roman$_n umerals_w ebservice Documentation

Release __version__ = '0.4.1'

Thorsten Beier

Jul 24, 2019

Contents:

CHAPTER 1

roman_numerals_webservice

A python package for a minimalistic web-service to convert roman numerals to arabic numerals and vice versa.

- Free software: MIT license
- Documentation: https://roman-numerals-webservice.readthedocs.io.
- Try package online

1.1 Features

- Python code to to convert Roman numerals to Arabic numerals and vice versa
- Minimalistic web-service to convert Roman numerals to Arabic numerals and vice versa
- Heavy tested
- Nightly builds on continous integration servers
- available as pypi-package package
- Documentation on readthedocs.org
- Available on dockerhub

CHAPTER 2

Installation

2.1 Stable release

To install roman_numerals_webservice, run this command in your terminal:

```
$ pip install roman_numerals_webservice
```

This is the preferred method to install roman_numerals_webservice, 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.

2.2 From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/DerThorsten/roman_numerals_webservice
```

Or download the tarball:

```
$ curl  -OL https://github.com/DerThorsten/roman_numerals_webservice/tarball/master
```

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

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

CHAPTER 3

Tests

To run the unit tests type

```
make test
```

To run code coverage analysis type

```
make coverage
```

This will open a html report in a browser once finished.

Warning: The tests will start servers at `localhost:8080`. Make sure that this port is free before running the tests otherwise the tests might not run properly.

CHAPTER 4

Usage

After installing the python package `roman_numerals_webservice` the webserver can be started with the following command

```
roman_numerals_webservice
```

Alternatively we provide a prebuild docker container to start the web-service

```
sudo docker run -p 8080:8080 derthorsten/roman_numerals_webservice:latest
```

Once the server is running requests can be send to the web-service. On a Unix system this can be done with `curl`

```
$ curl -d '{"roman" : "XL"}' -H "Content-Type: application/json" -X POST http://
↪localhost:8080/roman_to_arabic
"{\"arabic\": 40}"
```

```
$ curl -d '{"arabic" : 1987}' -H "Content-Type: application/json" -X POST http://
↪localhost:8080/arabic_to_roman
"{\"roman\": \"MCMLXXXVII\"}"
```

To roman_numerals_webservice can also be started from python

```
import cherrypy
from roman_numerals_webservice import RomanNumeralsWebservice



if __name__ == "__main__":
    config = {
        'server.socket_port': 8080,
        'server.socket_host': '0.0.0.0',
        'environment': 'production',
    }
```

(continues on next page)

(continued from previous page)

```
    cherrypy.config.update(config)
    cherrypy.quickstart(RomanNumeralsWebservice())
```

CHAPTER 5

Docker

Assuming you are in the root dir of the repository, the following will build the docker container

```
sudo docker build -t roman_numerals_webservice .
```

To start the server use the following:

```
docker run -p 8080:8080 roman_numerals_webservice
```

Alternatively one can use the prebuild docker image hosted at dockerhub:

```
sudo docker run -p 8080:8080 derthorsten/roman_numerals_webservice:latest
```

CHAPTER 6

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:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/DerThorsten/roman_numerals_webservice/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.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

6.1.3 Implement Features

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

6.1.4 Write Documentation

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

6.1.5 Submit Feedback

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

6.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/roman_numerals_webservice.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 roman_numerals_webservice
$ cd roman_numerals_webservice/
$ 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 roman_numerals_webservice 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.

6.3 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/DerThorsten/roman_numerals_webservice/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ py.test tests.test_roman_numerals_webservice
```

6.5 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 7

Credits

7.1 Development Lead

- Thorsten Beier <derthorstenbeier@gmail.com>

7.2 Contributors

None yet. Why not be the first?

CHAPTER 8

roman_numerals_webservice

8.1 roman_numerals_webservice package

8.1.1 Subpackages

roman_numerals_webservice.roman_numerals package

Submodules

roman_numerals_webservice.roman_numerals.arabic_to_roman module

class roman_numerals_webservice.roman_numerals.arabic_to_roman.**ArabicToRoman**
Bases: object

static convert(*arabic: int*) → str
Convert an Arabic numeral to a Roman numeral

To convert Arabic numerals we chose the algorithm from Paul M. Winkler presented in "Python Cookbook" by David Ascher, Alex Martelli ISBN: 0596001673. since it is arguably the most readable algorithm.

