Managing Python Packages

Concepts, Tooling, History, and Recommandations

09/12/2019

Guilhem Saurel

This presentation

Available at

`homepages.laas.fr/gsaurel/talks/managing-python-packages.pdf`

Under License

🅭🅯🄎

https://creativecommons.org/licenses/by-sa/4.0/

This presentation (continued)

Source

`gitlab.laas.fr/gsaurel/talks : managing-python-packages.md`

Discussions

https://im.laas.fr/#/room/#managing-python-packages:laas.fr

Outline

1 What are Python Packages ?

2 How can they be used ?

3 Which tools can help us ?

What are Python Packages ?

Modules

- `toto.py`
- `toto.so`
- `toto/__init__.py`

Modules

- `toto.py`
- `toto.so`
- `toto/__init__.py`

```
>>> import toto
```

example toto.py or toto/__init__.py

```python
from datetime import datetime
from pathlib import Path


def my_log(line: str, filename: Path):
    with filename.open('a') as handle:
        print(f'{datetime.now()}: {line}', file=handle)


if __name__ == '__main__':
    log_file = Path('/tmp/log.txt')
    my_log("test string", log_file)
```

Packages

```
toto/
├── __init__.py
├── robot.py
├── settings.py
├── simulation.py
└── user_interface.py
```

Packages

```
toto/
├── __init__.py
├── robot.py
├── settings.py
├── simulation.py
└── user_interface.py
```

```
>>> import toto.user_interface
```

Packages

```
toto/
├── __init__.py
├── robot.py
├── sensors/
│   ├── camera.py
│   ├── imu.py
│   └── __init__.py
├── settings.py
├── simulation.py
└── user_interface.py
```

Packages

```
toto/
├── __init__.py
├── robot.py
├── sensors/
│   ├── camera.py
│   ├── imu.py
│   └── __init__.py
├── settings.py
├── simulation.py
└── user_interface.py
```

```
>>> import toto.sensors.imu
```

Imports

don’t

```
from settings import ROBOT_SIZE
```

Imports

don’t

```
from settings import ROBOT_SIZE
```

do

```
from toto.settings import ROBOT_SIZE
# or
from .settings import ROBOT_SIZE
```

Imports

don’t

```python
from settings import ROBOT_SIZE
```

do

```python
from toto.settings import ROBOT_SIZE
# or
from .settings import ROBOT_SIZE

# or
import toto  # or import toto.settings
toto.settings.ROBOT_SIZE
```

Imports (continued)

don’t

```
from .robot import *
```

Imports (continued)

don’t

```
from .robot import *
```

do

```
from .robot import start, stop, turn, jump
# or
import .robot as rb
```

How can they be used ?

sys.path

```
In [1]: import sys

In [2]: sys.path
Out[2]:
['/usr/lib/python38.zip',
 '/usr/lib/python3.8',
 '/usr/lib/python3.8/lib-dynload',
 '',
 '/home/user/.local/lib/python3.8/site-packages',
 '/usr/local/lib/python3.8/dist-packages',
 '/usr/lib/python3/dist-packages']
```

sys.path

```
In [1]: import sys

In [2]: sys.path
Out[2]:
['/usr/lib/python38.zip',
 '/usr/lib/python3.8',
 '/usr/lib/python3.8/lib-dynload',
 '',
 '/home/user/.local/lib/python3.8/site-packages',
 '/usr/local/lib/python3.8/dist-packages',
 '/usr/lib/python3/dist-packages']
```

https://docs.python.org/3/library/site:

```
sys.prefix + 'lib/pythonX.Y/site-packages'
```

Managing paths

```
$ SP="lib/python3.8/site-packages"
$ export PYTHONPATH=/opt/openrobots/${SP}
$ # or
$ echo /opt/openrobots/${SP} > ~/.local/${SP}/my.pth
```

OS package managers

```
$ sudo apt install python3-numpy
$ dpkg -L python3-numpy
…
/usr/lib/python3/dist-packages/numpy/__init__.py
…
```

OS package managers

```
$ sudo apt install python3-numpy
$ dpkg -L python3-numpy
...
/usr/lib/python3/dist-packages/numpy/__init__.py
...
```

