# Środowiska wirtualne i zarządzanie zależnościami w Pythonie

Piotr Grzesik

# Dwa słowa o mnie


# Pip

(https://github.com/pypa/pip)

- Rekomendowany przez PyPA (Python Packaging Authority) instalator pakietów
- Dystrybuowany wraz z Pythonem >=2.7.9 i >=3.4
- Domyślnie instaluje paczki z pypi.python.org/pypi

# Pip - przydatne komendy

```
pip install package
pip install 'package>=1.0'
pip install package==1.0
pip install -r requirements.txt

pip uninstall package

pip list
pip list --outdated

pip freeze
pip freeze > requirements.txt

pip show package

pip search "query"
```

# Pyenv - zarządzamy wersjami Pythona

(https://github.com/pyenv/pyenv)

- Narzędzie do zarządzania wersjami intepretera Pythona
- Wspiera CPython, PyPy, Stackless, JPython, IronPython
- Podobne do nvm (node.js) czy rvm, rbenv (Ruby)
- Zarządzanie globalną wersją dla użytkownika
- Konfiguracja wersji dla konkretnego projektu
- Brak wsparcia dla Windowsa

# Jak działa pyenv - shims

Pyenv dodaje na początku PATH scieżkę *~/.pyenv/shims*, w której znajdują się *shimy*, będące plikami wykonywalnymi, odpowiedzialnymi za przekazanie wywołania do pyenva.

Gdy chcemy wywołać komendę np. **pip**, wykonane zostają następujące kroki:

1. Przeszukanie PATH w poszukiwaniu pliku wykonywalnego pip
2. Znalezienie *shim* pip
3. Wykonanie *shim*, który przekazuje wywołanie do pyenva

# Kolejność wyboru wersji interpretera

1. Zmienna środowiskowa PYENV_VERSION (komenda **pyenv shell**)
2. Plik .python-version w aktualnym katalogu (można go utworzyć komendą **pyenv local**)
3. Plik .python-version w katalogu nadrzędnym (aż do /)
4. Plik *~/.pyenv/version* (komenda **pyenv global**)

# Pyenv update

(https://github.com/pyenv/pyenv-update)

- Plugin dodający możliwość aktualizacji pyenv'a za pomocą komendy **pyenv update**
- Aktualizuje zarówno pyenva, jak i inne zainstalowane pluginy np. pyenv-virtualenv

# Demo pyenv

# Virtualenvs - środowiska wirtualne w Pythonie

- Środowiska izolowane od siebie i systemowego Pythona
- Każde ze środowisk ma odrębne zależności (np. środowiska mające różne wersje tej samej biblioteki)
- Pozwala używać bibliotek(zewnętrznych lub własnych), bez zaśmiecania systemowego Pythona
- Od Pythona 3.3 część biblioteki standardowej (PEP 405)

# Virtualenv - jak używać

## Python < 3.3

```
~ pip install virtualenv
~ virtualenv /path/to/env
```

## Python 3.3 i wyżej (deprecated od wersji 3.6)

```
~ pyvenv /path/to/env
```

## Rekomendowane od wersji 3.5

```
~ python3 -m venv /path/to/env
```

## Aktywacja utworzonego virtualenva

```
~ source /path/to/env/bin/activate
```

## Deaktywacja utworzonego virtualenva

```
~ deactivate
```

# Virtualenv - jak działa ?

- Activate - dodaje na początku PATH ścieżkę */path/to/env/bin*
- sys.path - ścieżki wyszukiwania dla modułów podczas importów
- 3rd party virtualenv - zmodyfikowany *site.py*, *orig-prefix.txt* pozwala na dołączenie biblioteki standardowej (sys.prefix)
- Wbudowany virtualenv - plik *pyvenv.cfg* (sys.prefix i sys.base_prefix)
- Pakiety instalowane są do */path/to/env/lib/pythonX.Y/site-packages*

