
semvermanager Documentation

Release 0.0.2-alpha11

Joe Drumgoole

Feb 07, 2019

Contents:

1	Indices and tables	3
	Python Module Index	5

semvermanager exports a single class `Version` which implements a restricted subset of the [SEMVER](#) standard.

`Version` defines a Semantic version using the following field structure:

```
# MAJOR.MINOR.PATCH-TAG

int MAJOR   # 0 to N
int MINOR   # 0 to N
int PATCH   # 0 to N
str TAG     # one of "alpha", "beta".
int TAG_VERSION # appended to the tag if the tag is alpha or beta.
```

Versions may be bumped by a single increment using any of the *bump* functions. Bumping a `PATCH` value simply increments it. Bumping a `MINOR` value zeros the `PATCH` value and bumping a `MAJOR` zeros the `MINOR` and the `PATCH` value.

semvermanager only supports Python 3.6 and greater.

```
class semvermanager.Version(major=0, minor=0, patch=0, tag='alpha', tag_version=0,
                             lhs='VERSION', separator='=')
```

Handle creation and storage of SEMVER version numbers. In this case SEMVERs must be of the form a.b.c-tag, Where a,b and c are integers in the range 0-n and tag is one of *Version.TAGS*.

Version numbers may be bumped by using the various bump functions. Bumping minor zeros patch, bumping major zeros minor.

```
bump_map()
```

a mapping of field names to corresponding bump methods :return: a dict of field names to bump methods

```
field(field)
```

Return the mapping from a field to its corresponding property. :param field: str in *Version.FIELDS* :return:

```
field_map()
```

Mapping of field names to field values. :return: A dict of field names to their properties.

```
static find(filename, lhs='VERSION', separator='=')
```

Look for the first instance of a `VERSION` definition in a file and try and parse it as a *Version*

```
read(filename, lhs=None, separator=None)
```

Read a single line from filename and parse it as version string.

Parameters filename – a file containing a single line `VERSION` string.

:param lhs : override the class field string :param separator: the character separating the `VERSION` label from the value :return: a *Version* object

:raises `VersionError` if it fails to parse the file.

```
static update(filename, version, lhs='VERSION', separator='=')
```

Find any line starting with “`VERSION`” and replace that line with the new *version*.

Parameters

- **filename** – A path to a file containing at least one `VERSION` line
- **version** – The new version object
- **lhs** – The label string
- **separator** – label<separator>value

Returns A tuple (number of lines updated, list(line_numbers))

```
write(filename)
```

Write a single line containing the version object to filename. This will overwrite the existing file if it exists.

Parameters **filename** – The file to create with the new version object

Returns A tuple of the filename and the version object

exception `semvermanager.VersionError`

Exception for handling errors in Version Class

CHAPTER 1

Indices and tables

- `genindex`
- `modindex`
- `search`

S

`semvermanager`, [1](#)

B

`bump_map()` (*semvermanager.Version method*), 1

F

`field()` (*semvermanager.Version method*), 1

`field_map()` (*semvermanager.Version method*), 1

`find()` (*semvermanager.Version static method*), 1

R

`read()` (*semvermanager.Version method*), 1

S

`semvermanager` (*module*), 1

U

`update()` (*semvermanager.Version static method*), 1

V

`Version` (*class in semvermanager*), 1

`VersionError`, 2

W

`write()` (*semvermanager.Version method*), 1