ritpytrading Documentation

Release 0.1.3

Samridha Man Shrestha

Mar 11, 2019

Contents:

1 ritpytrading **1**
- 1.1 RIT-trading-python . . . 1
- 1.2 Prerequisites . . . 2
- 1.3 Installing . . . 2
- 1.4 Building dists and running tests using makefile . . . 3
- 1.5 Running tests with the python unittest module . . . 3
- 1.6 Usage (Only on Windows) . . . 3
- 1.7 Built With . . . 4
- 1.8 Versioning . . . 4
- 1.9 Authors . . . 4
- 1.10 License . . . 4
- 1.11 Acknowledgments . . . 4
- 1.12 Contributions . . . 4
- 1.13 Disclaimer . . . 4

2 Installation **5**
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5

3 Usage **7**

4 ritpytrading **9**
- 4.1 ritpytrading package . . . 9

5 Contributing **15**
- 5.1 Types of Contributions . . . 15
- 5.2 Get Started! . . . 16
- 5.3 Pull Request Guidelines . . . 17
- 5.4 Tips . . . 17
- 5.5 Deploying . . . 17

6 Credits **19**
- 6.1 Development Lead . . . 19
- 6.2 Contributors . . . 19

7 History **21**

8 0.1.0 (2018-12-14) 23

9 0.1.1 (2018-12-15) 25

10 0.1.2 (2018-12-15) 27

11 0.1.3 (2019-01-11) 29

12 Indices and tables 31

Python Module Index 33

CHAPTER 1

ritpytrading

1.1 RIT-trading-python

Python trading module for the Rotman Interactive Trader trading software. PyPI page.

Full documentation available here.

1.2 Prerequisites

Python version 3

The Rotman Interactive Trading Client which can only be operated in a **Windows system**. However, development of the PyPI ritpytrading package can be in done in Linux/BSD envrionment as well.

The full documentation for the Rotman Interactive Trader Client REST API can be found at Swaggerhub. The documentation is also present in a JSON format in the swagger_client_generated folder.

1.3 Installing

1.3.1 RIT Client Software

The RIT Client for Windows system can be downloaded at http://rit.rotman.utoronto.ca/software.asp.

Instructions for setting up an RIT demonstration client account for the Liability Trading 3 case file can be found at RIT's website at http://rit.rotman.utoronto.ca/demo.asp.

Virtual environment packages with `virtualenv` or `anaconda` should be used for both Windows and Linux/BSD based systems.

1.3.2 Windows

Initialize the repository with git. Detailed instructions to download git for windows can be found at atlassian. The repository can then be initialized with git using:

```
$ git clone https://github.com/SamSamhuns/RIT-trading-python
```

Two options are available after this:

- Anaconda is recommended for Windows system. Set up up a virtual conda environment first. Then open the anaconda prompt and use the command `conda install --yes --file requirements.txt` to install all modules from requirements.txt.
- Or Install `python` and add it to your `PATH` system variable. Then install the `pip` package if not installed already also adding it to the `PATH` system variable. Then run the following commands.

```
$ pip install virtualenv
$ virtualenv venv
$ venv\Scripts\activate
$ pip install -r requirements.txt
```

1.3.3 Linux/BSD

After cloning the repository, install the required python packages using pip.

```
$ git clone https://github.com/SamSamhuns/RIT-trading-python
$ pip install virtualenv
$ virtualenv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
```

1.4 Building dists and running tests using makefile

For **Windows**, different options are available for using makefile. GnuWin's make provides a native port for Windows (without requiring a full runtime environment like Cygwin). After installing GnuWin, add `C:\Program Files (x86)\GnuWin32\bin` to your system `PATH` variable to run makefiles from any directory.

- For **Windows**, run makefile commands with `make -f Makefile.win <directive>`. Example `make -f Makefile.win help`
- For **Linux/BSD**, run makefile commands with `make <drective>`.

