jsonvalidate Documentation

Release 0.1.7

Robus Gauli

Nov 25, 2019

Contents:

CHAPTER 1

Installation

1.1 Stable release

To install jsonvalidate, run this command in your terminal:

```
$ pip install jsonvalidate
```

This is the preferred method to install jsonvalidate, 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.

1.2 From sources

The sources for jsonvalidate can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/robusgauli/jsonvalidate
```

Or download the tarball:

```
$ curl  -OL https://github.com/robusgauli/jsonvalidate/tarball/master
```

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

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

CHAPTER 2

Usage

To use jsonvalidate in a project:

```
import jsonvalidate
```

CHAPTER 3

jsonvalidate

3.1 jsonvalidate package

3.1.1 Submodules

3.1.2 jsonvalidate.jsonvalidate module

jsonvalidate.jsonvalidate

Module that provides a helper classes for defining schema and validation for json

class jsonvalidate.jsonvalidate.**BooleanContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Type*

class jsonvalidate.jsonvalidate.**Contract**(**args*, ***kwargs*)
Bases: object

Abstract Base class for both primitives types

check(**args*)
last method in MRO chain that will eventually return false as an error

class jsonvalidate.jsonvalidate.**EnumContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

check(*val*)
last method in MRO chain that will eventually return false as an error

class jsonvalidate.jsonvalidate.**EnumError**(*actual*, *enums*)
Bases: *jsonvalidate.jsonvalidate.Error*

A class that represents enum invalidation error

class jsonvalidate.jsonvalidate.**Error**
Bases: object

Base class that is subclassed by Concrete error types.

todict()
Converts python object to serializable dictionary.

class jsonvalidate.jsonvalidate.**FloatContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Type*

Type Contract for Float

class jsonvalidate.jsonvalidate.**IntegerContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Type*

Type Contract for Integer

class jsonvalidate.jsonvalidate.**KeyMissingContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

check(*val*)
Checks for key mismatch

class jsonvalidate.jsonvalidate.**KeyMissingError**
Bases: *jsonvalidate.jsonvalidate.Error*

A class that represents key mismatch error

class jsonvalidate.jsonvalidate.**LengthContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

check(*val*)
last method in MRO chain that will eventually return false as an error

class jsonvalidate.jsonvalidate.**LengthError**(*actual_length=None*, *expected_min_length=None*, *expected_max_length=None*)
Bases: *jsonvalidate.jsonvalidate.Error*

A class that represents length invalidation error

class jsonvalidate.jsonvalidate.**List**(*object_shape*)
Bases: *jsonvalidate.jsonvalidate.Contract*

check(*value*)
last method in MRO chain that will eventually return false as an error

class jsonvalidate.jsonvalidate.**NullContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

A Null Contract class that implements check method for nullable value

check(*val*)
Checks if the value is null and delegate the method call to next method in MRO

class jsonvalidate.jsonvalidate.**NullError**
Bases: *jsonvalidate.jsonvalidate.Error*

A class that represents null error

class jsonvalidate.jsonvalidate.**Object**(*object_shape*)
Bases: *jsonvalidate.jsonvalidate.Contract*

check(*value*)
last method in MRO chain that will eventually return false as an error

class jsonvalidate.jsonvalidate.**RangeContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

Applicable to Integer

check(*val*)
last method in MRO chain that will eventually return false as an error

class jsonvalidate.jsonvalidate.**RangeError**(*actual_val*, *valid_range*)
Bases: *jsonvalidate.jsonvalidate.Error*

A subclass of error for range validation

class jsonvalidate.jsonvalidate.**RegExContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

check(*val*)
Checks if the value match regex and delegate the method call to next method in MRO

class jsonvalidate.jsonvalidate.**RegExError**
Bases: *jsonvalidate.jsonvalidate.Error*

A class that represents regex error

class jsonvalidate.jsonvalidate.**StringContract**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Type*

Type Contract for String

class jsonvalidate.jsonvalidate.**Type**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.Contract*

Abstract Base class for Type validation

check(*val*)
Checks for type mismatch.

jsonvalidate.jsonvalidate.**err**(*error*)
Utility function for returning serializable json payload.

3.1.3 Module contents

Top-level package for jsonvalidate.

class jsonvalidate.**String**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.KeyMissingContract*, *jsonvalidate.jsonvalidate.NullContract*, *jsonvalidate.jsonvalidate.StringContract*, *jsonvalidate.jsonvalidate.RegExContract*, *jsonvalidate.jsonvalidate.LengthContract*, *jsonvalidate.jsonvalidate.EnumContract*

Composition/Mixins for String

class jsonvalidate.**Integer**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.KeyMissingContract*, *jsonvalidate.jsonvalidate.NullContract*, *jsonvalidate.jsonvalidate.IntegerContract*, *jsonvalidate.jsonvalidate.RangeContract*, *jsonvalidate.jsonvalidate.EnumContract*

Composition/Mixins for Integer

class jsonvalidate.**Float**(**args*, ***kwargs*)
Bases: *jsonvalidate.jsonvalidate.KeyMissingContract*, *jsonvalidate.jsonvalidate.NullContract*, *jsonvalidate.jsonvalidate.FloatContract*, *jsonvalidate.jsonvalidate.RangeContract*, *jsonvalidate.jsonvalidate.EnumContract*

Composition/Mixins for Float

class jsonvalidate.**Boolean**(**args*, ***kwargs*)

Bases: *jsonvalidate.jsonvalidate.KeyMissingContract*, *jsonvalidate.jsonvalidate.NullContract*, *jsonvalidate.jsonvalidate.BooleanContract*

Composition/Mixins for Boolean

class jsonvalidate.**Object**(*object_shape*)

Bases: *jsonvalidate.jsonvalidate.Contract*

check(*value*)

last method in MRO chain that will eventually return false as an error

class jsonvalidate.**List**(*object_shape*)

Bases: *jsonvalidate.jsonvalidate.Contract*

check(*value*)

last method in MRO chain that will eventually return false as an error

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/robusgauli/jsonvalidate/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" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

jsonvalidate could always use more documentation, whether as part of the official jsonvalidate 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/robusgauli/jsonvalidate/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 *jsonvalidate* for local development.

1. Fork the *jsonvalidate* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/jsonvalidate.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 jsonvalidate
$ cd jsonvalidate/
$ 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 jsonvalidate tests
$ python setup.py test or 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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/robusgauli/jsonvalidate/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_jsonvalidate
```

4.5 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:

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

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Robus Gauli <robusgauli@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2018-06-08)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

j

Index

T