
handy-archives

Release 0.2.0

Some handy archive helpers for Python.

Dominic Davis-Foster

May 15, 2024

Contents

1	Installation	1
1.1	from PyPI	1
1.2	from Anaconda	1
1.3	from GitHub	1
2	handy_archives	3
2.1	TarFile	3
2.2	ZipFile	4
2.3	is_tarfile	5
2.4	unpack_archive	6
3	handy_archives.testing	7
3.1	ArchiveFileRegressionFixture	7
3.2	archive_regression	8
4	Contributing	9
4.1	Coding style	9
4.2	Automated tests	9
4.3	Type Annotations	9
4.4	Build documentation locally	10
5	Downloading source code	11
5.1	Building from source	12
6	License	13
	Python Module Index	15
	Index	17

Installation

1.1 from PyPI

```
$ python3 -m pip install handy-archives --user
```

1.2 from Anaconda

First add the required channels

```
$ conda config --add channels https://conda.anaconda.org/conda-forge  
$ conda config --add channels https://conda.anaconda.org/domdfcoding
```

Then install

```
$ conda install handy-archives
```

1.3 from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/handy-archives@master --user
```


handy_archives

Some handy archive helpers for Python.

Classes:

<code>TarFile([name, mode, fileobj, format, ...])</code>	Subclass of <code>tarfile.TarFile</code> with additional methods.
<code>ZipFile(file[, mode, compression, ...])</code>	Subclass of <code>zipfile.ZipFile</code> with additional methods.

Functions:

<code>is_tarfile(name)</code>	Return <code>True</code> if <code>name</code> points to a tar archive that <code>tarfile</code> can handle, else return <code>False</code> .
<code>unpack_archive(filename[, extract_dir, format])</code>	Unpack an archive.

class TarFile (*name=None, mode='r', fileobj=None, format=None, tarinfo=None, dereference=None, ignore_zeros=None, encoding=None, errors='surrogateescape', pax_headers=None, debug=None, errorlevel=None, copybufsize=None*)
Subclass of `tarfile.TarFile` with additional methods.

Methods:

<code>extract(member[, path, set_attrs, ...])</code>	Wrapper around <code>tarfile.TarFile.extract()</code> with compatibility shim for PEP 706 on unpatched Pythons.
<code>extractall([path, members, numeric_owner, ...])</code>	Wrapper around <code>tarfile.TarFile.extractall()</code> with compatibility shim for PEP 706 on unpatched Pythons.
<code>extractfile(member)</code>	Extract a member from the archive as a file object.
<code>read_bytes(member)</code>	Returns the content of the given file as bytes.
<code>read_text(member, *, normalize_nl)</code>	Returns the content of the given file as a string.
<code>write_file(filename[, arcname, mtime])</code>	Add the file <code>filename</code> to the archive under the name <code>arcname</code> .

extract (*member, path='', set_attrs=True, *, numeric_owner=False, filter=None*)
Wrapper around `tarfile.TarFile.extract()` with compatibility shim for **PEP 706** on unpatched Pythons.

extractall (*path='.', members=None, *, numeric_owner=False, filter=None*)
Wrapper around `tarfile.TarFile.extractall()` with compatibility shim for **PEP 706** on unpatched Pythons.

extractfile (*member*)

Extract a member from the archive as a file object.

Parameters *member* (`Union[str, TarInfo]`) – A filename or a `tarfile.TarInfo` object.

If *member* is a regular file or a link, an `io.BufferedReader` object is returned. Otherwise `FileNotFoundError` is raised.

Return type `IO[bytes]`

read_bytes (*member*)

Returns the content of the given file as bytes.

Parameters *member* (`Union[str, TarInfo]`)

Raises `FileNotFoundError` – If the file is not found in the archive.

Return type `bytes`

read_text (*member*, *, *normalize_nl=False*)

Returns the content of the given file as a string.

Parameters

- **member** (`Union[str, TarInfo]`)
- **normalize_nl** (`bool`) – If `True`, line endings are normalized to `\n` (LF). Default `False`.

Raises `FileNotFoundError` – If the file is not found in the archive.

Return type `str`

New in version 0.2.0: Added the `normalize_nl` option.

write_file (*filename*, *arcname=None*, *mtime=None*)

Add the file *filename* to the archive under the name *arcname*.

Parameters

- **filename** (`Union[str, Path, PathLike]`)
- **arcname** (`Union[str, Path, PathLike, None]`) – An alternative name for the file in the archive. Default `None`.
- **mtime** (`Optional[datetime]`) – The last modified time of the file. Defaults to the value obtained from `os.stat()`.

class ZipFile (*file*, *mode='r'*, *compression=0*, *allowZip64=True*, *compresslevel=None*, *, *strict_timestamps=True*)

Subclass of `zipfile.ZipFile` with additional methods.

Methods:

<code>extractfile(member[, pwd])</code>	Extract a member from the archive as a file object.
<code>read_bytes(member[, pwd])</code>	Returns the content of the given file as bytes.
<code>read_text(member[, pwd, normalize_nl])</code>	Returns the content of the given file as a string.
<code>write_file(filename[, arcname, mtime])</code>	Put the bytes from <i>filename</i> into the archive under the name <i>arcname</i> .

extractfile (*member*, *pwd=None*)

Extract a member from the archive as a file object.

Parameters

- **member** (`Union[str, ZipInfo]`) – A filename or a `zipfile.ZipInfo` object.
- **pwd** (`Union[str, bytes, None]`) – The password to decrypt files. Default `None`.

Raises `FileNotFoundError` – If the file is not found in the archive.

Return type `IO[bytes]`

read_bytes (*member*, *pwd=None*)

Returns the content of the given file as bytes.

Parameters

- **member** (`Union[str, ZipInfo]`)
- **pwd** (`Union[str, bytes, None]`) – The password to decrypt files. Default `None`.

Raises `FileNotFoundError` – If the file is not found in the archive.

Return type `bytes`

read_text (*member*, *pwd=None*, *, *normalize_nl=False*)

Returns the content of the given file as a string.

Parameters

- **member** (`Union[str, ZipInfo]`)
- **pwd** (`Union[str, bytes, None]`) – The password to decrypt files. Default `None`.
- **normalize_nl** (`bool`) – If `True`, line endings are normalized to `\n` (LF). Default `False`.

Raises `FileNotFoundError` – If the file is not found in the archive.

Return type `str`

New in version 0.2.0: Added the `normalize_nl` option.

write_file (*filename*, *arcname=None*, *mtime=None*)

Put the bytes from `filename` into the archive under the name `arcname`.

Parameters

- **filename** (`Union[str, Path, PathLike]`)
- **arcname** (`Union[str, Path, PathLike, None]`) – An alternative name for the file in the archive. Default `None`.
- **mtime** (`Optional[datetime]`) – The last modified time of the file. Defaults to the value obtained from `os.stat()`.

is_tarfile (*name*)

Return `True` if `name` points to a tar archive that `tarfile` can handle, else return `False`.

Parameters **name** (`Union[str, Path, PathLike, IO[bytes]]`) – A string, file, or file-like object.

Return type `bool`

unpack_archive (*filename*, *extract_dir*=None, *format*=None)

Unpack an archive.

Parameters

- **filename** (`Union[str, Path, PathLike]`) – The name of the archive.
- **extract_dir** (`Union[str, Path, PathLike, None]`) – The name of the target directory, where the archive is unpacked. If not provided, the current working directory is used. Default `None`.
- **format** (`Optional[str]`) – The archive format: one of 'zip', 'tar', 'gztar', 'bztar', or 'xztar', or any other format registered through `shutil.register_unpack_format()`. If not provided, `unpack_archive` will use the filename extension and see if an unpacker was registered for that extension. Default `None`.

If no unpacker is found, a `ValueError` is raised.

handy_archives.testing

Pytest helpers.

Attention: This module has the following additional requirements:

