pyrh

Release 2.0

Unofficial Robinhood Python API

Sep 26, 2020

CONTENTS

1 *Please note that parts of this project maybe non-functional / under rapid development* **3**
- 1.1 Quick start . . . 3
- 1.2 How To Install: . . . 3
- 1.3 Running example.ipynb . . . 3
- 1.4 Related . . . 4

2 Developers **5**
- 2.1 Developer Guidelines . . . 5

3 API **7**
- 3.1 API Reference . . . 7

4 Changelog **23**
- 4.1 Changelog . . . 23

5 Indices and tables **25**

Python Module Index **27**

Index **29**

Python Framework to make trades with Unofficial Robinhood API. Supports Python 3.6+

CHAPTER

ONE

PLEASE NOTE THAT PARTS OF THIS PROJECT MAYBE NON-FUNCTIONAL / UNDER RAPID DEVELOPMENT

- A stable release is imminent

Documentation: https://pyrh.readthedocs.io/en/latest/

1.1 Quick start

```
from pyrh import Robinhood

rh = Robinhood()
rh.login(username="YOUR_EMAIL", password="YOUR_PASSWORD")
rh.print_quote("AAPL")
```

1.2 How To Install:

```
pip install pyrh
```

1.3 Running example.ipynb

Clone the repository and install jupyter capabilities.

```
$ git clone https://github.com/robinhood-unofficial/pyrh.git
$ cd pyrh
$ python --version # python 3.3+ for venv functionality
Python 3.7.6
$ python -m venv pyrh_env
$ source pyrh_env/bin/activate
(pyrh_env) $ pip install .[notebook]
(pyrh_env) $ cp .env.sample .env # update the values in here
(pyrh_env) $ jupyter notebook notebooks/example.ipynb
```

Now just run the files in the example.

1.4 Related

- robinhood-ruby - RubyGem for interacting with Robinhood API
- robinhood-node - NodeJS module to make trades with Robinhood Private API
- See the original blog post.

CHAPTER

TWO

DEVELOPERS

2.1 Developer Guidelines

We appreciate all types of contributions to pyrh: bug reports, documentation, and new features. The issue tracker is the best place to start if you are looking for small issues to get started with. Specifically look for those marked *Needs Contributor*.

2.1.1 Installation

- Python 3.7+ is required
- `poetry` is used to manage package dependencies
- `pre-commit` is used to manage the project's tooling and linting
  - `black`
  - `flake8`
  - `isort`
- `pyenv` / `pyenv-virtualenv` are great and highly recommended but not covered in this guide

```
$ git clone https://github.com/robinhood-unofficial/pyrh.git
$ cd Robinhood
$ brew install poetry
$ brew install pre-commit
$ python -m venv pyrh_env
$ source pyrh_env/bin/activate
(pyrh_env) $ poetry install
(pyrh_env) $ pre-commit install
```

Lint checks are automatically run when you try to push the code. To manually run them run the following command. If you want to skip the lint check, you can do that by using *–no-verify* when committing.

```
(pyrh_env) $ pre-commit run -a
(pyrh_env) $ git commit -am "Some patch commit" --no-verify # will skip lint checks
```

Running tests and viewing coverage

```
(pyrh_env) $ pytest # from the root directory
(pyrh_env) $ open htmlcov/index.html # opens coverage in browser (on macOS)
```

2.1.2 Contributing Code Changes

We operate using pull requests, please branch off of master to submit your changes. Make sure to go to GitHub fork the project and switch your remotes. Please try to update the docs and write some initial tests to aid with code review. Once you're all set please add a towncrier file of your changes to the newsfragment directory.

```
(pyrh_env) $ git remote set-url origin https://github.com/{YOUR_USER_NAME}/pyrh.git
(pyrh_env) $ git remote add upstream https://github.com/robinhood-unofficial/pyrh.git
(pyrh_env) $ git checkout -b some_changes
(pyrh_env) $ # now make and commit your changes
(pyrh_env) $ git push  --set-upstream origin some_changes
(pyrh_env) $ # now go to YOUR fork and submit a pull request upstream
```

CHAPTER

THREE

API

3.1 API Reference

Robinhood(username, password[, ...])	Wrapper class for fetching/parsing Robinhood endpoints.
load_session([path])	Load cached session parameters from a json file.
dump_session(robinhood[, path])	Save the current session parameters to a json file.
exceptions	Exceptions: custom exceptions for library

3.1.1 pyrh.Robinhood

class pyrh.**Robinhood**(*username*, *password*, *challenge_type='email'*, *headers=None*, *proxies=None*, ***kwargs*)

Wrapper class for fetching/parsing Robinhood endpoints.