Przykładowy plik *pyvenv.cfg*:

```
home = /usr/bin
include-system-site-packages = false
version = 3.5.2
```

# Pyenv + virtualenv

(https://github.com/pyenv/pyenv-virtualenv)

- Plugin do pyenva pozwalający na zarządzanie virtualenvami, a także środowiskami stworzonymi za pomocą conda
- Wspiera zarówno virtualenv jak i venv będący częścią biblioteki standardowej

# Pyenv-virtualenv demo

# Conda

(https://github.com/conda/conda)

- Pozwala zarządzać zarówno pakietami jak i środowiskami
- Wieloplatformowość (Win/macOS/Linux)
- Zarządzanie wersjami Pythona (od 2.6.8 do 3.6+)
- Instalacja paczek wewnątrz utworzonych środowisk również za pomocą pip'a
- Instalacja binarnych pakietów (brak konieczności kompilacji podczas instalacji)
- Możliwość instalacji pakietów napisanych w innych językach np. C/C++, Java, Scala, JavaScript, FORTRAN, R

# Zarządzanie zależnościami - dlaczego ?

- Lista bibliotek, z których korzysta nasza aplikacja/skrypt (najczęściej plik *requirements.txt*)
- Logiczny podział zależności (np. oddzielna lista zależności do testów, oddzielna lista dla ludzi i maszyn)
- **Stabilne i powtarzalne buildy na różnych środowiskach**

# pipreqs

(https://github.com/bndr/pipreqs)

Pozwala na generowanie pliku z zależnościami na podstawie importów w projekcie

Przykład użycia:

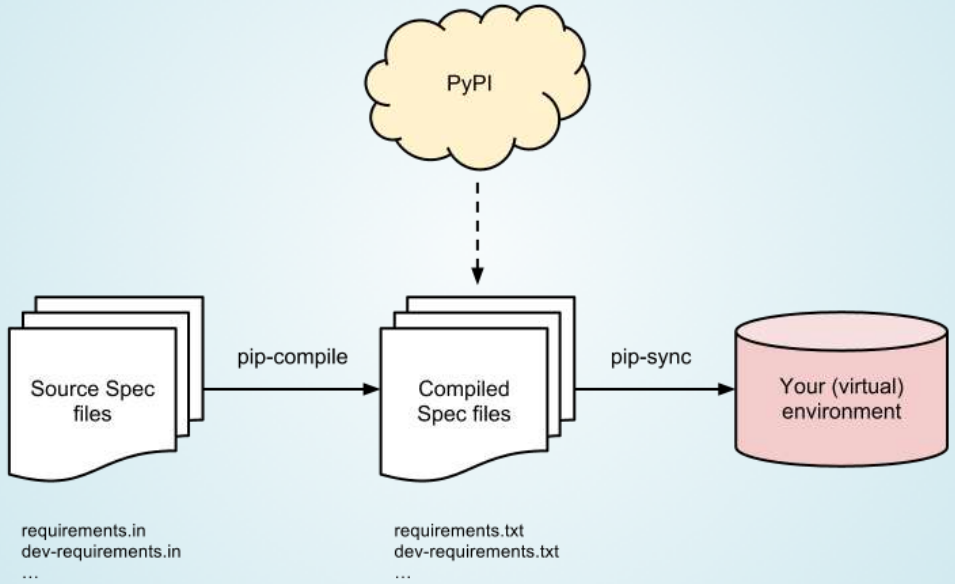
```
pipreqs /path/to/project
```

Przykładowy rezultat *requirements.txt*:

```
wheel==0.23.0
Yarg==0.1.9
docopt==0.6.2
```

# pip-tools

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




- Oddzielna lista zależności dla ludzi i maszyn
- Narzędzie pozwalające "przypinać" zależności
- pip-compile + pip-sync

# pip-tools demo

# Pipfile

(https://github.com/pypa/pipfile)

