Llic Documentation

Release 0.0.5

Hal Blackburn

August 19, 2014

Contents:

CHAPTER 1

Low-Level iCalendar (llic)

A Python library for efficient generation of iCalendar content. A streaming, incremental output model is used rather than the normal method of building up an in memory representation of the iCalendar document and outputting it in one go.

The library is driven by the need to generate iCalendar documents faster than the Python icalendar library currently does (by ~10x according to my benchmarks).

I wrote this library while optimising the iCal generation code of https://www.timetable.cam.ac.uk/.

- Free software: BSD license
- Documentation: https://llic.readthedocs.org.

CHAPTER 2

Installation

At the command line:

```
$ easy_install llic
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv llic
$ pip install llic
```

CHAPTER 3

Usage

To use Llic in a project:

```
import llic
```

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/h4l/llic/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

Llic could always use more documentation, whether as part of the official Llic 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/h4l/llic/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 *llic* for local development.

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

   ```
   $ git clone git@github.com:your_name_here/llic.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 llic
   $ cd llic/
   $ 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 llic 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, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/h4l/llic/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_llic
```

CHAPTER 5

TODO

- Add an optional validity check to detect llic methods being called in a sequence which would produce invalid output.
- More output types?
- C/C++ implementation?

CHAPTER 6

Credits

6.1 Development Lead

- Hal Blackburn <hal.blackburn@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

CHAPTER 8

0.0.5 (2014-08-18)

- Updated setup.py
- Added docs
- No functional changes from 0.0.4
- First release on PyPI.

CHAPTER 9

0.0.4 (2013-09-06)

- Fix an issue with setup.py indirectly importing pytz before pytz was installed via setup.py

CHAPTER 10

0.0.3 (2013-08-20)

- Added some unit tests
- Fixed an issue with write() always appending a “rn ” to wrapped output

0.0.2 (2013-06-17)

- Fixed setup.py not listing llic as a module to install

CHAPTER 12

0.0.1 (2013-06-17)

- Initial version

CHAPTER 13

Indices and tables

- *genindex*
- *modindex*
- *search*