# standardjson Documentation

*Release 0.3.1*

**Audrey Roy**

**Sep 27, 2017**

# Contents

Contents:

# standardjson

JSON encoder fully compliant with the ECMA-262 and ECMA-404 specifications.

- Free software: BSD license
- Documentation: http://standardjson.readthedocs.org.

## Features

Support for all objects that the Python stdlib's *json.JSONEncoder* can encode, plus:

- *datetime.datetime*
- *datetime.date*
- *datetime.time*
- *decimal.Decimal*

Works on Python 2.6, 2.7, 3.3. Probably works on 3.4 and 3.5 but I haven't set up tests for those with Tox yet.

## Quickstart

Use *StandardJSONEncoder* as you would use *json.JSONEncoder* from the Python standard library:

```python
>>> import datetime
>>> import json
>>> from standardjson import StandardJSONEncoder

>>> json.dumps({'day': datetime.date(2010, 2, 17)}, cls=StandardJSONEncoder)
'{"day": "2010-02-17"}'
```

# Installation

At the command line:

```
$ easy_install standardjson
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv standardjson
$ pip install standardjson
```

# Usage

Use *StandardJSONEncoder* as you would use *json.JSONEncoder* from the Python standard library:

```
>>> import datetime
>>> import json
>>> from standardjson import StandardJSONEncoder

>>> json.dumps({'day': datetime.date(2010, 2, 17)}, cls=StandardJSONEncoder)
'{"day": "2010-02-17"}'
```

You can encode a single Python data structure too:

```
>>> StandardJSONEncoder().encode({'day': datetime.date(2010, 2, 17)})
'{"day": "2010-02-17"}'
```

# 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:

## Types of Contributions

### Report Bugs

Report bugs at https://github.com/audreyr/standardjson/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.

### Fix Bugs

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

### Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

## Write Documentation

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

## Submit Feedback

The best way to send feedback is to file an issue at https://github.com/audreyr/standardjson/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 :)

# Get Started!

Ready to contribute? Here's how to set up *standardjson* for local development.

1. Fork the *standardjson* repo on GitHub.

2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/standardjson.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 standardjson
$ cd standardjson/
$ python setup.py develop
```

4. Create a branch for local development:

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

Now you can make your changes locally.

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

```
$ flake8 standardjson tests
$ python setup.py test
$ tox
```

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

6. 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
```

7. Submit a pull request through the GitHub website.

## 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.rst.

3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/audreyr/standardjson/pull_requests and make sure that the tests pass for all supported Python versions.

## Tips

To run a subset of tests:

```
$ python -m unittest tests.test_standardjson
```

Credits

## Development Lead

- Audrey Roy (@audreyr)

## Contributors

- Daniel Greenfeld (@pydanny)

CHAPTER 6

---

History

---

# CHAPTER 7

## 0.3.1 (2014-05-21)

- Full rename to *standardjson* (missed some files in 0.3.0).

# CHAPTER 8

# 0.3.0 (2014-05-21)

- Rename package to *standardjson*.
- *StandardJSONEncoder* is now in *encoders* module.
- Encoder functions are now in *encoder_funcs* module.

# 0.2.0 (2014-05-20)

- Full implementation with tests.
- Separate *encoders* module for encoder functions.
- Bump to Alpha.

## 0.1.0 (2014-05-18)

- First release on PyPI.

# CHAPTER 11

## Indices and tables

- genindex
- modindex
- search