pyproject-devenv

Release 0.1.4

Create virtual environments using pyproject.toml metadata.

Dominic Davis-Foster

May 15, 2024

Contents

Chapter

ONE

Installation

1.1 from PyPI

```
$ python3 -m pip install pyproject-devenv --user
```

1.2 from GitHub

```
$ python3 -m pip install git+https://github.com/repo-helper/pyproject-devenv@master --user
```

Chapter

TWO

Usage

`pyproject-devenv` creates a virtualenv using metadata defined in `pyproject.toml`.

2.1 Configuration

`pyproject-devenv` is configured using the project table in `pyproject.toml`, as defined in **PEP 621**. At a minimum, a value must be provided for name, and dependencies must have a value or be marked as dynamic. A value can also be provided for optional-dependencies, but this cannot be dynamic. Any requirements listed in dependencies or optional-dependencies are installed.

`pyproject-devenv` will also install anything listed in `build-system.requires`, which lists the project's build dependencies. Refer to **PEP 518** for more information on the `build-system` table.

Any requirements listed in *<pyproject_dir>*`/tests/requirements.txt` are also installed if the file exists.

2.2 Command Line Usage

Create virtual environments using pyproject.toml metadata.

```
devenv [OPTIONS] [DEST]
```

Options

-U, --upgrade
Upgrade all specified packages to the newest available version.

-v, --verbose
Show verbose output.

--colour, --no-colour
Whether to use coloured output.

-T, --traceback
Show the complete traceback on error.

--version
Show the version and exit.

Arguments

DEST

The directory to create the virtual environment in.

Optional argument. Default `'venv'`

2.3 Example Configuration

```
[build-system]
# Minimum requirements for the build system to execute.
requires = ["setuptools", "wheel"]  # PEP 508 specifications.

[project]
name = "spam"
version = "2020.0.0"
description = "Lovely Spam! Wonderful Spam!"
dependencies = [
  "httpx",
  "gidgethub[httpx]>4.0.0",
  "django>2.1; os_name != 'nt'",
  "django>2.0; os_name == 'nt'"
]

[project.optional-dependencies]
test = [
  "pytest[testing]<5.0.0",
  "pytest-cov"
]
```

Then run:

```
$ pyproject-devenv
```

```
Successfully created development virtualenv.
```

Output of `pip list`:

```
Package          Version
---------------- ---------
argcomplete      1.12.3
asgiref          3.3.4
atomicwrites     1.4.0
attrs            20.3.0
certifi          2020.12.5
cffi             1.14.5
chardet          4.0.0
coverage         5.5
cryptography     3.4.7
Django           3.2
gidgethub        5.0.1
h11              0.12.0
httpcore         0.12.3
httpx            0.17.1
hypothesis       6.10.0
idna             2.10
```

(continues on next page)

(continued from previous page)

```
more-itertools   8.7.0
nose             1.3.7
packaging        20.9
pip              21.0.1
pluggy           0.13.1
py               1.10.0
pycparser        2.20
PyJWT            2.0.1
pyparsing        2.4.7
pytest           4.6.11
pytest-cov       2.11.1
pytz             2021.1
requests         2.25.1
rfc3986          1.4.0
setuptools       54.2.0
six              1.15.0
sniffio          1.2.0
sortedcontainers 2.3.0
sqlparse         0.4.1
uritemplate      3.0.1
urllib3          1.26.4
wcwidth          0.2.5
wheel            0.36.2
```

Chapter

THREE

Public API

Create virtual environments using `pyproject.toml` metadata.

Exceptions:

BaseInstallError	Base `Exception` to indicate an error occurred when installing packages.
InstallError(*requirements)	`Exception` to indicate an error occurred when installing packages.
InstallFromFileError(filename)	`Exception` to indicate an error occurred when installing packages from a requirements file.

Functions:

mkdevenv(project_dir[, venv_dir, verbosity, ...])	Create a "devenv".

mkdevenv (*project_dir*, *venv_dir='venv'*, ***, *verbosity=1*, *upgrade=False*)

Create a "devenv".

Parameters

- **project_dir** (Union[str, Path, PathLike]) – The root of the project to create the devenv for.
- **venv_dir** (Union[str, Path, PathLike]) – The directory to create the devenv in, relative to `repo_dir`. Default `'venv'`.
- **verbosity** (int) – The verbosity of the function. `0` = quiet, `2` = very verbose. Default `1`.
- **upgrade** (bool) – Whether to upgrade all specified packages to the newest available version. Default `False`.

Return type int

exception BaseInstallError

Bases: `RuntimeError`

Base `Exception` to indicate an error occurred when installing packages.

exception InstallFromFileError (*filename*)

Bases: *pyproject_devenv.BaseInstallError*

`Exception` to indicate an error occurred when installing packages from a requirements file.

Parameters **filename** (Union[str, Path, PathLike]) – The file listing the packages to install.

filename

Type: str

The file listing the packages to install.

exception InstallError(**requirements*)

Bases: *pyproject_devenv.BaseInstallError*

`Exception` to indicate an error occurred when installing packages.

Parameters ***requirements** (`Union`[`str`, `Requirement`]) – The requirements being installed.

requirements

Type: `List`[`str`]

The requirements being installed.

3.5 pyproject_devenv.config

Read the `pyproject-devenv` config from `pyproject.toml`.

