standardjson Documentation

Release 0.3.1

Audrey Roy

December 19, 2016

Contents

Contents:

CHAPTER 1

standardjson

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

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

1.1 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.

1.2 Quickstart

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"}'
```

1.3 FAQ

1.3.1 Does StandardJSONEncoder provide info about the Python type of the object?

No. *StandardJSONEncoder* purposely does not, in favor of a human-style, type-agnostic approach.

When encoded by *StandardJSONEncoder*, there is no differentiation between the string *"2010-02-17"* and the date object *date(2010, 2, 17)}*. This is the same approach described in ECMA-404, Introduction, paragraph 2:

"JSON is agnostic about numbers. In any programming language, there can be a variety of number types of various capacities and complements, fixed or floating, binary or decimal. That can make interchange between different programming languages difficult. JSON instead offers only the representation of numbers that humans use: a sequence of digits. All programming languages know how to make sense of digit sequences even if they disagree on internal representations. That is enough to allow interchange."

1.3.2 What if my application requires Python language-dependent JSON?

In that case, it's not a good use case for this package. The use case I have in mind is for taking Python objects and turning them into language-independent JSON. This is in the spirit of what JSON is designed for.

As described on json.org:

"JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language."

Most real-world use cases of JSON should be fine with language-independent JSON, of course.

CHAPTER 2

Installation

At the command line:

```
$ easy_install standardjson
```

Or, if you have virtualenvwrapper installed:

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

CHAPTER 3

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"}'
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 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.

4.1.5 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 :)

4.2 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.

4.3 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.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.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Audrey Roy (@audreyr)

5.2 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.

CHAPTER 9

0.2.0 (2014-05-20)

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

CHAPTER 10

0.1.0 (2014-05-18)

- First release on PyPI.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search