# python の仮想環境管理 venv と pipx

東京エリア・関西合同 Debian 勉強会

Norimitsu SUGIMOTO (杉本 典充)
dictoss@live.jp

2024-05-18

# アジェンダ

- 自己紹介
- これまでの python パッケージのインストール
- Debian 12 bookworm における python の変更
- python の仮想環境 venv
- アプリケーション向け仮想環境管理 pipx
- これまで通りのやり方をしたいあなたに
- まとめ
- 参考資料

# 自己紹介

- Norimitsu SUGIMOTO (杉本 典充)
- dictoss@live.jp
- Twitter: @dictoss
- Debian を使い始めたのは 3.1 sarge が testing の頃
- 仕事はソフトウェア開発者をやってます
- python と Django の組み合わせで使うことが多いです

# これまでのpythonパッケージのインストール

# これまでの python パッケージのインストール

- これまでは特に気にせずに以下コマンドを実行

```
# pip3 install xyz
```

- Debian 12 bookworm で実行した場合

```
# cat /etc/debian_version
12.5

# pip3 install xyz
error: externally-managed-environment

× This environment is externally managed
|--> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.
  (以下略)
```

- エラーになって困りませんでしたか？

# Debian 12 bookworm における python の変更

# PEP-668 と externally-managed (1)

- Debian 12 bookworm リリースノート[1]
  - Debian が提供する python3 インタプリタパッケージ (python3.11 および pypy3) は、PEP-668 に従って externally-managed とマークされるようになりました。
  - Debian で提供されるバージョンの python3-pip はこれに伴って、–break-system-packages オプションが指定されない限り、Debian の python インタプリタ環境へ手作業でのパッケージインストールを拒否します。
  - 詳細については /usr/share/doc/python3.11/README.venv を参照してください。
- よくわからない言葉が出てきます
  - PEP-668、externally-managed
  - python3-pip、–break-system-packages オプション

[1] https://www.debian.org/releases/bookworm/amd64/release-notes/ch-information.ja.html#python3-pep-668

# PEP-668 と externally-managed (2)

- PEP-668[2]
  - Marking Python base environments as “ externally managed ”
  - python を実行する環境より外部の環境で python が管理される、という意味
  - 外部の環境 = Debian における deb パッケージによる管理
- externally-managed として管理されてることを示すファイル
  - /usr/lib/python3.11/EXTERNALLY-MANAGED
  - 中身は "pip3 install xyz" を実行したときのエラーメッセージとほぼ同じ
  - この状態で「pip3 install xyz」を実行するとエラーで止まる

[2] https://peps.python.org/pep-0668/

# PEP-668 と externally-managed (3)

- Debian ではパッケージ化されていない python パッケージをインストールするには？
  - 【非推奨】「pip3 install xyz –break-system-packages」で強引に実行
  - 【推奨】python の仮想環境を作成して、その仮想環境の中へ python パッケージをインストールする
    - venv を使う (python ライブラリ向け)
    - pipx を使う (python アプリケーション向け)

# python の仮想環境 venv

# venv モジュール

- venv — 仮想環境の作成
  - https://docs.python.org/ja/3.11/library/venv.html
- 2012/9/29 に リリースした python3.3 から標準機能として利用できる仮想環境の作成手段
- 以下コマンドで仮想環境を作成できる

```
$ mkdir -p myvenv
$ cd myvenv
$ python3 -m venv .
$ ls
bin  include  lib  lib64  pyvenv.cfg
$ cat pyvenv.cfg
home = /usr/bin
include-system-site-packages = false
version = 3.11.2
executable = /usr/bin/python3.11
command = /usr/bin/python3 -m venv /home/dictoss/myvenv
```

# venv を使った仮想環境のパッケージ管理 (1)

bin/activate を実行すると仮想環境へ入る
仮想環境の中の bin ディレクトリが PATH 環境変数へ追加され、python コマンドと pip コマンドは仮想環境の中のコマンドを優先して実行するよう切り替える

```
$ ls bin
Activate.ps1  activate.csh   pip   pip3.11  python3
activate      activate.fish  pip3  python   python3.11

$ source bin/activate
(myvenv) $ which python
/home/dictoss/myvenv/bin/python
(myvenv) $ which pip
/home/dictoss/myvenv/bin/pip
```

deactivate を実行すると仮想環境を抜ける

```
(myvenv) $ deactivate
$
```

# venv を使った仮想環境のパッケージ管理 (2)

pip コマンドで python ライブラリをインストールしたとき、
仮想環境の中の lib ディレクトリへインストールされる

```
(myvenv) $ pip install python-dateutil
(myvenv) $ pip freeze
python-dateutil==2.9.0.post0
six==1.16.0

$ ls lib/python3.11/site-packages/
__pycache__                 pkg_resources
_distutils_hack             python_dateutil-2.9.0.post0.dist-info
dateutil                    setuptools
distutils-precedence.pth    setuptools-66.1.1.dist-info
pip                         six-1.16.0.dist-info
pip-23.0.1.dist-info        six.py
```

python-dateutil ライブラリを仮想環境内で利用できている

```
(myvenv) $ python
>>> import dateutil.parser
>>> dateutil.parser.parse('Sat, 18 May 2024 14:15:16 +0900')
datetime.datetime(2024, 5, 18, 14, 15, 16, tzinfo=tzoffset(None, 32400))
```

