polyline

Release 2.0.2

Frederick Jansen

Feb 12, 2024

CONTENTS

1 Installation **3**

2 API Documentation **5**
- 2.1 Encoding 5
- 2.2 Decoding 5

3 Development **7**
- 3.1 Documentation 7
- 3.2 Testing and Conventions 7
- 3.3 Contributions 7
- 3.4 Versioning 8
- 3.5 Publishing 8

`polyline` is a Python implementation of Google's Encoded Polyline Algorithm Format. It is essentially a port of Mapbox polyline with some additional features.

CHAPTER

ONE

INSTALLATION

`polyline` can be installed using `pip`:

```
$ pip install polyline
```

Starting from `v2.0.0` only Python 3.7 and above is supported. For Python 2 support, please install `v1.4.0`:

```
$ pip install polyline==1.4.0
```

CHAPTER

TWO

API DOCUMENTATION

2.1 Encoding

To get the encoded polyline representation of a given set of (lat, lon) coordinates:

```
import polyline
polyline.encode([(38.5, -120.2), (40.7, -120.9), (43.2, -126.4)], 5)
```

This should return `_p~iF~ps|U_ulL~ugC_hgN~eq`@`.

You can set the required precision with the optional `precision` parameter. The default value is 5.

You can encode (lon, lat) tuples by setting `geojson=True`.

2.2 Decoding

To get a set of coordinates represented by a given encoded polyline string:

```
import polyline
polyline.decode('u{~vFvyys@fS]', 5)
```

This should return `[(40.63179, -8.65708), (40.62855, -8.65693)]` in (lat, lon) order.

You can set the required precision with the optional `precision` parameter. The default value is 5.

You can decode into (lon, lat) tuples by setting `geojson=True`.

CHAPTER

THREE

DEVELOPMENT

All installation and development dependencies are fully specified in `pyproject.toml`. The `project.optional-dependencies` object is used to specify optional requirements for various development tasks. This makes it possible to specify additional options when performing installation using `pip`:

```
python -m pip install .[dev]
```

3.1 Documentation

The documentation can be generated automatically from the source files using Sphinx:

```
python -m sphinx.cmd.build -b html docs docs/_build/html
```

3.2 Testing and Conventions

All unit tests are executed and their coverage is measured when using pytest:

```
python -m pytest
```

Style conventions are enforced using Pylint:

```
python -m pylint polyline
```

3.3 Contributions

In order to contribute to the source code, open an issue or submit a pull request on the GitHub page for this library.

3.4 Versioning

Beginning with version 0.1.0, the version number format for this library and the changes to the library associated with version number increments conform with Semantic Versioning 2.0.0.

3.5 Publishing

This library can be published as a package on PyPI by a package maintainer. Ensure that the correct version number appears in `pyproject.toml`, and that any links in this README document to the Read the Docs documentation of this package (or its dependencies) have appropriate version numbers. Also ensure that the Read the Docs project for this library has an automation rule that activates and sets as the default all tagged versions. Create and push a tag for this version (replacing `?.?.?` with the version number):

```
git tag ?.?.?
git push origin ?.?.?
```

Remove any old build/distribution files. Then, package the source into a distribution archive:

```
rm -rf build dist src/*.egg-info
python -m build --sdist --wheel .
```

Finally, upload the package distribution archive to PyPI:

```
python -m twine upload dist/*
```