```
coincidence>=0.2.0
pytest>=6.0.0
```

These can be installed as follows:

```
$ python -m pip install handy-archives[testing]
```

Classes:

<code>ArchiveFileRegressionFixture(datadir, ...)</code>	Class for performing regression checks on files in <code>tar</code> and <code>zip</code> archives.
---	--

Functions:

<code>archive_regression(datadir, ...)</code>	Pytest fixture for performing regression tests on files in <code>tar</code> and <code>zip</code> archives.
---	--

class `ArchiveFileRegressionFixture` (*datadir, original_datadir, request*)

Bases: `AdvancedFileRegressionFixture`

Class for performing regression checks on files in `tar` and `zip` archives.

Methods:

<code>check_archive</code> (<i>archive, filename, **kwargs</i>)	Checks a text file in <i>archive</i> against a previously recorded version, or generates a new file.
<code>check_archive_binary</code> (<i>archive, filename, **kwargs</i>)	Checks a binary file in <i>archive</i> against a previously recorded version, or generates a new file.

check_archive (*archive, filename, **kwargs*)

Checks a text file in *archive* against a previously recorded version, or generates a new file.

Parameters

- **archive** (`Union[TarFile, ZipFile]`) – The `tar` archive.
- **filename** (`str`) – The name of the file in the archive to check.
- ****kwargs** – Additional keyword arguments passed to `pytest_regressions.file_regression.FileRegressionFixture.check()`.

check_archive_binary (*archive*, *filename*, ***kwargs*)

Checks a binary file in *archive* against a previously recorded version, or generates a new file.

Parameters

- **archive** (`Union[TarFile, ZipFile]`)
- **filename** (`str`) – The name of the file in the archive to check.
- ****kwargs** – Additional keyword arguments passed to `pytest_regressions.file_regression.FileRegressionFixture.check()`.

archive_regression (*datadir*, *original_datadir*, *request*)

Pytest fixture for performing regression tests on files in `tar` and `zip` archives.

Return type `AdvancedFileRegressionFixture`

Contributing

handy-archives uses `tox` to automate testing and packaging, and `pre-commit` to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```

4.1 Coding style

`formate` is used for code formatting.

It can be run manually via `pre-commit`:

