet/enums.py
  inflating: chardet/escprober.py
  inflating: chardet/escsm.py
  inflating: chardet/eucjpprober.py
  inflating: chardet/euckrfreq.py
  inflating: chardet/euckrprober.py
  inflating: chardet/euctwfreq.py
```

Installer dans un venv - wheel

Comment installer un fichier binaire ?

```
/tmp/tmp  mv ../chardet-5.2.0-py3-none-any.whl .
/tmp/tmp  ls
chardet-5.2.0-py3-none-any.whl
/tmp/tmp  unzip chardet-5.2.0-py3-none-any.whl
Archive:  chardet-5.2.0-py3-none-any.whl
  inflating: chardet/__init__.py
  inflating: chardet/__main__.py
  inflating: chardet/big5freq.py
  inflating: chardet/big5prober.py
  inflating: chardet/chardistribution.py
  inflating: chardet/charsetgroupprober.py
  inflating: chardet/charsetprober.py
  inflating: chardet/codingstatemachine.py
  inflating: chardet/codingstatemachinedict.py
  inflating: chardet/cp949prober.py
  inflating: chardet/enums.py
  inflating: chardet/escprober.py
  inflating: chardet/escsm.py
  inflating: chardet/eucjpprober.py
  inflating: chardet/euckrfreq.py
  inflating: chardet/euckrprober.py
  inflating: chardet/euctwfreq.py
```

```
➜  scripts venv/bin/pip inst
Collecting chardet
  Downloading chardet-5.2.0-
Downloading chardet-5.2.0-py
Installing collected package
Successfully installed charc
```

https://packaging.python.org/en/latest/specifications/binary-distribution-format/

Installer dans un venv - wheel

Comment installer un fichier binaire ?

1. Parse et vérifie la compatibilité
2. Unpack dans site-packages
3. Unpack `.dist-info` et `.data`
4. Met les bons paths dans les scripts
5. Compile les `.py` en des `.pyc`

https://packaging.python.org/en/latest/specifications/binary-distribution-format/

Installer dans un venv - wheel

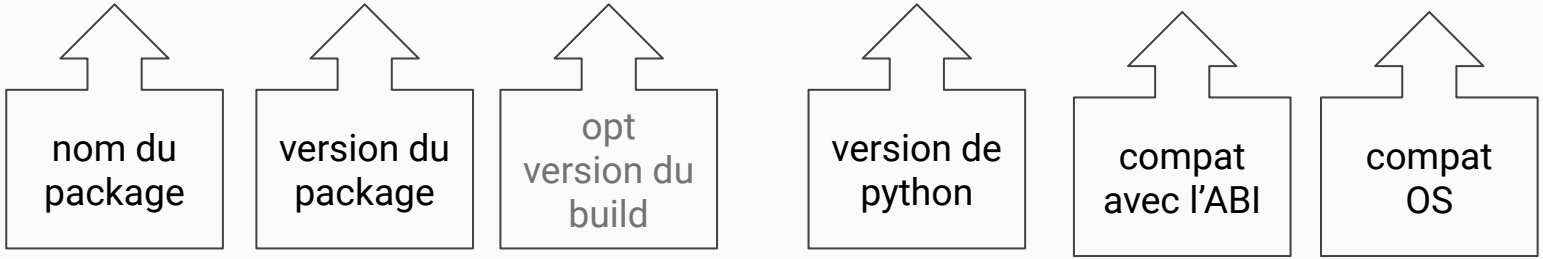
La compatibilité?

```
{distribution}-{version}(-{build tag})?-{python tag}-{abi tag}-{platform tag}.whl
```

Installer dans un venv - wheel

La compatibilité?