Please see `SessionManager` for login functionality.

Provides a global convenience wrapper for the following manager objects:

- InstrumentManager
- TODO: Add to this list

__init__(*username*, *password*, *challenge_type='email'*, *headers=None*, *proxies=None*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

Return type `None`

Methods

__init__(username, password[, ...])	Initialize self.
adjusted_previous_close([stock])	Get adjusted previous closing price for a stock.
ask_price([stock])	Get asking price for a stock.
ask_size([stock])	Get ask size for a stock.
bid_price([stock])	Get bid price for a stock.
bid_size([stock])	Get bid size for a stock.
cancel_order(order_id)	Cancels specified order and returns the response.
dividends()	Wrapper for portfolios
fundamentals([stock])	Wrapper for get_fundamentlals function
get(url[, params, headers, raise_errors, ...])	Run a wrapped session HTTP GET request.

Continued on next page

Table 2 – continued from previous page

get_account()	Fetch account information.
get_fundamentals([stock])	Find stock fundamentals data
get_historical_quotes(stock, interval, span)	
get_news(stock)	Fetch news endpoint.
get_open_orders()	Returns all currently open (cancellable) orders.
get_option_chainid(symbol)	
get_option_marketdata(instrument)	
get_option_quote(arg_dict)	
get_options(stock, expiration_dates, option_type)	Get a list (chain) of options contracts belonging to a particular stock
get_popularity([stock])	Get the number of robinhood users who own the given stock
get_quote([stock])	Wrapper for quote_data.
get_quote_list([stock, key])	Returns multiple stock info and keys from quote_data (prompt if blank)
get_stock_marketdata(instruments)	
get_tickers_by_tag([tag])	Get a list of instruments belonging to a tag
get_url(url)	Flat wrapper for fetching URL directly/
instrument([symbol, id_])	Get a single instrument using a provided query parameter.
instruments([query])	Get a generator of instruments.
investment_profile()	Fetch investment_profile.
last_trade_price([stock])	Get last trade price for a stock
last_updated_at([stock])	Get last update datetime.
last_updated_at_datetime([stock])	Get last updated datetime.
login([force_refresh])	Login to the session.
logout()	Logout from the session.
options_owned()	
order_history([orderId])	Wrapper for portfolios
place_buy_order(instrument, quantity[, ...])	Wrapper for placing buy orders
place_limit_buy_order([instrument_URL, ...])	Wrapper for placing limit buy orders
place_limit_sell_order([instrument_URL, ...])	Wrapper for placing limit sell orders
place_market_buy_order([instrument_URL, ...])	Wrapper for placing market buy orders
place_market_sell_order([instrument_URL, ...])	Wrapper for placing market sell orders
place_order(instrument[, quantity, price, ...])	Place an order with Robinhood
place_sell_order(instrument, quantity[, ...])	Wrapper for placing sell orders
place_stop_limit_buy_order([instrument_URL, ...])	Wrapper for placing stop limit buy orders
place_stop_limit_sell_order([...])	Wrapper for placing stop limit sell orders
place_stop_loss_buy_order([instrument_URL, ...])	Wrapper for placing stop loss buy orders
place_stop_loss_sell_order([instrument_URL, ...])	Wrapper for placing stop loss sell orders
portfolio()	Returns the user's portfolio data
positions()	Returns the user's positions data
post(url[, data, headers, raise_errors, ...])	Run a wrapped session HTTP POST request.