Parameters **arabic** (*int*) – Arabic numeral represented as integer. The number must be be in [1,...,3999]

Raises

- TypeError – arabic does not satisfy isinstance(arabic, numbers.Integral) must be true.
- ValueError – arabic does not satisfy 1 <= v <= 3999

Returns string encoding the input as Roman numeral

Return type str

roman_numerals_webservice.roman_numerals.arabic_to_roman.**arabic_to_roman**(*arabic: int*) → str

Convert an Arabic numeral to a Roman numeral

Shorthand for *ArabicToRoman.convert()*, see *ArabicToRoman.convert()* for full documentation.

Parameters **arabic** – Arabic numeral represented as integer.

Raises

- TypeError – arabic does not satisfy isinstance(arabic, numbers.Integral) must be true.
- ValueError – arabic does not satisfy 1 <= v <= 3999

Returns string encoding the input as Roman numeral

Return type str

roman_numerals_webservice.roman_numerals.roman_to_arabic module

class roman_numerals_webservice.roman_numerals.roman_to_arabic.**RomanToArabic**

Bases: object

static convert(*roman: str*) → int

Convert a Roman numeral to an Arabic Numeral.

To convert Arabic numerals we chose the algorithm from Paul M. Winkler presented in "Python Cookbook" by David Ascher, Alex Martelli ISBN: 0596001673. since it is arguably the most readable algorithm.

Parameters **roman** (*str*) – Roman numeral represented as string.

Raises

- TypeError – roman is not a string
- ValueError – roman is not a valid Roman numeral

Returns int encoding the input as Arabic numeral

Return type int

roman_numerals_webservice.roman_numerals.roman_to_arabic.**roman_to_arabic**(*arabic: int*) → str

Convert a Roman numeral to an Arabic Numeral.

Shorthand for *RomanToArabic.convert()*, see

RomanToArabic.convert() for full documentation.

Parameters **roman** (*str*) – Roman numeral represented as string.

Raises

- TypeError – roman is not a string
- ValueError – roman is not a valid Roman numeral

Returns int encoding the input as Arabic numeral

Return type int

Module contents

8.1.2 Submodules

8.1.3 roman_numerals_webservice.cli module

Console script for roman_numerals_webservice.

8.1.4 roman_numerals_webservice.roman_numerals_webservice module

Main module.

class roman_numerals_webservice.roman_numerals_webservice.**RomanNumeralsWebservice**

Bases: object

arabic_to_roman()

Implements endpoint for Arabic to Roman web-service

This method expects json request with a json payload of the following form:

```
'{"roman": <input_str>}'
```

Where <input_str> is the input Arabic numerals as integer.

With curl, one can use this endpoint in the following way:

```
$ curl -d '{"arabic" : 1987}' -H "Content-Type: application/json" -X POST␣
↪http://localhost:8080/arabic_to_roman
```

The output will be:

```
"{"roman": "MCMLXXXVII"}"
```

Returns

json string of the form '{"roman": <res_str>}' where

<res_str> is an str which represents the input Arabic numeral converted to a Roman Numeral.

Return type str

Raises cherrypy.HTTPError – If the input is not a in the range [1,...399], if the json payload is ill-formed, a cherrypy.HTTPError(status=400) is raised. This error will translate to a BAD_REQUEST HTML status code.

roman_to_arabic()

Implements endpoint for Roman to Arabic web-service

This method expects json request with a json payload of the following form:

```
'{"roman": <input_str>}'
```

Where <input_str> is the input Roman numerals as string. The string can be in CAPTIAL lower or mIxeD case.

With curl, one can use this endpoint in the following way:

```
$ curl -d '{"roman" : "XL"}' -H "Content-Type: application/json" -X POST␣
↪http://localhost:8080/roman_to_arabic
```

The output will be:

```
{"arabic": 40}"
```

Returns

json string of the form '{"arabic": <res_int>}' where

<res_int> is an integer which represents the input Roman numeral converted to an Arabic Numeral.

Return type str

Raises cherrypy.HTTPError – If the input it not a valid Roman numeral or if the json payload is ill-formed, a cherrypy.HTTPError(status=400) is raised. This error will translate to a BAD_REQUEST HTML status code.

8.1.5 Module contents

Top-level package for roman_numerals_webservice.

CHAPTER 9

History

9.1 0.4.1 (2019-07-24)

- fixed urls

9.2 0.4.0 (2019-07-24)

- improved documentation
- added nightly builds on travis
- renamed int_to_x to arabic_to_x
- added better notebook
- added Dockerfile

9.3 0.3.3 (2019-07-24)

- changed entry point

9.4 0.3.2 (2019-07-23)

- fix pyi version

9.5 0.3.1 (2019-07-23)

- fixed version issue

9.6 0.3.0 (2019-07-23)

- fixed missing subpackages
- added examples

9.7 0.2.0 (2019-07-23)

- First release on PyPI.

9.8 0.1.0 (2019-07-23)

- Initial version

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

A

C

R