Run the following command to get a list of all Makefile command options.

```
$ make help
```

To run tests

```
$ make test
$ make test-all
```

To ensure the README.rst will be rendered in PyPI [might be outdated. Check twine check below]

```
$ python setup.py check --restructuredtext
```

To report any problems rendering your README. If your markup renders fine, the command will output Checking distribution FILENAME: Passed. To run the check on sdist and wheel.

```
$ twine check dist/*
```

To build the source and wheel package.

```
$ make build
```

1.5 Running tests with the python unittest module

Once python has been added to the `PATH` system variable in Windows, the code for running the scripts on Windows and Linux/BSD based systems are the same.

From the main directory, run:

```
$ python -m unittest
```

If no tests are run from the command above, run the verbose mode.

Verbose mode

```
$ python -m unittest discover -v
```

1.6 Usage (Only on Windows)

IMPORTANT: The RIT Trading client must also be running to make sure the REST RIT API Client requests can be made. In each script your **RIT Client API key** must be entered and the **requests** module be imported to make API calls.

To test out the ritpytrading package, install using pip inside a virtual environment:

```
$ pip install ritpytrading
```

Examples scripts are present inside the `examples` folder. Documentation for usage available here.

1.7 Built With

- Python 3 - The Programming tool used

1.8 Versioning

Version tracked with Git

1.9 Authors

- **Samridha Shrestha**

1.10 License

This project is licensed under the Apahce 2.0 License - see the LICENSE.md file for details

1.11 Acknowledgments

- Rotman School of Manangement, University of Toronto http://www.rotman.utoronto.ca/
- Rotman Interactive Trader http://rit.rotman.utoronto.ca/
- Python open source libraries
- Joel Hasbrouck, NYU Stern Principles of Securities Trading, FINC-UB.0049, Spring 201. http://people.stern.nyu.edu/jhasbrou/
- This project directory was created based on Cookiecutter and the audreyr/cookiecutter-pypackage project template.
- README conversion for PyPI. Pandoc.org.

1.12 Contributions

1.13 Disclaimer

All RIT software and external RIT links are provided by the Rotman School of Management and are their exclusive property.

CHAPTER 2

Installation

2.1 Stable release

To install ritpytrading, run this command in your terminal:

```
$ pip install ritpytrading
```

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

You can either clone the public repository:

```
$ git clone git://github.com/SamSamhuns/ritpytrading
```

Or download the tarball:

```
$ curl  -OL https://github.com/SamSamhuns/ritpytrading/tarball/master
```

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

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

CHAPTER 3

Usage

To use ritpytrading in a project:

```
import ritpytrading
```

CHAPTER 4

ritpytrading

4.1 ritpytrading package

4.1.1 Submodules

4.1.2 ritpytrading.assets module

This script contains results for the /assets module

Sample JSON output formats for the function returns News object return value: JSON formatted [

{ “ticker”: “string”, “type”: “CONTAINER”, “description”: “string”, “total_quantity”: 0, “available_quantity”: 0, “lease_price”: 0, “convert_from”: [

{ “ticker”: “string”, “quantity”: 0

}

], “convert_to”: [

{ “ticker”: “string”, “quantity”: 0

}

], “containment”: {

“ticker”: “string”, “quantity”: 0

}, “ticks_per_conversion”: 0, “ticks_per_lease”: 0, “is_available”: true, “start_period”: 0, “stop_period”: 0

}

] Parameters for the news GET HTTP request - ticker string (query)

exception ritpytrading.assets.**ApiException**

Bases: exceptions.Exception

to print error messages and stop the program when needed

class ritpytrading.assets.**Asset**(*asset_response*)
case_response is a json obj returned from the API get request

ritpytrading.assets.**asset**(*ses*, *ticker_sym*)
function that returns a single asset object given for a given ticker

ritpytrading.assets.**assets_dict**(*ses*)
function that returns a dictionary of the assets object

ritpytrading.assets.**assets_list**(*ses*)
returns a list of JSON fomratted output for assets object