```
{distribution}-{version}(-{build tag})?-{python tag}-{abi tag}-{platform tag}.whl
```




```
pandas-2.2.3-cp313-cp313t-musllinux_1_2_x86_64.whl
```

Installer dans un venv - wheel

Comment installer un fichier binai

```
/tmp/tmp/downpg2  unzip psycopg2_binary-2.9.10-cp313-cp313-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl
Archive:  psycopg2_binary-2.9.10-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
   creating: psycopg2_binary-2.9.10.dist-info/
   creating: psycopg2_binary.libs/
   creating: psycopg2/
  inflating: psycopg2_binary-2.9.10.dist-info/LICENSE
  inflating: psycopg2_binary-2.9.10.dist-info/top_level.txt
  inflating: psycopg2_binary-2.9.10.dist-info/RECORD
  inflating: psycopg2_binary-2.9.10.dist-info/METADATA
  inflating: psycopg2_binary-2.9.10.dist-info/WHEEL
  inflating: psycopg2_binary.libs/libpq-e8a033dd.so.5.16
  inflating: psycopg2_binary.libs/libk5crypto-b1f99d5c.so.3.1
  inflating: psycopg2_binary.libs/libkrb5support-d0bcff84.so.0.1
  inflating: psycopg2_binary.libs/libselinux-0922c95c.so.1
  inflating: psycopg2_binary.libs/libkrb5-fcafa220.so.3.3
  inflating: psycopg2_binary.libs/liblber-e0f57070.so.2.0.200
  inflating: psycopg2_binary.libs/libcom_err-2abe824b.so.2.1
  inflating: psycopg2_binary.libs/libkeyutils-dfe70bd6.so.1.5
  inflating: psycopg2_binary.libs/libcrypto-ea28cefb.so.1.1
  inflating: psycopg2_binary.libs/libssl-3e69114b.so.1.1
  inflating: psycopg2_binary.libs/libpcre-9513aab5.so.1.2.0
  inflating: psycopg2_binary.libs/libgssapi_krb5-497db0c6.so.2.2
  inflating: psycopg2_binary.libs/libldap-c37ed727.so.2.0.200
  inflating: psycopg2_binary.libs/libsasl2-883649fd.so.3.0.0
  inflating: psycopg2/pool.py
  inflating: psycopg2/errors.py
  inflating: psycopg2/_json.py
  inflating: psycopg2/extensions.py
  inflating: psycopg2/extras.py
  inflating: psycopg2/sql.py
  inflating: psycopg2/_range.py
  inflating: psycopg2/_ipaddress.py
  inflating: psycopg2/tz.py
  inflating: psycopg2/__init__.py
  inflating: psycopg2/_psycopg.cpython-313-x86_64-linux-gnu.so
  inflating: psycopg2/errorcodes.py
```

Installer dans un venv - wheel

Comment installer un fichier binai

.so - dynamic libraries

code Python

```
/tmp/tmp/downpg2  unzip psycopg2_binary-2.9.10-cp313-cp313-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl
Archive:  psycopg2_binary-2.9.10-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
   creating: psycopg2_binary-2.9.10.dist-info/
   creating: psycopg2_binary.libs/
   creating: psycopg2/
  inflating: psycopg2_binary-2.9.10.dist-info/LICENSE
  inflating: psycopg2_binary-2.9.10.dist-info/top_level.txt
  inflating: psycopg2_binary-2.9.10.dist-info/RECORD
  inflating: psycopg2_binary-2.9.10.dist-info/METADATA
  inflating: psycopg2_binary-2.9.10.dist-info/WHEEL
  inflating: psycopg2_binary.libs/libpq-e8a033dd.so.5.16
  inflating: psycopg2_binary.libs/libk5crypto-b1f99d5c.so.3.1
  inflating: psycopg2_binary.libs/libkrb5support-d0bcff84.so.0.1
  inflating: psycopg2_binary.libs/libselinux-0922c95c.so.1
  inflating: psycopg2_binary.libs/libkrb5-fcafa220.so.3.3
  inflating: psycopg2_binary.libs/liblber-e0f57070.so.2.0.200
  inflating: psycopg2_binary.libs/libcom_err-2abe824b.so.2.1
  inflating: psycopg2_binary.libs/libkeyutils-dfe70bd6.so.1.5
  inflating: psycopg2_binary.libs/libcrypto-ea28cefb.so.1.1
  inflating: psycopg2_binary.libs/libssl-3e69114b.so.1.1
  inflating: psycopg2_binary.libs/libpcre-9513aab5.so.1.2.0
  inflating: psycopg2_binary.libs/libgssapi_krb5-497db0c6.so.2.2
  inflating: psycopg2_binary.libs/libldap-c37ed727.so.2.0.200
  inflating: psycopg2_binary.libs/libsasl2-883649fd.so.3.0.0
  inflating: psycopg2/pool.py
  inflating: psycopg2/errors.py
  inflating: psycopg2/_json.py
  inflating: psycopg2/extensions.py
  inflating: psycopg2/extras.py
  inflating: psycopg2/sql.py
  inflating: psycopg2/_range.py
  inflating: psycopg2/_ipaddress.py
  inflating: psycopg2/tz.py
  inflating: psycopg2/__init__.py
  inflating: psycopg2/_psycopg.cpython-313-x86_64-linux-gnu.so
  inflating: psycopg2/errorcodes.py
```

Installer dans un venv - dist-info

C’est quoi ce dossier ?

