
pyproject-devenv

Release 0.1.4

**Create virtual environments using pyproject.toml
metadata.**

Dominic Davis-Foster

May 15, 2024

Contents

1	Installation	1
1.1	from PyPI	1
1.2	from GitHub	1
2	Usage	3
2.1	Configuration	3
2.2	Command Line Usage	3
2.3	Example Configuration	4
3	Public API	7
3.1	mkdevenv	7
3.2	BaseInstallError	7
3.3	InstallFromFileError	7
3.4	InstallError	9
3.5	pyproject_devenv.config	9
4	Downloading source code	11
4.1	Building from source	12
5	License	13
	Python Module Index	15
	Index	17

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
```


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

Public API

Create virtual environments using `pyproject.toml` metadata.

Exceptions:

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

Functions:

<code>mkdevenv(project_dir[, venv_dir, verbosity, ...])</code>	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:

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

Functions:

<code>load_toml(filename)</code>	Load the <code>pyproject-devenv</code> 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`

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

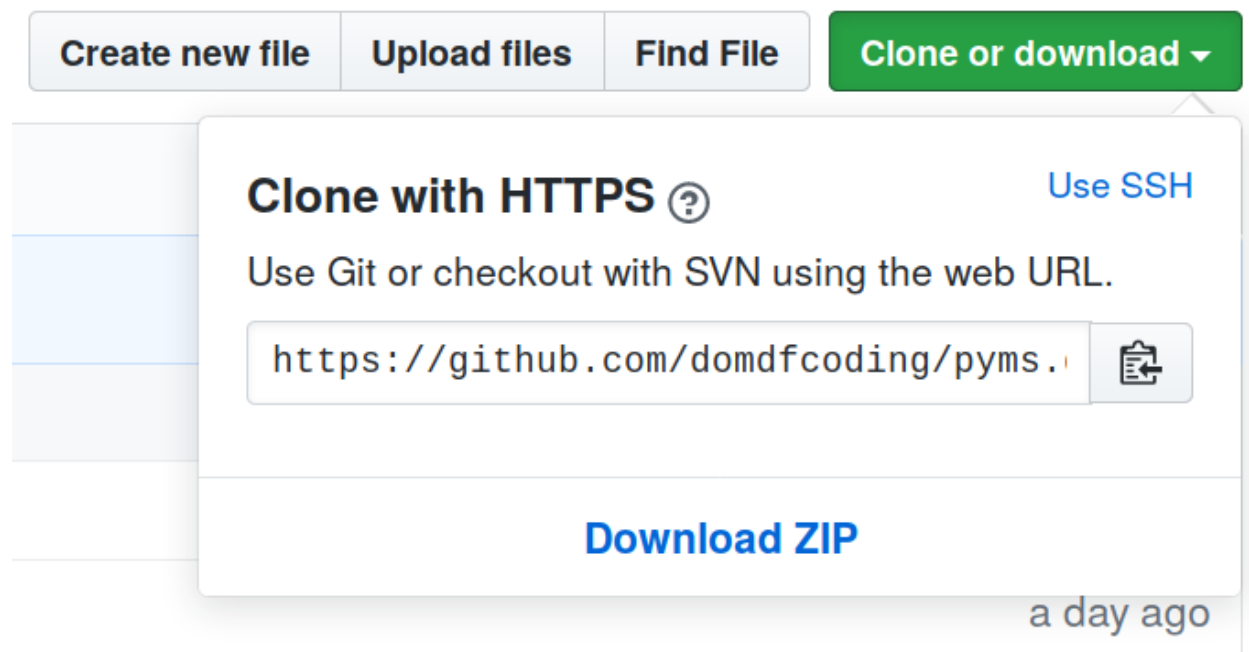


Fig. 1: Downloading a ‘zip’ file of the source code

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.

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](#) ⇒

```
Copyright (c) 2021 Dominic Davis-Foster
```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
OR OTHER DEALINGS IN THE SOFTWARE.
```


Python Module Index

p

`pyproject_devenv`, [7](#)
`pyproject_devenv.config`, [9](#)

Symbols

-T
 devenv command line option, 3
 -U
 devenv command line option, 3
 --colour
 devenv command line option, 3
 --no-colour
 devenv command line option, 3
 --traceback
 devenv command line option, 3
 --upgrade
 devenv command line option, 3
 --verbose
 devenv command line option, 3
 --version
 devenv command line option, 3
 -v
 devenv command line option, 3

B

BaseInstallError, 7

C

ConfigDict (*typeddict in pyproject_devenv.config*), 9
 ConfigTracebackHandler (*class in pyproject_devenv.config*), 9

D

DEST
 devenv command line option, 4
 devenv command line option
 -T, 3
 -U, 3
 --colour, 3
 --no-colour, 3
 --traceback, 3
 --upgrade, 3
 --verbose, 3
 --version, 3
 -v, 3
 DEST, 4

F

filename (*InstallFromFileError attribute*), 7

I

InstallError, 9
 InstallFromFileError, 7

K

keys (*PEP621Parser attribute*), 10

L

load_toml() (*in module pyproject_devenv.config*), 9

M

MIT License, 13
 mkdevenv() (*in module pyproject_devenv*), 7
 module
 pyproject_devenv, 7
 pyproject_devenv.config, 9

P

parse() (*PEP621Parser method*), 10
 PEP621Parser (*class in pyproject_devenv.config*), 10
 pyproject_devenv
 module, 7
 pyproject_devenv.config
 module, 9
 Python Enhancement Proposals
 PEP 517, 12
 PEP 518, 3
 PEP 621, 3, 9, 10
 PEP
 621#dependencies-optional-dependencies,
 3
 PEP 621#dynamic, 3
 PEP 621#name, 3
 PEP 621#table-name, 3

R

requirements (*InstallError attribute*), 9