FHanalyze Documentation

Margaret Johnson

Apr 02, 2020

Contents:

CHAPTER 1

FHanalyze

Analyze power readings provided by the FHmonitor package.

- Free software: MIT license
- Documentation: https://FHanalyze.readthedocs.io.

1.1 Features

- Gathers the readings stored by FHmonitor into the Raspberry Pi's mongodb and provides a variety of data analysis.

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install FHanalyze, run this command in your terminal:

```
$ pip install FHanalyze
```

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

You can either clone the public repository:

```
$ git clone git://github.com/bitknitting/FHanalyze
```

Or download the tarball:

```
$ curl -OJL https://github.com/bitknitting/FHanalyze/tarball/master
```

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

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

CHAPTER 3

Usage

To use FHanalyze in a project:

```
import FHanalyze
```

CHAPTER 4

FHanalyze package

4.1 FHanalyze package

The FHanalyze package analyzes the readings that were stored into the Raspberry Pi's mongo db by calls to the FHmonitor package.

4.1.1 Analyze class

The analyze class takes the power readings from the mongodb store and provides methods to analyze the readings.

class FHanalyze.analyze.**Analyze**(*mongodb_path='mongodb://127.0.0.1:27017'*, *db_str='FitHome'*, *collection_name='aggregate'*)

The Analyze class connects to the mongodb collection that contains the power readings saved by the FHmonitor.

To get an instance of the Analyze class, pass into __init__:

Parameters

- **mongodb_path** – The mongodb connection string. See the connection string documentation. Defaults to "mongodb://127.0.0.1:27017".
- **db_str** – The database within mongodb that holds the readings. Defaults to "FitHome".
- **collection_name** – The collection within the database where the readings are stored. Defaults to "aggregate".

get_DataFrame_for_date(*date_str='*'*)

Return the active Power readings for a specific date or for all dates.

Parameters **date** – isodate formatted date or "*". Defaults to "*"

Returns A pandas DataFrame with a dateindex and a column named 'pA' of active power readings.

get_always_on_watts(*date_str='*'*, *start_time='*'*, *end_time='*'*, *quantile=0.3*)

Return the amount of power (in watts) that are wasted by appliances that are always plugged in and use electricity, even when the appliance is not being used.

Parameters

- **date_str** – Either a date in isodate format to use readings in the calculation for that date or '*' to use all available dates for the calculation. Defaults to '*'
- **start_time** – Use readings that are taken at this time. The start_time is the beginning of an amount of time to filter results. Defaults to '*'
- **end_time** – Use readings that are taken before this time. end_time is used with start_time to filter readings to be within the time between start_time and end_time. Defaults to '*'
- **quantile** – A floating point number between 0 and 1 that cuts the distribution of readings. For example, a value of .3 means the always_on_watts value is taken at the place where 30% of the readings are found. Defaults to .3.

get_isodate_list()

Get the list of dates in isodate format that contain active and reactive power readings.

Returns list of isodates that contain power readings.

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/bitknitting/FHanalyze/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

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

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/FHanalyze.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 FHanalyze
$ cd FHanalyze/
$ 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 FHanalyze tests
$ python setup.py test or pytest
$ 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.

5.3 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/bitknitting/FHanalyze/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_FHanalyze
```

5.5 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:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Margaret Johnson <happyday.mjohnson@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.0.1 (2020-03-16)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index