# アプリケーション向け仮想環境管理 pipx

# pipx とは

- pipx は python のアプリケーション単位で仮想環境を管理するコマンド
  - venv を使った仮想環境は基本的にアプリケーションごとに用意すると良いのだが、多いと管理がつらい
  - その仮想環境の管理を任せる仕組みが pipx
- pypi で提供されており[3]、debian では pipx パッケージで提供している

```
$ sudo apt-get install pipx
$ pipx --version
1.1.0
```

[3] https://pypi.org/project/pipx/

# pipx の使い方 (パッケージ単体の場合)

pipx のインストール後は、pipx ensurepath を実行する

```
$ pipx ensurepath
Success! Added /home/dictoss/.local/bin to the PATH environment variable.

Consider adding shell completions for pipx. Run 'pipx completions' for
instructions.

You will need to open a new terminal or re-login for the PATH changes
to take effect.

Otherwise pipx is ready to go!
```

${HOME}/.local/bin が PATH に追加される

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/home/dictoss/.local/bin

$ grep -1 pipx ~/.bashrc

# Created by `pipx` on 2024-05-15 13:51:28
export PATH="$PATH:/home/dictoss/.local/bin"
```

# pipx の使い方 (パッケージ単体の場合)

例：python アプリケーションの pelican をインストールする

```
$ pipx install pelican
  installed package pelican 4.9.1, installed using Python 3.11.2
  These apps are now globally available
    - pelican
    - pelican-import
    - pelican-plugins
    - pelican-quickstart
    - pelican-themes
done!
```

pelican コマンドが ${HOME}/.local/bin/pelican に追加され、実体は venv で作成した仮想環境の中のインストールした pelican コマンドのシンボリックリンクになっている

```
$ which pelican
/home/dictoss/.local/bin/pelican

$ ls -l /home/dictoss/.local/bin/pelican
lrwxrwxrwx 1 dictoss dictoss 53  5 月 15 23:04
  /home/dictoss/.local/bin/pelican
  -> /home/dictoss/.local/pipx/venvs/pelican/bin/pelican
```

# pipx の使い方 (requirements.txt を使う場合)

既存で動いている python アプリケーションがあり、それは requirements.txt でバージョンが指定されている場合

```
$ git clone https://github.com/dictoss/xddc.git
$ cd xddc
$ cat requirements.txt
pelican==4.8.0
Jinja2==3.1.3
```

requirements.txt に書かれている pelican==4.8.0 がインストールされる

```
$ pipx install pelican --pip-args "-r requirements.txt"
installed package pelican 4.8.0, installed using Python 3.11.2
These apps are now globally available
  - pelican
  - pelican-import
  - pelican-plugins
  - pelican-quickstart
  - pelican-themes
done!
```

通方あ
までやり
これのりた
りをしい
なたに

# pip3 の –break-system-packages オプション

- python3-pip パッケージをインストールするとこれまで通り pip3 コマンドを利用できる
- pip3 コマンドに –break-system-packages オプションを付与して実行すれば、これまで通り python パッケージは一応インストールできる
  - pip3 install xyz –break-system-packages
  - pip3 uninstall xyz –break-system-packages
- pip3 コマンドで –break-system-packages オプションを指定する方法を続けてもよさそうな状況
  - 1 台のホスト環境の中で python アプリケーションを 1 つしか動かさない場合
  - 例）docker、k8s、lxc、chroot、KVM 上の仮想マシンなどで Debian 環境自体を分離して実行している

# まとめ

# まとめ

- Debian 12 bookworm で python アプリケーションを使う場合に覚えておくとよい仕様変更の「externally-managed」
- python の標準機能の仮想環境を作成する venv
- python アプリケーションごとに仮想環境を作成・管理する手間を減らせる pipx コマンド
- これまで通りのスタイルを通すなら python3-pip の –break-system-packages という奥の手

# 参考文献


- pipx - pypi https://pypi.org/project/pipx/
- PEP 668 https://peps.python.org/pep-0668/
- venv モジュール https://docs.python.org/ja/3.11/library/venv.html