verbose-version-info Documentation

Release 0.0.1

Sebastian Weigand

Aug 27, 2023

CONTENTS:

CHAPTER

ONE

VERBOSE-VERSION-INFO

Generate verbose version information for python packages

- Free software: Apache Software License 2.0
- Documentation: https://verbose-version-info.readthedocs.io.

1.1 Features

Implemented

- Basic version retrieval
- Customizable string for not found version
- Commit_id for `pip install git+<url>`
- Split off cli to an extra
- Detect `pip install -e` installation and get path
- commit id for `pip install -e .` if `.git` exists
- commit id for `pip install .` if `.git` exists
- Determine dist time for `pip install .` (needed for better commit_id)
- get commit id for `pip install .` if `.git` exists, for the closest commit at installation time
- use find_url_info in vv_info for tarball installation
- Add dist_mtime time to VerboseVersionInfo
- Add warning if repo of source install is dirty (`git status -s != ""`)

TODO

- Use a Singleton class instead of dicts for settings
- Reset settings function (mostly notebook showoff)
- setting formatter: `Mapping[str, format_function]` (used for sha)
- extract minimal required versions (useful for CI tests, of the min version)
- export minimal requirements to file (pip or conda style)

- add conda support
- create github markdown summary

1.2 Contributors

Thanks goes to these wonderful people (emoji key):

This project follows the all-contributors specification. Contributions of any kind welcome!

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install verbose-version-info, run this command in your terminal:

```
$ pip install verbose_version_info
```

This is the preferred method to install verbose-version-info, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for verbose-version-info can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/s-weigand/verbose-version-info
```

Or download the tarball:

```
$ curl -OJL https://github.com/s-weigand/verbose-version-info/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use verbose-version-info in a project:

```
import verbose_version_info
```

CHAPTER

FOUR

INNER WORKINGS

This is the detailed documentation of the inner workings of `verbose_version_info`.

verbose_version_info	Top-level package for verbose-version-info.

4.1 verbose_version_info

Top-level package for verbose-version-info.

Modules

verbose_version_info.data_containers	Module for data container classes.
verbose_version_info.resource_finders	Module containing function to look up resources.
verbose_version_info.settings	Module containing all settings related functionalities.
verbose_version_info.utils	Utility modules with convenience functions.
verbose_version_info.vcs	Module containing code for version control system retrieval.
verbose_version_info.verbose_version_info	Main module.

4.1.1 data_containers

Module for data container classes.

Classes

Summary

VcsInfo	Container for vcs information.
VerboseVersionInfo	Information container for verbose version information.

VcsInfo

class **VcsInfo**(*vcs_name: str*, *commit_id: str*)

Container for vcs information.

Create new instance of VcsInfo(vcs_name, commit_id)

Attributes Summary

commit_id	Alias for field number 1
vcs_name	Alias for field number 0

Methods Summary

count	Return number of occurrences of value.
index	Return first index of value.

count

VcsInfo.**count**(*value*, /)

Return number of occurrences of value.

index

VcsInfo.**index**(*value*, *start=0*, *stop=9223372036854775807*, /)

Return first index of value.

Raises ValueError if the value is not present.

Methods Documentation

count(*value*, /)

Return number of occurrences of value.

index(*value*, *start=0*, *stop=9223372036854775807*, /)

Return first index of value.

Raises ValueError if the value is not present.

VerboseVersionInfo

class VerboseVersionInfo(*release_version: str*, *dist_time: datetime*, *url: str = ''*, *commit_id: str = ''*, *vcs_name: str = ''*)

Information container for verbose version information.

Create new instance of VerboseVersionInfo(release_version, dist_time, url, commit_id, vcs_name)

Attributes Summary

commit_id	Alias for field number 3
dist_time	Alias for field number 1
release_version	Alias for field number 0
url	Alias for field number 2
vcs_name	Alias for field number 4

Methods Summary

count	Return number of occurrences of value.
index	Return first index of value.

count

VerboseVersionInfo.**count**(*value*, /)

Return number of occurrences of value.

index

VerboseVersionInfo.**index**(*value*, *start=0*, *stop=9223372036854775807*, /)

Return first index of value.

Raises ValueError if the value is not present.

Methods Documentation

count(*value*, /)

Return number of occurrences of value.

index(*value*, *start=0*, *stop=9223372036854775807*, /)

Return first index of value.

Raises ValueError if the value is not present.

4.1.2 resource_finders

Module containing function to look up resources.