4.1.3 ritpytrading.case module

4.1.4 ritpytrading.news module

This script contains results for the /news module

Sample JSON output formats for the function returns News object return value: JSON formatted [

{ "news_id": 0, "period": 0, "tick": 0, "ticker": "string", "headline": "string", "body": "string"

}

] Parameters for the news GET HTTP request - since number (query) - limit number (query)

exception ritpytrading.news.**ApiException**
Bases: exceptions.Exception

to print error messages and stop the program when needed

class ritpytrading.news.**News**(*news_response*)
case_response is a json obj returned from the API get request

ritpytrading.news.**news_dict**(*ses*, *since_id=None*, *limit_itm=None*)
function that returns the news object

ritpytrading.news.**news_json**(*ses*, *since_id=None*, *limit_itm=None*)
returns a list of JSON fomratted output for news object

4.1.5 ritpytrading.orders module

order return object attributes param possible order attributes: JSON formatted i.e. get_order_response(ses, url_end, param="order_id") {

"order_id": 1221, "period": 1, "tick": 10, "trader_id": "trader49", "ticker": "CRZY", "type": "LIMIT", "quantity": 100, "action": "BUY", "price": 14.21, "quantity_filled": 10, "vwap": 14.21, "status": "OPEN"

}

exception ritpytrading.orders.**ApiException**
Bases: exceptions.Exception

to print error messages and stop the program when needed

class ritpytrading.orders.**Order**(*order_response*)
order_response is a json obj returned from the API get request

ritpytrading.orders.**order**(*ses*, *orderId*, *status='OPEN'*)
status can be OPEN, TRANSACTED or CLOSED status OPEN by default returns a Order object of the order class given an order id

ritpytrading.orders.**orders_dict**(*ses*, *status='OPEN'*)
returns all the orders as a dict with the order_ids as key

ritpytrading.orders.**orders_json**(*ses*, *status='OPEN'*)
returns all the attribs of all orders in a json type list format

4.1.6 ritpytrading.securities module

The securities HTTP module gets a list of available securities and associated positions.

securities object attribute values: JSON formatted [

{ "ticker": "string", "type": "SPOT", "size": 0, "position": 0, "vwap": 0, "nlv": 0, "last": 0, "bid": 0, "bid_size": 0, "ask": 0, "ask_size": 0, "volume": 0, "unrealized": 0, "realized": 0, "currency": "string", "total_volume": 0, "limits": [

{ "name": "string", "units": 0

}

], "interest_rate": 0, "is_tradeable": true, "is_shortable": true, "start_period": 0, "stop_period": 0

}

]

Parameters for the securities GET HTTP request - ticker* required string (query)

exception ritpytrading.securities.**ApiException**
Bases: exceptions.Exception

to print error messages and stop the program when needed

class ritpytrading.securities.**Security**(*security_response*)
Security class takes a security_response object (a list of json objects) as its initializing paramenter to extract all relevant information security_response is a json obj returned from the API get request

ritpytrading.securities.**security_dict**(*ses*, *ticker_sym=None*)
By default no specific ticker_sym is None returns the list of available securities as a dict of security objects with ticker name as keys

ritpytrading.securities.**security_json**(*ses*, *ticker_sym=None*)
returns the list of available securities with all info in a json format

4.1.7 ritpytrading.securities_book module

The /securities/book HTTP module gets the order book of a security

securities_book object attribute values: JSON formatted {

"bids": [

{ "order_id": 1221, "period": 1, "tick": 10, "trader_id": "trader49", "ticker": "CRZY", "type": "LIMIT", "quantity": 100, "action": "BUY", "price": 14.21, "quantity_filled": 10, "vwap": 14.21, "status": "OPEN"

}

], “asks”: [

{ “order_id”: 1221, “period”: 1, “tick”: 10, “trader_id”: “trader49”, “ticker”: “CRZY”, “type”: “LIMIT”, “quantity”: 100, “action”: “BUY”, “price”: 14.21, “quantity_filled”: 10, “vwap”: 14.21, “status”: “OPEN”