Classes:

ConfigDict	`typing.TypedDict` representing the configuration returned by *load_toml()*.
ConfigTracebackHandler([exception])	`consolekit.tracebacks.TracebackHandler` which handles `dom_toml.parser.BadConfigError` and *BaseInstallError*.
PEP621Parser()	Parser for a reduced subset of the **PEP 621** metadata from `pyproject.toml`.

Functions:

load_toml(filename)	Load the `pyproject-devenv` configuration mapping from the given TOML file.

load_toml(*filename*)

Load the `pyproject-devenv` configuration mapping from the given TOML file.

Parameters **filename** (`Union`[`str`, `Path`, `PathLike`])

Return type *ConfigDict*

class ConfigTracebackHandler

Bases: `pyproject_parser.cli.ConfigTracebackHandler`

`consolekit.tracebacks.TracebackHandler` which handles `dom_toml.parser.BadConfigError` and *BaseInstallError*.

typeddict ConfigDict

Bases: `TypedDict`

`typing.TypedDict` representing the configuration returned by *load_toml()*.

Required Keys

- **name** (`str`)
- **dependencies** (`List`[`ComparableRequirement`])
- **optional_dependencies** (`Dict`[`str`, `List`[`ComparableRequirement`]])

- **build_dependencies** (Optional[List[ComparableRequirement]])

class PEP621Parser

Bases: PEP621Parser

Parser for a reduced subset of the **PEP 621** metadata from `pyproject.toml`.

keys = ['name', 'dependencies', 'optional-dependencies']

Type: List[str]

The list of keys parsed from `pyproject.toml`

parse(*config*, *set_defaults=False*)

Parse the TOML configuration.

Parameters

- **config** (Dict[str, Any])
- **set_defaults** (bool) – If `True`, the values in `self.defaults` and `self.factories` will be set as defaults for the returned mapping. Default `False`.

Return type ProjectDict

Chapter

FOUR

Downloading source code

The `pyproject-devenv` source code is available on GitHub, and can be accessed from the following URL: https://github.com/repo-helper/pyproject-devenv

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/repo-helper/pyproject-devenv
```

```
Cloning into 'pyproject-devenv'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a 'zip' file by clicking:
Clone or download –> Download Zip

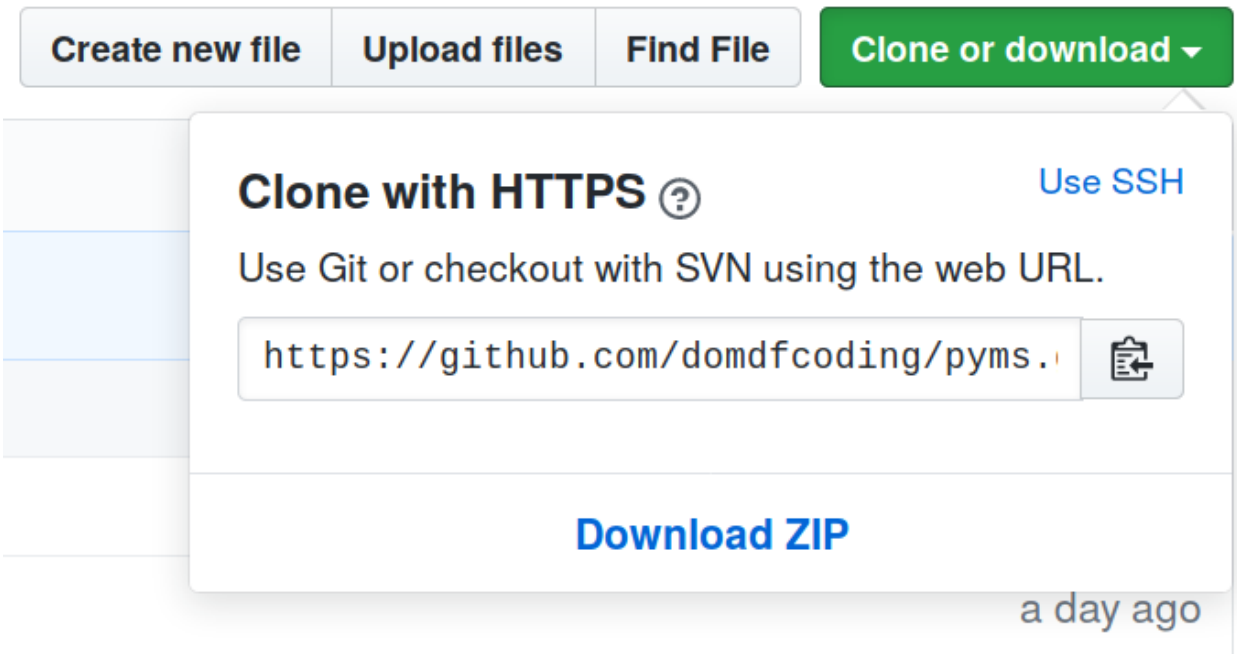

Fig. 1: Downloading a 'zip' file of the source code

4.1 Building from source

The recommended way to build `pyproject-devenv` is to use tox:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use pep517.build or another **PEP 517**-compatible build tool.

Chapter

FIVE

License

`pyproject-devenv` is licensed under the MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Private use – The licensed material may be used and modified in private.

Conditions

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.

Limitations

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

See more information on choosealicense.com ⇒

Python Module Index

p

Index

Symbols

B

C

D

F

I

K

L

M

P

R