```
➜  scripts ls venv/lib/python3.12/site-packages/pandas-2.2.3.dist-info
entry_points.txt  INSTALLER  LICENSE  METADATA  RECORD  REQUESTED  WHEEL
➜  scripts
```

Installer dans un venv - dist-info

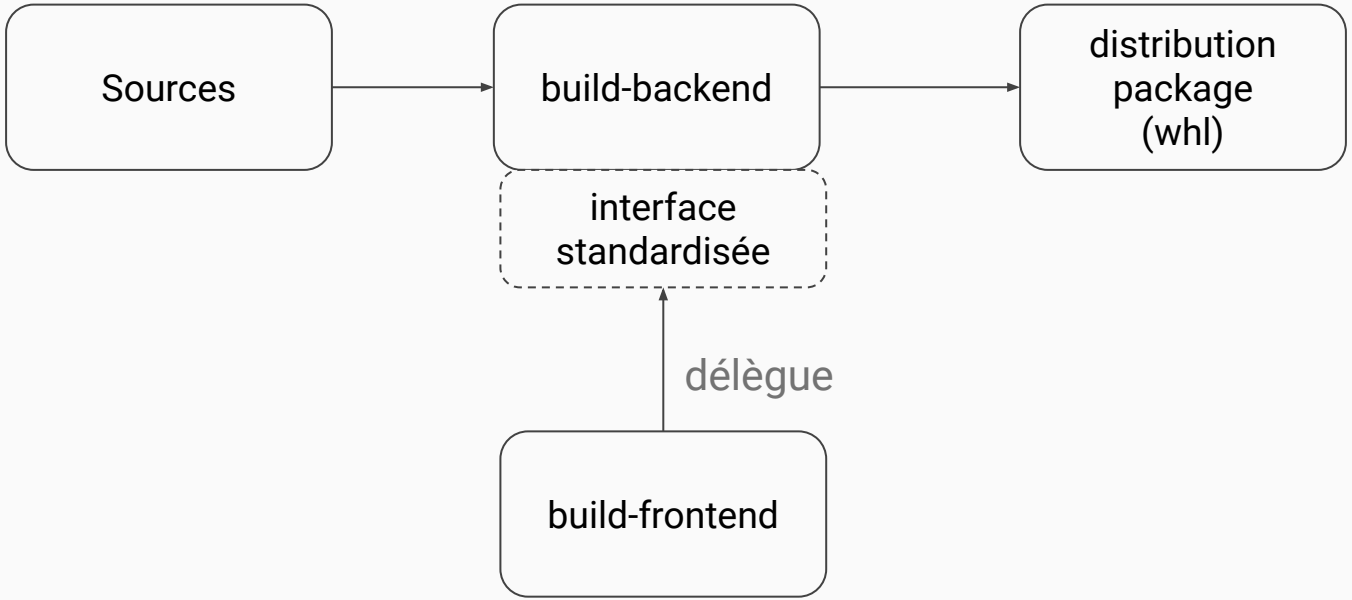
C’est quoi ce dossier ?