Functions

Summary

dist_info_mtime	Modification time of the dist info, current time if editable installed.
egg_link_lines	Lines of an `.egg-link` file if it exists.
file_uri_to_path	Convert file uri to a path if the path exists.
find_editable_install_basepath	Find basepath of an as editable installed package.
find_url_info	Extract package information for packages installed from an url or locally.
local_install_basepath	Extract base installation path for packages installed from local resource.

dist_info_mtime

dist_info_mtime(*distribution_name: str*) → datetime

Modification time of the dist info, current time if editable installed.

This should basically be the same as the installation time for packages installed from source in a none editable mode.

Parameters
distribution_name (*str*) – The name of the distribution package as a string.

Returns
Time the dist-info was packaged or current time if not found.

Return type
datetime

egg_link_lines

egg_link_lines(*distribution_name: str*) → List[str] | None

Lines of an `.egg-link` file if it exists.

This assumes that a file with `<distribution_name>.egg-link` exists somewhere in the path (which is at least for pip the case).

Parameters
distribution_name (*str*) – The name of the distribution package as a string.

Returns
Lines read from `<distribution_name>.egg-link` with striped newline.

Return type
Optional[List[str]]

See also:

find_editable_install_basepath

file_uri_to_path

file_uri_to_path(*uri: str*) → Path | None

Convert file uri to a path if the path exists.

Used to get the base path of local installations from source, e.g. `pip install .`.

Parameters
uri (*str*) – Uri to a file e.g. 'file:///tmp/dist'

Returns
Path of the file if it exists

Return type
Path

find_editable_install_basepath

find_editable_install_basepath(*distribution_name: str*) → Path | None

Find basepath of an as editable installed package.

Parameters
distribution_name (*str*) – The name of the distribution package as a string.

Returns
Path to the root of an as editable installed package. E.g. `pip install -e .`

Return type
Optional[Path]

See also:

egg_link_lines

find_url_info

find_url_info(*distribution_name: str*, *dist_time: datetime | None = None*) → *VerboseVersionInfo* | None

Extract package information for packages installed from an url or locally.

If the packages was installed using an url 'direct_url.json' will be parsed and the information extracted. If a `vcs` (e.g. `git`) was used, the used `vcs` and `commit_id` will be retrieved as well.

Parameters

- **distribution_name** (*str*) – The name of the distribution package as a string.
- **dist_time** (*datetime*) – Datetime instance of when the distribution was created.

Examples

If the package was installed using git: `pip install git+https://github.com/s-weigand/git-install-test-distribution.git`

```
>>> find_url_info("git-install-test-distribution", datetime(2021, 2, 28))
VcsInfo(
    release_version="0.0.2",
    dist_time=datetime(2021, 2, 28),
    url="https://github.com/s-weigand/git-install-test-distribution.git",
    commit_id="a7f7bf28dbe9bfceba1af8a259383e398a942ad0",
    vcs="git",
)
```

If the package was installed by an url to an archive on '2021-02-27': `pip install https://github.com/s-weigand/git-install-test-distribution/archive/main.zip`

```
>>> find_url_info("git-install-test-distribution")
VcsInfo(
    release_version="0.0.2",
    dist_time=datetime(2021, 2, 27),
    url="https://github.com/s-weigand/git-install-test-distribution/
↪archive/main.zip",
    commit_id="",
    vcs="",
)
```

If the package was not installed from an url or locally: `pip install package-on-pypi`

```
>>> find_url_info("package-on-pypi")
None
```

Returns

VerboseVersionInfo
If the package was installed from a url resource.

None
If the package was installed from as editable or PyPi.

Return type
VerboseVersionInfo | None

local_install_basepath

local_install_basepath(*distribution_name: str*, ***, *vv_info:* VerboseVersionInfo | *None = None*) → Path | None

Extract base installation path for packages installed from local resource.

Parameters

- **distribution_name** (*str*) – The name of the distribution package as a string.
- **vv_info** (*Optional[VerboseVersionInfo]*) – Verbose version info generated by *find_url_info()*.

Returns
Path to the root of a package which was installed from a local resource.

Return type
Optional[Path]

See also:

find_url_info, *file_uri_to_path*, *find_editable_install_basepath*

4.1.3 settings

Module containing all settings related functionalities.

4.1.4 utils

Utility modules with convenience functions.

Functions

Summary

dist_files	List of PackagePaths even if the package is broken.
distribution	Get the `Distribution` instance for the named package.

dist_files

dist_files(*distribution_name: str*) → List[PackagePath]

List of PackagePaths even if the package is broken.