Continued on next page

Table 2 – continued from previous page

previous_close([stock])	Get previous closing price for a stock.
previous_close_date([stock])	Get previous closing date for a stock.
print_quote([stock])	Print quote information.
print_quotes(stocks)	Print a collection of stocks.
quote_data([stock])	Fetch stock quote.
quotes_data(stocks)	Fetch quote for multiple stocks, in one single Robinhood API call.
securities_owned()	Returns list of securities' symbols that the user has shares in
submit_buy_order([instrument_URL, symbol, ...])	Submits buy order to Robinhood
submit_sell_order([instrument_URL, symbol, ...])	Submits order to Robinhood
symbol([stock])	Get symbol for a stock.
user()	

Attributes

authenticated	Check if the session is authenticated.
login_set	Check if login info is properly configured.
token_expired	Check if the issued auth token has expired.

adjusted_previous_close(*stock=''*)
Get adjusted previous closing price for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns adjusted previous closing price

Return type (float)

ask_price(*stock=''*)
Get asking price for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns ask price

Return type (float)

ask_size(*stock=''*)
Get ask size for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns ask size

Return type (int)

bid_price(*stock=''*)
Get bid price for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns bid price

Return type (float)

bid_size(*stock=''*)
Get bid size for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns bid size

Return type (int)

cancel_order(*order_id*)
Cancels specified order and returns the response.

If order cannot be cancelled, *None* is returned. (results from *orders* command).

Parameters **order_id** (*str or dict*) – Order ID string that is to be cancelled or open order dict returned from order get.

Returns result from *orders* put command

Return type (requests.request)

dividends()
Wrapper for portfolios

Returns obj: *dict*): JSON dict from getting dividends

Return type

(

fundamentals(*stock=''*)
Wrapper for get_fundamentlals function

get_account()
Fetch account information.

Returns *accounts* endpoint payload

Return type (dict)

get_fundamentals(*stock=''*)
Find stock fundamentals data

Parameters **(str)** – stock ticker

Returns contents of *fundamentals* endpoint

Return type (dict)

get_news(*stock*)
Fetch news endpoint.

Parameters **stock** (*str*) – stock ticker

Returns (dict) values returned from *news* endpoint

get_open_orders()
Returns all currently open (cancellable) orders.

If not orders are currently open, *None* is returned.

TODO: Is there a way to get these from the API endpoint without stepping through order history?

get_options(*stock*, *expiration_dates*, *option_type*)
Get a list (chain) of options contracts belonging to a particular stock

Args: stock ticker (str), list of expiration dates to filter on (YYYY-MM-DD), and whether or not its a 'put' or a 'call' option type (str).

Returns a list (chain) of contracts for a given underlying equity instrument

Return type Options Contracts (List)

get_popularity(*stock=''*)
Get the number of robinhood users who own the given stock

Parameters **stock** (*str*) – stock ticker

Returns number of users who own the stock

Return type (int)

get_quote(*stock=''*)
Wrapper for quote_data.

get_quote_list(*stock=''*, *key=''*)
Returns multiple stock info and keys from quote_data (prompt if blank)

Parameters

- **stock** (*str*) – stock ticker (or tickers separated by a comma)
- **prompt if blank** (,) –
- **key** (*str*) – key attributes that the function should return

Returns

Returns values from each stock or empty list if none of the stocks were valid

Return type (list)

get_tickers_by_tag(*tag=None*)
Get a list of instruments belonging to a tag

Args: tag - Tags may include but are not limited to:

- top-movers
- etf
- 100-most-popular
- mutual-fund
- finance
- cap-weighted
- investment-trust-or-fund

Returns a list of Ticker strings

Return type (List)

get_url(*url*)
Flat wrapper for fetching URL directly/

investment_profile()
Fetch investment_profile.

last_trade_price(*stock=''*)
Get last trade price for a stock

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns last trade price

Return type (float)

last_updated_at(*stock=''*)
Get last update datetime.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns last update datetime

Return type (str)

last_updated_at_datetime(*stock=''*)
Get last updated datetime.

Note: queries *quote* endpoint, dict wrapper *self.last_updated_at* returns time as *str* in format: 'YYYY-MM-ddTHH:mm:ss:000Z'

Parameters **stock** (*str*) – stock ticker

Returns last update datetime

Return type (datetime)

order_history(*orderId=None*)
Wrapper for portfolios

Optional Args: add an order ID to retrieve information about a single order.

Returns JSON dict from getting orders

Return type (dict)

place_buy_order(*instrument*, *quantity*, *ask_price=0.0*)
Wrapper for placing buy orders

Parameters

- **instrument** (*dict*) – the RH URL and symbol in dict for the instrument to be traded
- **quantity** (*int*) – quantity of stocks in order
- **ask_price** (*float*) – price for order (OPTIONAL! If not given, ask_price is automatic.)