```
→ scripts ls venv/lib/python3.12/site-packages/pandas-2.2.3.dist-info
entry_points.txt  INSTALLER  LICENSE  METADATA  RECORD  REQUESTED  WHEEL
→ scripts
```

- **METADATA**: contains project metadata
- **RECORD**: liste des fichiers installés
- **WHEEL**: config d’installation de la wheel
- INSTALLER: nom de l’outil pour installer
- entry_points.txt: liste des entrypoints disponibles
- direct_url.json: quand on install directement depuis une URL
- n’importe quel autre fichier

Installer dans un venv - wheel

Build-backend vs build-frontend



6. Les outils pour gérer

Aller plus loin

Pourquoi ?

- Plusieurs installations Python
- Séparation automatique
- Plus vite

pyenv

https://github.com/pyenv/pyenv

Séparation d’environnements Python

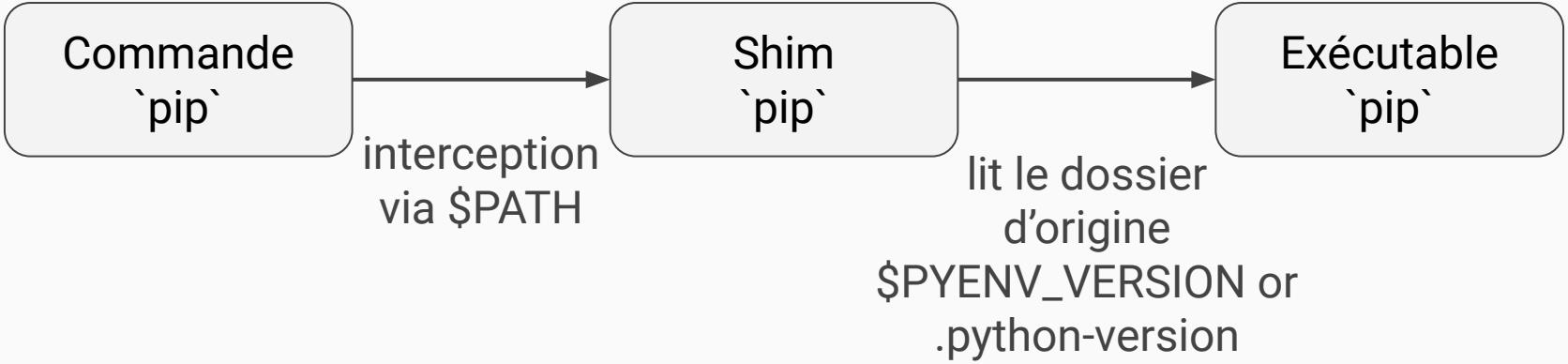
- Plusieurs installations Python
- Interception de commandes Python via des SHIMs

pyenv

https://github.com/pyenv/pyenv

Séparation d'environnements Python

- Plusieurs installations Python
- Interception de commandes Python via des SHIMs



pipx

https://pipx.pypa.io/stable/

Installation d’applications Python

- Installer en isolation et dispo pour le shell
- Virtualenv séparés dans `~/.local/share/pipx/venvs/PACKAGE`
-

```
sudo apt update
sudo apt install pipx
pipx ensurepath
```

pipx

https://pipx.pypa.io/stable/

Installation d'applications Python

- Installer en isolation et dispo pour le shell
- Virtualenv séparés dans `~/.local/share/pipx/venvs/PACKAGE`
- One-off avec `pipx run`

```
sudo apt update
sudo apt install pipx
pipx ensurepath
```

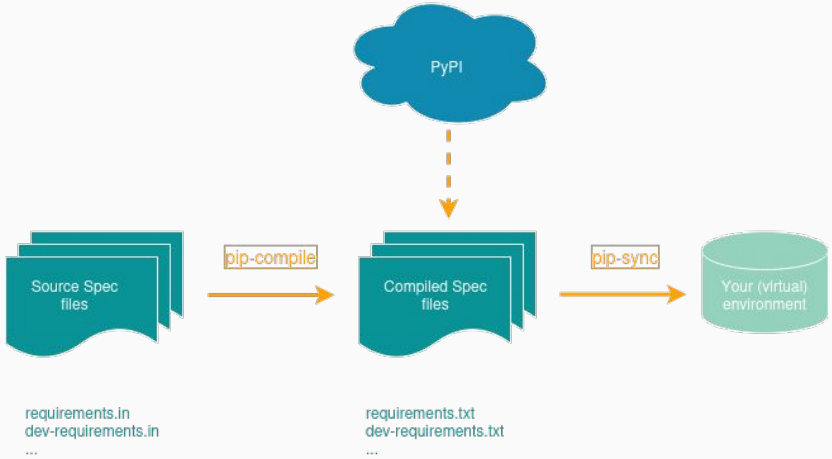
```
~ pipx run pycowsay moooo!                                          ✔ 22:17:21