- Works out of the box
- Intended package for:
  - your system’s other applications
  - any other user of your system

pip

don’t

```
$ sudo pip install --upgrade numpy
…
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.17.4
    Not uninstalling numpy at /usr/lib/python3/dist-pack
    Can't uninstall 'numpy'. No files were found to unin
Successfully installed numpy-1.21.2
```

```
>>> import numpy
>>> numpy.__version__
'1.21.2'
>>> numpy.__file__
'/usr/local/lib/python3.8/dist-packages/numpy/…'
```

pip

do

```
$ python -m pip install --user --upgrade numpy
…
Installing collected packages: numpy
Successfully installed numpy-1.21.2
```

```
>>> import numpy
>>> numpy.__version__
'1.21.2'
>>> numpy.__file__
'/home/user/.local/lib/python3.8/site-packages/…'
```

Package your own project

With a `setup.py`:

```python
from setuptools import setup

setup(
    name="toto",
    version="1.0.0",
    packages=["toto"],
)
```

pip (continued)

Local path

```
$ python -m pip install --user .
```

pip (continued)

Local path

```
$ python -m pip install --user .
```

Editable path

```
$ python -m pip install --user --editable .
```

pip (continued)

Local path

```
$ python -m pip install --user .
```

Editable path

```
$ python -m pip install --user --editable .
```

VCS URL

```
$ python -m pip install --user \
    'git+https://github.com/psf/black#egg=black'
```

Which tools can help us ?

requirements.txt

```
$ cat requirements.txt
numpy>=1.16
quadprog==0.1.7
requests
tqdm

python -m pip install --user -r requirements.txt
```

pip-tools

```
$ cat requirements.in
django<4
$ python -m pip install --user pip-tools
$ pip-compile
```

pip-tools (continued)

```
$ cat requirements.txt
#
# This file is autogenerated by pip-compile with python
# To update, run:
#
#    pip-compile
#
asgiref==3.4.1
    # via django
django==3.2.9
    # via -r requirements.in
pytz==2021.3
    # via django
sqlparse==0.4.2
    # via django
```

virtualenv

```
$ python -m venv ~/my_v_env
```

virtualenv

```
$ python -m venv ~/my_v_env
```

```
~/my_v_env/
├── bin/
│   ├── activate
│   ├── pip
│   └── python -> /usr/bin/python3
├── include/
├── lib/
│   └── python3.8/
│       └── site-packages/
│           ├── pip/
│           └── setuptools/
└── pyvenv.cfg
```

virtualenv (continued)

```
$ . ~/my_v_env/bin/activate
(my_v_env) $ which python
/home/user/my_v_env/bin/python
```

```
In [1]: import sys

In [2]: sys.path
Out[2]:
['/home/user/my_v_env/bin',
 '/usr/lib/python38.zip',
 '/usr/lib/python3.8',
 '/usr/lib/python3.8/lib-dynload',
 '',
 '/home/user/my_v_env/lib/python3.8/site-packages',
 '/home/user/.ipython']
```

virtualenvwrapper

- `mkvirtualenv`
- `workon`
- `deactivate`
- `.venv` file
- `mktmpenv`

virtualenvwrapper

- `mkvirtualenv`
- `workon`
- `deactivate`
- `.venv` file
- `mktmpenv`

virtualfish

Pipenv

> *Python Development Workflow for Humans*
>
> — Kenneth Reitz

- pip w/ version pinning
- virtualenv w/ wrapper

Poetry

- pip w/ version pinning (like Pipenv)
- virtualenv w/ wrapper (like Pipenv)
- with replacement for `setup.py` / `setup.cfg`
- with better handling of versions
- without Kenneth Reitz

Anaconda

> *The World's Most Popular Data Science Platform*
>
> — https://www.anaconda.com/

- a separate package format
- a separate package repository (condaforge)
- a separate package manager (conda)
- incompatible with everything else you already have

Anaconda

> *The World's Most Popular Data Science Platform*
>
> — https://www.anaconda.com/

- a separate package format
- a separate package repository (condaforge)
- a separate package manager (conda)
- incompatible with everything else you already have
- with commercial support

My recommandations

1. `sudo apt install python3-toto`
2. `python -m pip install --user --upgrade toto`

My recommandations

1. `sudo apt install python3-toto`
2. `python -m pip install --user --upgrade toto`
3. `python -m venv venv`

My recommandations

1. `sudo apt install python3-toto`
2. `python -m pip install --user --upgrade toto`
3. `python -m venv venv`
4. poetry

Questions ?

Thanks for your attention :)