Returns result from *orders* put command

Return type (requests.request)

place_limit_buy_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *price=None*, *quantity=None*)
Wrapper for placing limit buy orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **price** (*float*) – The max price you're willing to pay per share
- **quantity** (*int*) – Number of shares to buy

Returns result from *orders* put command

Return type (requests.request)

place_limit_sell_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *price=None*, *quantity=None*)
Wrapper for placing limit sell orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **price** (*float*) – The minimum price you're willing to get per share
- **quantity** (*int*) – Number of shares to sell

Returns result from *orders* put command

Return type (requests.request)

place_market_buy_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *quantity=None*)

Wrapper for placing market buy orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **quantity** (*int*) – Number of shares to buy

Returns result from *orders* put command

Return type (requests.request)

place_market_sell_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *quantity=None*)

Wrapper for placing market sell orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **quantity** (*int*) – Number of shares to sell

Returns result from *orders* put command

Return type (requests.request)

place_order(*instrument*, *quantity=1*, *price=0.0*, *transaction=None*, *trigger='immediate'*, *order='market'*, *time_in_force='gfd'*)

Place an order with Robinhood

Parameters

- **instrument** (*dict*) – the RH URL and symbol in dict for the instrument to be traded
- **quantity** (*int*) – quantity of stocks in order
- **bid_price** (*float*) – price for order
- **transaction** (Transaction) – BUY or SELL enum
- **trigger** (Trigger) – IMMEDIATE or STOP enum
- **order** (Order) – MARKET or LIMIT
- **time_in_force** (TIME_IN_FORCE) – GFD or GTC (day or until cancelled)

Returns result from *orders* put command

Return type (requests.request)

place_sell_order(*instrument*, *quantity*, *bid_price=0.0*)

Wrapper for placing sell orders

Parameters

- **instrument** (*dict*) – the RH URL and symbol in dict for the instrument to be traded
- **quantity** (*int*) – quantity of stocks in order
- **bid_price** (*float*) – price for order (OPTIONAL! If not given, bid_price is automatic.)

Returns result from *orders* put command

Return type (requests.request)

place_stop_limit_buy_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *stop_price=None*, *price=None*, *quantity=None*)

Wrapper for placing stop limit buy orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **stop_price** (*float*) – The price at which this becomes a limit order
- **price** (*float*) – The max price you're willing to pay per share
- **quantity** (*int*) – Number of shares to buy

Returns result from *orders* put command

Return type (requests.request)

place_stop_limit_sell_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *price=None*, *stop_price=None*, *quantity=None*)

Wrapper for placing stop limit sell orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **stop_price** (*float*) – The price at which this becomes a limit order
- **price** (*float*) – The max price you're willing to get per share
- **quantity** (*int*) – Number of shares to sell

Returns result from *orders* put command

Return type (requests.request)

place_stop_loss_buy_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *stop_price=None*, *quantity=None*)

Wrapper for placing stop loss buy orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **stop_price** (*float*) – The price at which this becomes a market order
- **quantity** (*int*) – Number of shares to buy

Returns result from *orders* put command

Return type (requests.request)

place_stop_loss_sell_order(*instrument_URL=None*, *symbol=None*, *time_in_force=None*, *stop_price=None*, *quantity=None*)

Wrapper for placing stop loss sell orders

Notes

If only one of the instrument_URL or symbol are passed as arguments the other will be looked up automatically.

Parameters

- **instrument_URL** (*str*) – The RH URL of the instrument
- **symbol** (*str*) – The ticker symbol of the instrument
- **time_in_force** (*str*) – 'GFD' or 'GTC' (day or until cancelled)
- **stop_price** (*float*) – The price at which this becomes a market order
- **quantity** (*int*) – Number of shares to sell

Returns result from *orders* put command

Return type (requests.request)

portfolio()
Returns the user's portfolio data

positions()
Returns the user's positions data

Returns object: *dict*): JSON dict from getting positions

Return type

(

previous_close(*stock=''*)
Get previous closing price for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns previous closing price

Return type (float)

previous_close_date(*stock=''*)
Get previous closing date for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns previous close date

Return type (str)

print_quote(*stock=''*)
Print quote information.