  ------
< moooo! >
  ------
   \   ^__^
    \  (oo)\_______
       (__)\       )\/\
           ||----w |
           ||     ||

~ pipx uninstall pycowsay                                           ✔ 22:17:27
Nothing to uninstall for pycowsay 😴
~ pipx install pycowsay                                             1 ✘ 22:17:34
  installed package pycowsay 0.0.0.2, installed using Python 3.12.3
  These apps are now globally available
    - pycowsay
  These manual pages are now globally available
    - man6/pycowsay.6
done! ✨ 🌟 ✨
~ pipx run pycowsay moooo!
⚠️ pycowsay is already on your PATH and installed at /home/denis/.local/bin/pycowsay. Downloading and running anyway.

  ------
< moooo! >
  ------
   \   ^__^
    \  (oo)\_______
       (__)\       )\/\
           ||----w |
           ||     ||
```

pip-tools

https://github.com/jazzband/pip-tools

Facilement garder ses dépendances à jour

- `pip-compile` et `pip-sync`
- virtualenv laissé à l'utilisateur
- `requirements.txt` devient un fichier de lock



poetry

https://python-poetry.org/

Project manager

- Gestion automatique du virtualenv
- Introduction du `pyproject.toml`
- Pin avec un lock file (`poetry.lock`)

```
$ poetry add pendulum

Using version ^2.0.5 for pendulum

Updating dependencies
Resolving dependencies... (1.5s)

Package operations: 4 installs, 0 updates, 0 removals

  - Installing six (1.13.0): Downloading... 25%
  - Updating pytzdata (2019.3 -> 2020.4): Installing...
  - Installing pendulum (2.0.5)

Writing lock file
```

If you want to get basic information about the currently activated virtual environment, you can use the `env info` command

```
poetry env info
```

will output something similar to this:

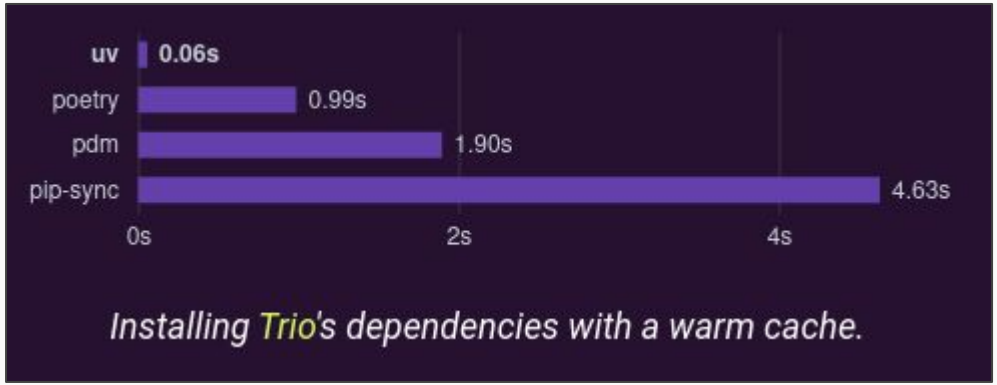
```
Virtualenv
Python:         3.7.1
Implementation: CPython
Path:           /path/to/poetry/cache/virtualenvs/test-O3eWbxRl-py3.7
Valid:          True

