cipher$_jat2167Documentation$

Julia Tache

Nov 16, 2020

CONTENTS:

1 Installation **3**

1.1 Stable release 3

1.2 From sources 3

2 Usage **5**

3 Contributing **7**

3.1 Types of Contributions 7

3.2 Get Started! 8

3.3 Pull Request Guidelines 9

3.4 Code of Conduct 9

4 Code of Conduct **11**

4.1 Our Pledge 11

4.2 Our Standards 11

4.3 Our Responsibilities 11

4.4 Scope 12

4.5 Enforcement 12

4.6 Attribution 12

5 Indices and tables **13**

This cookiecutter creates a boilerplate for a Python cipher project.

To get started, check out the sections below:

CHAPTER ONE

INSTALLATION

1.1 Stable release

To install cipher_jat2167, run this command in your terminal:

```
$ pip install -u cipher_jat2167
```

This is the preferred method to install cipher_jat2167, 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 cipher_jat2167 can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/julia-tache/cipher_jat2167
```

Or download the tarball:

```
$ curl  -OL https://github.com/julia-tache/cipher_jat2167/tarball/main
```

Once you have a copy of the source, you can install it. The method of installation will depend on the packaging library being used.

For example, if *setuptools* is being used (a setup.py file is present), install cipher_jat2167 with:

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

If *poetry* is being used (poetry.lock and pyproject.toml files are present), install cipher_jat2167 with:

```
$ poetry install
```

CHAPTER TWO

USAGE

To use cipher_jat2167 in a project:

```
import cipher_jat2167
```

CHAPTER

THREE

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:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/julia-tache/cipher_jat2167/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.

3.1.2 Fix Bugs

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

3.1.3 Implement Features

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

3.1.4 Write Documentation

cipher_jat2167 could always use more documentation, whether as part of the official cipher_jat2167 docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/julia-tache/cipher_jat2167/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 :)

3.2 Get Started!

Ready to contribute? Here's how to set up *cipher_jat2167* for local development.

1. Fork the *cipher_jat2167* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/cipher_jat2167.git
```

3. Install your local copy (it is recommended to do this with a virtual environment). The method of installation will depend on the packaging library being used. For example, if *setuptools* is being used (a setup.py file is present), install cipher_jat2167 with:

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

If *poetry* is being used (poetry.lock and pyproject.toml files are present), install cipher_jat2167 with:

```
$ poetry install
```

4. Create a branch for local development and make your changes locally:

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

5. When you're done making changes, check that your changes conform to any code formatting requirements and pass any tests. For example, if the package uses the poetry package management library, black formatting style and pytest for testing:

```
$ poetry run black cipher_jat2167
$ poetry run pytest
```

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.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include additional tests if appropriate.
2. If the pull request adds functionality, the docs should be updated.
3. The pull request should work for all currently supported operating systems and versions of Python.

3.4 Code of Conduct

Please note that the cipher_jat2167 project is released with a Contributor Code of Conduct. By contributing to this project you agree to abide by its terms.

CHAPTER

FOUR

CODE OF CONDUCT

4.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

4.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

4.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

4.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant homepage, version 1.4.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search