This is a convenience function since Distribution.files could be None. I.e. if RECORD for dist-info or SOURCES.txt for egg-info.

See: importlib.metadata.Distribution.files

Parameters
distribution_name (*str*) – The name of the package as a string.

Returns
Paths of files used by the package.

Return type
List[PackagePath]

distribution

distribution(*distribution_name: str*) → Distribution

Get the `Distribution` instance for the named package.

Parameters
distribution_name (*str*) – The name of the package as a string.

Returns
Distribution instance of the package

Return type
Distribution

Classes

Summary

NotFoundDistribution	Distribution of package which couldn't be found.

NotFoundDistribution

class NotFoundDistribution

Distribution of package which couldn't be found.

This class is used not to repeat the try-except pattern over and over again. If a package isn't found an instance of this class is returned by `get_distribution()` and the parts of the API which are used by `verbose_version_info` are kept intact for the way they are used.

See also:

`get_distribution`

Attributes Summary

`entry_points`	
`files`	List of files in this distribution.
`metadata`	Return the parsed metadata for this Distribution.
`name`	Return the 'Name' metadata for the distribution package.
`requires`	List of generated requirements specified for this Distribution.
`version`	User set string if no version was found.

Methods Summary

at	Return a Distribution for the indicated metadata path
discover	Return an iterable of Distribution objects for all packages.
from_name	Return the Distribution for the given package name.
locate_file	Given a path to a file in this distribution, return a path to it.
read_text	Attempt to load metadata file given by the name.

at

static NotFoundDistribution.**at**(*path*)

Return a Distribution for the indicated metadata path

Parameters
path – a string or path-like object

Returns
a concrete Distribution instance for the path

discover

classmethod NotFoundDistribution.**discover**(***kwargs*)

Return an iterable of Distribution objects for all packages.

Pass a `context` or pass keyword arguments for constructing a context.

Context
A `DistributionFinder.Context` object.

Returns
Iterable of Distribution objects for all packages.

from_name

classmethod NotFoundDistribution.**from_name**(*name*)

Return the Distribution for the given package name.

Parameters
name – The name of the distribution package to search for.

Returns
The Distribution instance (or subclass thereof) for the named package, if found.

Raises
PackageNotFoundError – When the named package's distribution metadata cannot be found.

locate_file

NotFoundDistribution.**locate_file**(*path: PathLike | str*) → PathLike

Given a path to a file in this distribution, return a path to it.

Just added to satisfy mypy, since the superclass one is an abstractmethod.

Parameters
path (*PathLike | str*) – Path to a file.

Returns
Just the initial path ensure to be type PathLike.

Return type
PathLike

read_text

NotFoundDistribution.**read_text**(*filename: str*) → None

Attempt to load metadata file given by the name.

Just added to satisfy mypy, since the superclass one is an abstractmethod.

Parameters
filename (*str*) – Name of a file in the distribution

Return type
None

Methods Documentation

static at(*path*)

Return a Distribution for the indicated metadata path

Parameters
path – a string or path-like object

Returns
a concrete Distribution instance for the path

classmethod discover(***kwargs*)

Return an iterable of Distribution objects for all packages.

Pass a context or pass keyword arguments for constructing a context.

Context
A DistributionFinder.Context object.

Returns
Iterable of Distribution objects for all packages.

classmethod from_name(*name*)

Return the Distribution for the given package name.

Parameters
name – The name of the distribution package to search for.

Returns
The Distribution instance (or subclass thereof) for the named package, if found.

Raises
PackageNotFoundError – When the named package's distribution metadata cannot be found.

locate_file(*path: PathLike | str*) → PathLike

Given a path to a file in this distribution, return a path to it.

Just added to satisfy mypy, since the superclass one is an `abstractmethod`.

Parameters
path (*PathLike | str*) – Path to a file.

Returns
Just the initial `path` ensure to be type `PathLike`.

Return type
PathLike

read_text(*filename: str*) → None

Attempt to load metadata file given by the name.

Just added to satisfy mypy, since the superclass one is an `abstractmethod`.

Parameters
filename (*str*) – Name of a file in the distribution

Return type
None

4.1.5 vcs

Module containing code for version control system retrieval.