}

]

}

Parameters for the securities_book GET HTTP request - ticker* required string (query) - period number (query)

exception ritpytrading.securities_book.**ApiException**
Bases: exceptions.Exception

to print error messages and stop the program when needed

ritpytrading.securities_book.**get_all_asks**(*ses*, *ticker_sym*)

ritpytrading.securities_book.**get_all_bids**(*ses*, *ticker_sym*)

ritpytrading.securities_book.**get_all_bids_asks**(*ses*, *ticker_sym*)
Returns a list of JSON objects representing all_flag the orders in the Bid and Ask side of the book

ritpytrading.securities_book.**get_bbo**(*ses*, *ticker_sym*)

ritpytrading.securities_book.**get_best_ask**(*ses*, *ticker_sym*)

ritpytrading.securities_book.**get_best_bid**(*ses*, *ticker_sym*)

ritpytrading.securities_book.**get_security_info**(*ses*, *ticker_sym*, *side*, *param*)
All possible values for the param parameter are listed at the top

Returns the value of the param for the given ticker from the given side side = bids / asks

4.1.8 ritpytrading.securities_history module

The /securities/history module gets the OHLC history for a security.

functions related to the history of a security securities_history object attribute values: JSON formatted [

{ “tick”: 11, “open”: 4.12, “high”: 4.21, “low”: 4.1, “close”: 4.15

}

]

Parameters for the securities_history GET HTTP request

- ticker* required string (query)
- period number (query)

Period to retrieve data from. Defaults to the current period. - limit number (query) Result set limit, counting backwards from the most recent tick. Defaults to retrieving the entire period.

exception ritpytrading.securities_history.**ApiException**
Bases: exceptions.Exception

to print error messages and stop the program when needed

class ritpytrading.securities_history.**Security_History**(*sec_history*)
sec_history is a json obj returned from the API get request

ritpytrading.securities_history.**security_history_dict**(*ses*, *ticker_sym*, *period_numb=None*, *lim_numb=None*)

function to get values of different parameters

ritpytrading.securities_history.**security_history_json**(*ses*, *ticker_sym*, *period_numb=None*, *lim_numb=None*)

get all full JSON response for the securities history get request

4.1.9 ritpytrading.submit_cancel_orders module

exception ritpytrading.submit_cancel_orders.**ApiException**

Bases: exceptions.Exception

to print error messages and stop the program when needed

ritpytrading.submit_cancel_orders.**cancel_order**(*ses*, *order_id*)

function requires a requests.Session() object as the ses argument with a loaded API_KEY

ritpytrading.submit_cancel_orders.**cancel_order_bulk**(*ses*, *price_direc*, *price_lim*, *volume_direc*, *volume_lim*, *all_flag=0*)

Volume < 0 for cancelling all open sell orders and Volume > 0 for cancelling all open buy orders query_gen example 'Price < 20.0 AND Volume > 0'

ritpytrading.submit_cancel_orders.**limit_order**(*ses*, *ticker*, *side*, *quantity*, *price*)

function requires a requests.Session() object as the ses argument with a loaded API_KEY

ritpytrading.submit_cancel_orders.**market_order**(*ses*, *ticker*, *side*, *quantity*)

submitting a market order side = BUY/SELL function requires a requests.Session() object as the ses argument with a loaded API_KEY

4.1.10 ritpytrading.tenders module

This script contains results for the /tenders module

Sample JSON output formats for the function returns Tender object return value: JSON formatted [

{ "tender_id": 0, "period": 0, "tick": 0, "expires": 0, "caption": "string", "quantity": 0, "action": "BUY", "is_fixed_bid": true, "price": 0

}

]

exception ritpytrading.tenders.**ApiException**

Bases: exceptions.Exception

to print error messages and stop the program when needed

class ritpytrading.tenders.**Tender**(*tender_response*)

case_response is a json obj returned from the API get request

ritpytrading.tenders.**accept_tender**(*ses*, *tender_iden*, *price_tender=None*)

ritpytrading.tenders.**decline_tender**(*ses*, *tender_iden*)

ritpytrading.tenders.**tenders_dict**(*ses*)

function that returns the tender object

ritpytrading.tenders.**tenders_json**(*ses*)
returns a list of JSON fomratted output for tender object