Parameters **stock** (*str*) – ticker to fetch

Returns None

print_quotes(*stocks*)
Print a collection of stocks.

Parameters **stocks** (list) – list of stocks to pirnt

Returns None

quote_data(*stock=''*)
Fetch stock quote.

Parameters **stock** (*str or dict*) – stock ticker symbol or stock instrument

Returns JSON contents from *quotes* endpoint

Return type (dict)

quotes_data(*stocks*)
Fetch quote for multiple stocks, in one single Robinhood API call.

Parameters **stocks** (*list<str>*) – stock tickers

Returns List of JSON contents from *quotes* endpoint, in the same order of input args. If any ticker is invalid, a None will occur at that position.

Return type (list of dict)

securities_owned()
Returns list of securities' symbols that the user has shares in

Returns object: *dict*): Non-zero positions

Return type

(

submit_buy_order(*instrument_URL=None*, *symbol=None*, *order_type=None*, *time_in_force=None*, *trigger=None*, *price=None*, *stop_price=None*, *quantity=None*, *side=None*)
Submits buy order to Robinhood

Notes

This is normally not called directly. Most programs should use one of the following instead:

place_market_buy_order() place_limit_buy_order() place_stop_loss_buy_order() place_stop_limit_buy_order() place_market_sell_order() place_limit_sell_order() place_stop_loss_sell_order() place_stop_limit_sell_order()

Parameters

- **instrument_URL** (*str*) – the RH URL for the instrument
- **symbol** (*str*) – the ticker symbol for the instrument
- **order_type** (*str*) – 'market' or 'limit'
- **time_in_force** (TIME_IN_FORCE) – 'gfd' or 'gtc' (day or until cancelled)
- **trigger** (*str*) – 'immediate' or 'stop' enum
- **price** (*float*) – The share price you'll accept
- **stop_price** (*float*) – The price at which the order becomes a market or limit order
- **quantity** (*int*) – The number of shares to buy/sell

- **side** (*str*) – BUY or sell

Returns result from *orders* put command

Return type (requests.request)

submit_sell_order(*instrument_URL=None*, *symbol=None*, *order_type=None*, *time_in_force=None*, *trigger=None*, *price=None*, *stop_price=None*, *quantity=None*, *side=None*)

Submits order to Robinhood

Notes

This is normally not called directly. Most programs should use one of the following instead:

place_market_buy_order() place_limit_buy_order() place_stop_loss_buy_order() place_stop_limit_buy_order() place_market_sell_order() place_limit_sell_order() place_stop_loss_sell_order() place_stop_limit_sell_order()

Parameters

- **instrument_URL** (*str*) – the RH URL for the instrument
- **symbol** (*str*) – the ticker symbol for the instrument
- **order_type** (*str*) – 'MARKET' or 'LIMIT'
- **time_in_force** (TIME_IN_FORCE) – GFD or GTC (day or until cancelled)
- **trigger** (*str*) – IMMEDIATE or STOP enum
- **price** (*float*) – The share price you'll accept
- **stop_price** (*float*) – The price at which the order becomes a market or limit order
- **quantity** (*int*) – The number of shares to buy/sell
- **side** (*str*) – BUY or sell

Returns result from *orders* put command

Return type (requests.request)

symbol(*stock=''*)

Get symbol for a stock.

Note: queries *quote* endpoint, dict wrapper

Parameters **stock** (*str*) – stock ticker

Returns stock symbol

Return type (str)

3.1.2 pyrh.load_session

pyrh.**load_session**(*path=None*)
Load cached session parameters from a json file.

Note: This function defaults to caching this information to ~/.robinhood/login.json

Parameters **path** (Union[Path, str, None]) – The location and file name to load from.

Return type Robinhood

Returns A configured instance of SessionManager.

Raises ***InvalidCacheFile*** – If the cache file cannot be decoded or does not exist

3.1.3 pyrh.dump_session

pyrh.**dump_session**(*robinhood*, *path=None*)
Save the current session parameters to a json file.

Note: This function defaults to caching this information to ~/.robinhood/login.json

Parameters

- **robinhood** (Robinhood) – A Robinhood instance.
- **path** (Union[Path, str, None]) – The location to save the file and its name.