Base
Platform: darwin
OS:       posix
Python:   /path/to/main/python
```

uv

https://docs.astral.sh/uv/

Un outil pour les amener tous, et dans les ténèbres les lier

- Écrit en Rust par les auteurs de `ruff`
- “Remplace pip, pipx, pip-tools, pyenv, virtualenv”
- Notion de projet par dossier de travail
- virtualenv géré automatiquement

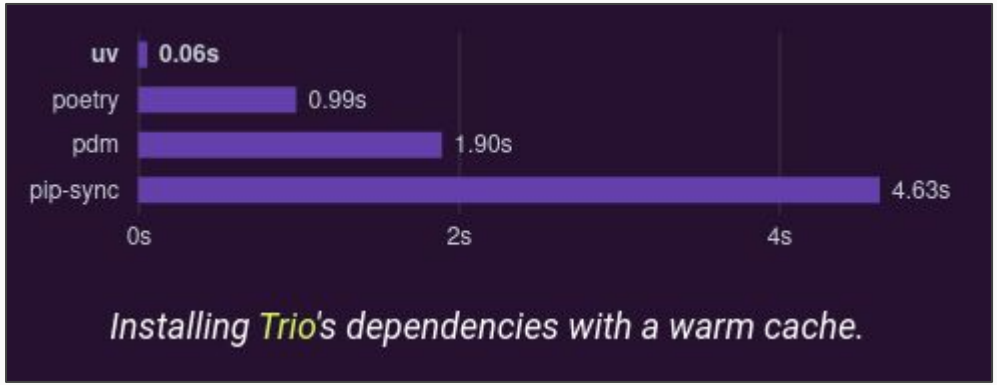


uv

https://docs.astral.sh/uv/

Un outil pour les amener tous, et dans les ténèbres les lier

- Écrit en Rust par les auteurs de `ruff`
- “Remplace pip, pipx, pip-tools, pyenv, virtualenv”
- Notion de projet par dossier de travail
- virtualenv géré automatiquement




```
root@80119c87896f:/tmp/scripts# uv
An extremely fast Python package manager.

Usage: uv [OPTIONS] <COMMAND>

Commands:
  run      Run a command or script
  init     Create a new project
  add      Add dependencies to the project
  remove   Remove dependencies from the project
  sync     Update the project's environment
  lock     Update the project's lockfile
  export   Export the project's lockfile to an alternate
  tree     Display the project's dependency tree
  tool     Run and install commands provided by Python pa
  python   Manage Python versions and installations
  pip      Manage Python packages with a pip-compatible i
  venv     Create a virtual environment
  build    Build Python packages into source distribution
  publish  Upload distributions to an index
  cache    Manage uv's cache
  self     Manage the uv executable
  version  Display uv's version
  help     Display documentation for a command
```

pdm

Similaire à uv

- (pas d’expérience personnelle avec)
- utilisation de `__pypackages__`
- peut utiliser uv comme installer

pex/shiv

https://github.com/pex-tool/pex

Générer un exécutable via un virtualenv

- https://www.youtube.com/watch?v=NmpnGhRwsu0
- Combinaison virtualenv + zip + shebang
- On peut bouger des “virtualenv” pour exécuter des scripts

```
➜  scripts cat <<EOF > myapp.py
from flask import Flask
app = Flask(__name__)
@app.get("/")
def say_hello():
    return "<p>Hello</p>"
app.run()
EOF
➜  scripts venv/bin/pex flask -o flask.pex
➜  scripts ./flask.pex myapp.py
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use
oduction WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

pex/shiv

https://github.com/pex-tool/pex

Générer un exécutable via un virtualenv (exemple repris du lightning talk)

```
❯ cat <<EOF > hello_world.py
def say_hello():
    print("hello potato")
say_hello()
EOF

❯ python .
/home/denis/.pyenv/versions/3.11.4/bin/python: can't find '__main__' module in '/tmp/project'
❯ mv hello_world.py __main__.py
❯ python .
hello potato
```

pex/shiv

https://github.com/pex-tool/pex

Générer un exécutable via un virtualenv (exemple repris du lightning talk)

```
❯ cat <<EOF > hello_world.py
def say_hello():
    print("hello potato")
say_hello()
EOF

❯ python .
/home/denis/.pyenv/versions/3.11.4/bin/python: can't find '__main__' module in '/tmp/project'
❯ mv hello_world.py __main__.py
❯ python .
hello potato
```