4.1.11 ritpytrading.trader module

4.1.12 Module contents

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/SamSamhuns/ritpytrading/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

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

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

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

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 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/SamSamhuns/ritpytrading/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_assets
```

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:

```
$ bumpversion 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

- Samridha Man Shrestha <sms1198@nyu.edu>

6.2 Contributors

Contributions are absolutely welcomed.

CHAPTER 7

History

CHAPTER 8

0.1.0 (2018-12-14)

- First release on PyPI.

CHAPTER 9

0.1.1 (2018-12-15)

- Second release on PyPI.

CHAPTER 10

0.1.2 (2018-12-15)

- Third release on PyPI with correctly rendering README.

CHAPTER 11

0.1.3 (2019-01-11)

- Fourth release with major corrections.

CHAPTER 12

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

ritpytrading, 14
ritpytrading.assets, 9
ritpytrading.news, 10
ritpytrading.orders, 10
ritpytrading.securities, 11
ritpytrading.securities_book, 11
ritpytrading.securities_history, 12
ritpytrading.submit_cancel_orders, 13
ritpytrading.tenders, 13

Index

A

accept_tender() (*in module ritpytrading.tenders*), 13
ApiException, 9–13
Asset (*class in ritpytrading.assets*), 9
asset() (*in module ritpytrading.assets*), 10
assets_dict() (*in module ritpytrading.assets*), 10
assets_list() (*in module ritpytrading.assets*), 10

C

cancel_order() (*in module ritpytrading.submit_cancel_orders*), 13
cancel_order_bulk() (*in module ritpytrading.submit_cancel_orders*), 13

D

decline_tender() (*in module ritpytrading.tenders*), 13

G

get_all_asks() (*in module ritpytrading.securities_book*), 12
get_all_bids() (*in module ritpytrading.securities_book*), 12
get_all_bids_asks() (*in module ritpytrading.securities_book*), 12
get_bbo() (*in module ritpytrading.securities_book*), 12
get_best_ask() (*in module ritpytrading.securities_book*), 12
get_best_bid() (*in module ritpytrading.securities_book*), 12
get_security_info() (*in module ritpytrading.securities_book*), 12

L

limit_order() (*in module ritpytrading.submit_cancel_orders*), 13

M

market_order() (*in module ritpytrading.submit_cancel_orders*), 13

N

News (*class in ritpytrading.news*), 10
news_dict() (*in module ritpytrading.news*), 10
news_json() (*in module ritpytrading.news*), 10

O

Order (*class in ritpytrading.orders*), 10
order() (*in module ritpytrading.orders*), 10
orders_dict() (*in module ritpytrading.orders*), 11
orders_json() (*in module ritpytrading.orders*), 11

R

ritpytrading (*module*), 14
ritpytrading.assets (*module*), 9
ritpytrading.news (*module*), 10
ritpytrading.orders (*module*), 10
ritpytrading.securities (*module*), 11
ritpytrading.securities_book (*module*), 11
ritpytrading.securities_history (*module*), 12
ritpytrading.submit_cancel_orders (*module*), 13
ritpytrading.tenders (*module*), 13

S

Security (*class in ritpytrading.securities*), 11
security_dict() (*in module ritpytrading.securities*), 11
Security_History (*class in ritpytrading.securities_history*), 12
security_history_dict() (*in module ritpytrading.securities_history*), 13
security_history_json() (*in module ritpytrading.securities_history*), 13

security_json() (*in module ritpytrading.securities*), 11

T

Tender (*class in ritpytrading.tenders*), 13
tenders_dict() (*in module ritpytrading.tenders*), 13
tenders_json() (*in module ritpytrading.tenders*), 13