```
$ pre-commit run formate -a
```

Or, to run the complete autoformatting suite:

```
$ pre-commit run -a
```

4.2 Automated tests

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```

4.3 Type Annotations

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```

4.4 Build documentation locally

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```

Downloading source code

The handy-archives source code is available on GitHub, and can be accessed from the following URL: <https://github.com/domdfcoding/handy-archives>

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

```
$ git clone https://github.com/domdfcoding/handy-archives
```

```
Cloning into 'handy-archives'...
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

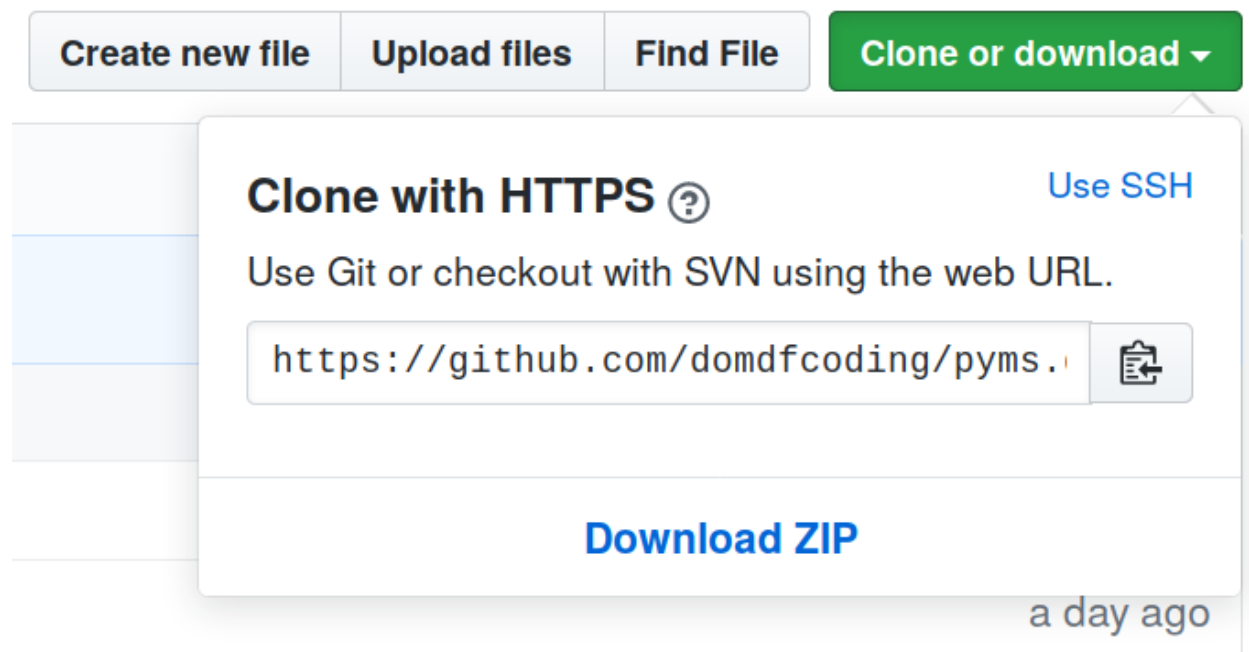


Fig. 1: Downloading a ‘zip’ file of the source code

5.1 Building from source

The recommended way to build `handy-archives` 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

handy-archives 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

h

`handy_archives`, [3](#)
`handy_archives.testing`, [7](#)

A

`archive_regression()` (in module `handy_archives.testing`), 8
`ArchiveFileRegressionFixture` (class in `handy_archives.testing`), 7

C

`check_archive()` (`ArchiveFileRegressionFixture` method), 7
`check_archive_binary()` (`ArchiveFileRegressionFixture` method), 7

E

`extract()` (`TarFile` method), 3
`extractall()` (`TarFile` method), 3
`extractfile()` (`TarFile` method), 3
`extractfile()` (`ZipFile` method), 4

H

`handy_archives`
 module, 3
`handy_archives.testing`
 module, 7

I

`is_tarfile()` (in module `handy_archives`), 5

M

MIT License, 13
 module
 `handy_archives`, 3
 `handy_archives.testing`, 7

P

Python Enhancement Proposals
 PEP 517, 12
 PEP 706, 3

R

`read_bytes()` (`TarFile` method), 4
`read_bytes()` (`ZipFile` method), 5
`read_text()` (`TarFile` method), 4
`read_text()` (`ZipFile` method), 5

T

`TarFile` (class in `handy_archives`), 3

U

`unpack_archive()` (in module `handy_archives`), 5

W

`write_file()` (`TarFile` method), 4
`write_file()` (`ZipFile` method), 5

Z

`ZipFile` (class in `handy_archives`), 4