- "Requirements.txt" 2.0 - propozycja nowego standardu
- Inicjatywa PyPA (Python Packaging Authority)
- Zastąpienie "requirements.txt" przy pomocy Pipfile oraz Pipfile.lock
- Pipfile - składnia TOML, zawiera podstawowe zależności oraz ich źródło
- Pipfile.lock - zawiera deterministyczny zestaw zależności, pozwalający na powtarzalny *deployment* aplikacji

# Przykładowy Pipfile

```
[[source]]
url = "https://pypi.python.org/simple"
verify_ssl = true

[dev-packages]
pytest = "*"

[packages]
requests = "*"


[requires]
python_version = "3.6"
```

# Przykładowy Pipfile.lock


```
{
    "_meta": {
        "hash": {
            "sha256": "a97c6ee0bb0a72c606f78b9f7a8088b61d838cb6312a26574d2126ad09acfead"
        },
        "host-environment-markers": {
            "implementation_name": "cpython",
            "implementation_version": "3.6.2",
            "os_name": "posix",
            "platform_machine": "x86_64",
            "platform_python_implementation": "CPython",
            "platform_release": "4.4.0-53-generic",
            "platform_system": "Linux",
            "platform_version": "#74-Ubuntu SMP Fri Dec 2 15:59:10 UTC 2016",
            "python_full_version": "3.6.2",
            "python_version": "3.6",
            "sys_platform": "linux"
        },
        "pipfile-spec": 3,
        "requires": {
            "python_version": "3.6"
        },
        "sources": [
            {
                "url": "https://pypi.python.org/simple",
                "verify_ssl": true
            }
        ]
    },
    "default": {
        "certifi": {
            "hashes": [
                "sha256:54a07c09c586b0e4c619f02a5e94e36619da8e2b053e20f594348c0611803704",
                "sha256:40523d2efb60523e113b44602298f0960e900388cf3bb6043f645cf57ea9e3f5"
            ],
            "version": "==2017.7.27.1"
        },
```

# pipenv

(https://github.com/kennethreitz/pipenv)

- Narzędzie rekomendowane przez PyPA
- Połączenie pip, Pipfile oraz virtualenv
- Automatycznie tworzy virtualenva jeśli nie istnieje
- Automatycznie dodaje/usuwa zależności do Pipfile podczas ich instalacji/usuwania
- Tworzenie pliku Pipfile.lock
- Instalacja zależności z Pipfile.lock
- Z defaultu używa hashy do weryfikacji pakietów

# Pipenv demo

# pipsi

(https://github.com/mitsuhiko/pipsi)

- Wrapper na pip i virtualenv
- Służy do instalacji pakietów do oddzielnych środowisk wirtualnych

# pipdeptree

(https://github.com/naiquevin/pipdeptree)

- pip freeze "na sterydach"
- Służy do wyświetlania zainstalowanych pakietów w postaci drzewa zależności

```
ipython==6.1.0
  - decorator [required: Any, installed: 4.1.2]
  - jedi [required: >=0.10, installed: 0.10.2]
  - pexpect [required: Any, installed: 4.2.1]
    - ptyprocess [required: >=0.5, installed: 0.5.2]
  - pickleshare [required: Any, installed: 0.7.4]
  - prompt-toolkit [required: <2.0.0,>=1.0.4, installed: 1.0.15]
    - six [required: >=1.9.0, installed: 1.10.0]
    - wcwidth [required: Any, installed: 0.1.7]
  - pygments [required: Any, installed: 2.2.0]
  - setuptools [required: >=18.5, installed: 28.8.0]
  - simplegeneric [required: >0.8, installed: 0.8.1]
  - traitlets [required: >=4.2, installed: 4.3.2]
    - decorator [required: Any, installed: 4.1.2]
    - ipython-genutils [required: Any, installed: 0.2.0]
    - six [required: Any, installed: 1.10.0]
pipdeptree==0.10.1
  - pip [required: >=6.0.0, installed: 9.0.1]
```

# Dziękuję!

@p_grzesik

pj.grzesik@gmail.com