Functions

Summary

add_vcs_commit_id_reader	Add vcs commit_id reader function to the list of registered function.
local_git_commit_id	Get git commit_id of locally installed package.
run_vcs_commit_id_command	Inner function of commit_id retrieval functions.

add_vcs_commit_id_reader

add_vcs_commit_id_reader(*func: Callable[[Path, datetime], VcsInfo | None]*) → Callable[[Path, datetime], *VcsInfo* | None]

Add vcs commit_id reader function to the list of registered function.

This is pretty much the most simple decorator possible, there isn't any sanity checking (e.g. functools function signature) since this package doesn't have a pluginsystem and the sanity check is done by mypy.

Parameters
func (*VcsCommitIdReader*) – Function to be added

Returns
Originally added function.

Return type
VcsCommitIdReader

local_git_commit_id

local_git_commit_id(*local_install_basepath: Path*, *dist_mtime: datetime*) → *VcsInfo* | None

Get git commit_id of locally installed package.

Parameters

- **local_install_basepath** (*Path*) – Basepath of the local installation.
- **dist_mtime** (*datetime*) – Time the packaged distribution was modified. This is only important for none editable installations from source.

Returns
(vcs_name, commit_id)

Return type
Optional[*VcsInfo*]

See also:

run_vcs_commit_id_command, *verbose_version_info.resource_finders.dist_info_mtime*

run_vcs_commit_id_command

run_vcs_commit_id_command(*, *vcs_name: str*, *commit_id_command: List[str] | Tuple[str, ...]*, *local_install_basepath: Path*, *need_to_exist_path_child: str = '.'*, *check_dirty_command: List[str] | Tuple[str, ...] | None = None*) → *VcsInfo* | None

Inner function of commit_id retrieval functions.

Parameters

- **vcs_name** (*str*) – Name if the vcs, which will be used as part of the result.
- **commit_id_command** (*Union[List[str], Tuple[str, ...]]*) – Shell command to return the commit_id. E.g. for `git`: `("git", "log", "--before", f"'{date_string}'", "-n", "1", "--pretty=format:%H",)`
- **local_install_basepath** (*Path*) – Basepath of the local installation.
- **need_to_exist_path_child** (*str*) – Childitem that needs to exists inside of local_install_basepath. E.g. for `git`: `".git"`. by default “.”
- **check_dirty_command** (*Optional[Union[List[str], Tuple[str, ...]]]*) – Command to be run for checking if a directory contains uncommitted changes. E.g. for `git`: ``(“git”, “status”, “-s”)``by default None

Returns
(vcs_name, commit_id)

Return type
Optional[*VcsInfo*]

See also:

`get_local_git_commit_id`

Exceptions

Exception Summary

UncommittedChangesWarning	Warning thrown if a director under source control has uncommitted changes.

UncommittedChangesWarning

exception UncommittedChangesWarning

Warning thrown if a director under source control has uncommitted changes.

4.1.6 verbose_version_info

Main module.

Functions

Summary

release_version	Retrieve the release version of a distribution.
vv_info	Verbose version information of an installed package.

release_version

release_version(*distribution_name: str*) → str

Retrieve the release version of a distribution.

Parameters
distribution_name (*str*) – The name of the distribution package as a string.

Returns
Version string of the distribution

Return type
str

vv_info

vv_info(*distribution_name: str*) → *VerboseVersionInfo*

Verbose version information of an installed package.

Known limitations:

- Does not include uncommitted changes.

- **Can't determine vcs information for tarball installations.**
  E.g. `pip install https://github.com/s-weigand/git-install-test-distribution/archive/main.zip`

Parameters
distribution_name (*str*) – The name of the distribution package as a string.

Returns
Verbose version information of the installed package, as detailed as possible.

Return type
VerboseVersionInfo

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/s-weigand/verbose-version-info/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

verbose-version-info could always use more documentation, whether as part of the official verbose-version-info docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/s-weigand/verbose-version-info/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up `verbose_version_info` for local development.

1. Fork the `verbose-version-info` repo on GitHub.
2. Clone your fork locally:

```
$ git clone --recursive git@github.com:your_name_here/verbose_version_info.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv verbose_version_info
$ cd verbose-version-info/
$ pip install -e .
```

4. install the `pre-commit` and `pre-push` hooks:

```
$ pre-commit install && pre-commit install -t pre-push
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.md.
3. The pull request should work for Python 3.8, 3.9, 3.10 and 3.11, and for PyPy. Check https://github.com/s-weigand/verbose-version-info/actions and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_verbose_version_info
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push --follow-tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

HISTORY

6.1 0.0.1 (2021-02-18)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

V

INDEX