nombot Documentation

Release 2.2.8

Bobby

Oct 18, 2019

Contents:

CHAPTER 1

nombot

A flexible cryptocurrency trading bot written in Python and `bors`.

1.1 Dependencies

Some assumptions are made that you are familiar with Python and operate in Linux. There is currently no way to test on Windows or Apple operating systems, and while you may use the OTS strategies that come with this repository, in order to expand on your strategy development time is a likely requirement.

If you are able to use `docker`, this will solve the first problem. If you are able to pay for development services, that will solve the second.

1.1.1 Python Dependencies

- Python 3.6+ is assumed
- `bors`
- `ccxt`
- `dataclasses`

1.2 Key Features

1.2.1 Exchanges

- Extensible for any exchange using CCXT.
- Exchange helper facilities; _connection pooling, websockets, shared contexts.
- Mutiple exchange connectivity allowing for arbitrage.

- Strive toward dynamic, forward-compatible interfaces.

1.2.2 Currencies

All currencies supported; *if the exchange supports it, it will work*. * Wallet support, allowing for automated coin transfers. *(coming soon)*

1.2.3 Algorithms

Implementation is independent of strategy, allowing for maximal reuse, flexibility, and enforcing DRY principles.

1.2.4 Strategies

- "Plugable" *strategy* architecture using `IStrategy` inheritance.
- "Stackable" architecture allows you to isolate grouped functionality for reuse.
- Utilizes a middleware pipeline for processing.
- Shared context objects allow for maximum versatility in complex scenarios.
- Utilization of algorithms as backend functional libraries (strategy equates to "business logic").
- Automatic configuration pass-through.
- See `bors` for more information.

1.2.5 Configuration

Flexible modularized configuration using JSON. See `config.json.example`.

1.2.6 Security

- Namespaced immutable configuration will greatly reduce your chance of information leakage and manipulation.
- File system storage & security (requires careful consideration of file permissions; see install notes below).

1.2.7 Coming soon...

- Backtesting and supporting documentation.
- More documentation around creating exchanges, algorithms, and strategies.
- Tests, tests, tests!
- For a deployment example, see `siphonexchange`.

1.3 Setup

1.3.1 Installation

```
git clone https://github.com/karma0/nombot.git nombot && cd $_
pip install requests numpy pandas
```

1.3.2 Upgrading to the latest

The `master` branch will contain the latest release. `develop` will contain the latest developments and may break things.

```
git pull
```

1.3.3 Configuration

1. Create your strategy class, using any available algorithms, or creating your own algorithms.
2. Copy `config.json.example` to `config.json` and execute `chmod 600 config.json`.
3. Change the configuration required for your strategy, exchange(s), API calls, etc. based on the examples.

1.3.4 Execution

```
./trader.py
```

1.4 Contributing

Options: 1. Follow the instructions here: https://help.github.com/articles/fork-a-repo/ 2. Submit an issue or feature request here.

CHAPTER 2

Installation

2.1 Stable release

To install nombot, run this command in your terminal:

```
$ pip install nombot
```

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

You can either clone the public repository:

```
$ git clone git://github.com/karma0/nombot
```

Or download the tarball:

```
$ curl  -OL https://github.com/karma0/nombot/tarball/master
```

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

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

CHAPTER 3

Usage

To use nombot in a project:

```
import nombot
```

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/karma0/nombot/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

nombot could always use more documentation, whether as part of the official nombot 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/karma0/nombot/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 *nombot* for local development.

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

```
$ git clone git@github.com:your_name_here/nombot.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 nombot
$ cd nombot/
$ 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 nombot 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/karma0/nombot/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_nombot
```

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

- Bobby <karma0@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 2.2.8 (2018-06-26)

- Added *ccxt* to the requirements.

6.2 2.2.7 (2018-06-26)

- Update README with badges.
- Upgraded dependencies.
- Added several unit tests.

6.3 2.2.6 (2018-06-26)

- Fixed CI and added some tests around configuration.

6.4 2.2.5 (2018-06-25)

- Setup readthedocs.org.

6.5 2.2.4 (2018-06-25)

- Replaced packaging with cookiecutter-pypackage.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search