D'où le `if __name__ == "__main__":`

pex/shiv

https://github.com/pex-tool/pex

Générer un exécutable via un virtualenv (exemple repris du lightning talk)

```
❯ zip myworld.zip __main__.py
  adding: __main__.py (deflated 22%)
❯ python myworld.zip
hello potato
❯ cat <(echo '#!/usr/bin/env python') myworld.zip > mymodule.exec
❯ chmod u+x mymodule.exec
❯ ./mymodule.exec
hello potato
❯ head -1 mymodule.exec
#!/usr/bin/env python
❯ unzip mymodule.exec
Archive:  mymodule.exec
warning [mymodule.exec]:  22 extra bytes at beginning or within zipfile
  (attempting to process anyway)
replace __main__.py? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: __main__.py
```

pex/shiv

https://github.com/pex-tool/pex

Générer un exécutable via un virtualenv (exemple repris du lightning talk)

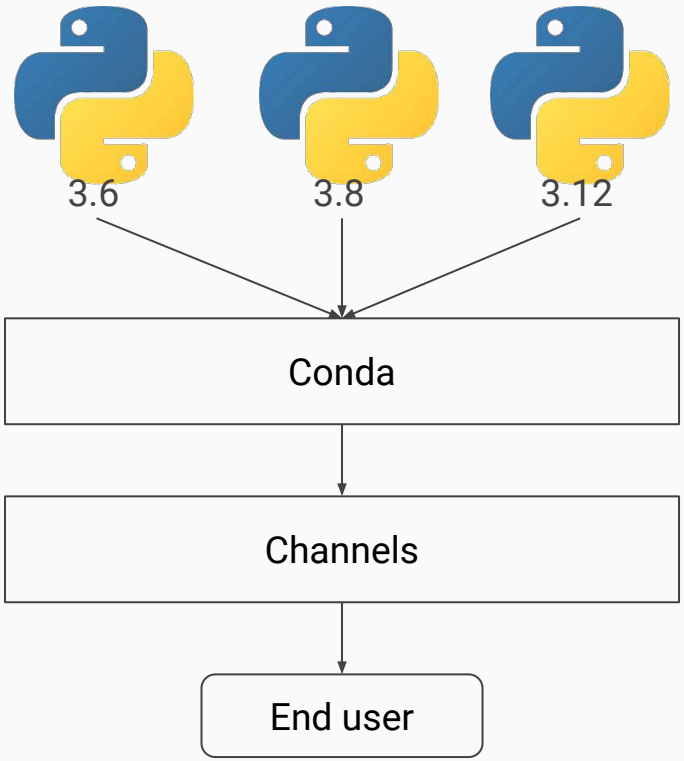
```
➜  scripts cat <<EOF > myapp.py
from flask import Flask
app = Flask(__name__)
@app.get("/")
def say_hello():
    return "<p>Hello</p>"
app.run()
EOF
➜  scripts venv/bin/pex flask -o flask.pex
➜  scripts ./flask.pex myapp.py
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use
oduction WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

```
❯ ls -a
.  ..  flask.pex  mini_server.py  venv
❯ head -1 flask.pex
#!/usr/bin/env python3.11
❯ mkdir -p ./flask_pex
❯ unzip -q -d ./flask_pex flask.pex
❯ tree -a -L 2 flask_pex
flask_pex
├── .bootstrap
│   └── pex
├── .deps
│   ├── blinker-1.8.2-py3-none-any.whl
│   ├── click-8.1.7-py3-none-any.whl
│   ├── flask-3.0.3-py3-none-any.whl
│   ├── itsdangerous-2.2.0-py3-none-any.whl
│   ├── jinja2-3.1.4-py3-none-any.whl
│   ├── MarkupSafe-3.0.2-cp311-cp311-manylinu
│   └── werkzeug-3.1.0-py3-none-any.whl
├── __main__.py
├── __pex__
│   └── __init__.py
└── PEX-INFO

12 directories, 3 files
```

conda

À la fois environment manager et package manager

- `pyenv` + `pip`
- notion de channels
- gestion de `Python`, R, C, `C++`



Conclusion

Conclusion

Conclusion

On peut comprendre
ce qu’il y a
dans nos outils

Conclusion

On peut comprendre
ce qu'il y a
dans nos outils

Installer des
packages
c'est surtout
`unzip+mv`

Conclusion

On peut comprendre ce qu'il y a dans nos outils

Installer des packages c'est surtout `unzip+mv`

… mais il y a plein de subtilités

Merci pour votre attention

Annexe