Return type None

3.1.4 pyrh.exceptions

Exceptions: custom exceptions for library

Exceptions

AuthenticationError	Error when trying to login to robinhood.
InvalidCacheFile	Error when the cache config file is found to be invalid.
InvalidOperation	An invalid operation was requsted to be performed.
InvalidOptionId	When an invalid option id is given/
InvalidTickerSymbol	When an invalid ticker (stock symbol) is given/
PyrhException	Wrapper for custom robinhood library exceptions.
PyrhValueError	Value Error for the pyrh library.

exception pyrh.exceptions.**AuthenticationError**
Error when trying to login to robinhood.

exception pyrh.exceptions.**InvalidCacheFile**
Error when the cache config file is found to be invalid.

exception pyrh.exceptions.**InvalidOperation**

An invalid operation was requsted to be performed.

exception pyrh.exceptions.**InvalidOptionId**
When an invalid option id is given/

exception pyrh.exceptions.**InvalidTickerSymbol**
When an invalid ticker (stock symbol) is given/

exception pyrh.exceptions.**PyrhException**
Wrapper for custom robinhood library exceptions.

exception pyrh.exceptions.**PyrhValueError**
Value Error for the pyrh library.

CHAPTER

FOUR

CHANGELOG

4.1 Changelog

4.1.1 pyrh 2.0 (2020-03-18)

- Fixed 2fa connection issues
- Last version to support python 2

4.1.2 pyrh 1.0.1 (2017-11-07)

- Added custom exception

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

pyrh.exceptions, 20

INDEX

Symbols

__init__() (*pyrh.Robinhood method*), 7

A

adjusted_previous_close() (*pyrh.Robinhood method*), 9
ask_price() (*pyrh.Robinhood method*), 9
ask_size() (*pyrh.Robinhood method*), 9
AuthenticationError, 20

B

bid_price() (*pyrh.Robinhood method*), 10
bid_size() (*pyrh.Robinhood method*), 10

C

cancel_order() (*pyrh.Robinhood method*), 10

D

dividends() (*pyrh.Robinhood method*), 10
dump_session() (*in module pyrh*), 20

F

fundamentals() (*pyrh.Robinhood method*), 10

G

get_account() (*pyrh.Robinhood method*), 10
get_fundamentals() (*pyrh.Robinhood method*), 10
get_news() (*pyrh.Robinhood method*), 11
get_open_orders() (*pyrh.Robinhood method*), 11
get_options() (*pyrh.Robinhood method*), 11
get_popularity() (*pyrh.Robinhood method*), 11
get_quote() (*pyrh.Robinhood method*), 11
get_quote_list() (*pyrh.Robinhood method*), 11
get_tickers_by_tag() (*pyrh.Robinhood method*), 11
get_url() (*pyrh.Robinhood method*), 12

I

InvalidCacheFile, 20
InvalidOperation, 20
InvalidOptionId, 21
InvalidTickerSymbol, 21
investment_profile() (*pyrh.Robinhood method*), 12

L

last_trade_price() (*pyrh.Robinhood method*), 12
last_updated_at() (*pyrh.Robinhood method*), 12
last_updated_at_datetime() (*pyrh.Robinhood method*), 12
load_session() (*in module pyrh*), 20

O

order_history() (*pyrh.Robinhood method*), 13

P

place_buy_order() (*pyrh.Robinhood method*), 13
place_limit_buy_order() (*pyrh.Robinhood method*), 13
place_limit_sell_order() (*pyrh.Robinhood method*), 13
place_market_buy_order() (*pyrh.Robinhood method*), 14
place_market_sell_order() (*pyrh.Robinhood method*), 14
place_order() (*pyrh.Robinhood method*), 15
place_sell_order() (*pyrh.Robinhood method*), 15
place_stop_limit_buy_order() (*pyrh.Robinhood method*), 15
place_stop_limit_sell_order() (*pyrh.Robinhood method*), 16
place_stop_loss_buy_order() (*pyrh.Robinhood method*), 16
place_stop_loss_sell_order() (*pyrh.Robinhood method*), 16
portfolio() (*pyrh.Robinhood method*), 17
positions() (*pyrh.Robinhood method*), 17
previous_close() (*pyrh.Robinhood method*), 17
previous_close_date() (*pyrh.Robinhood method*), 17
print_quote() (*pyrh.Robinhood method*), 17
print_quotes() (*pyrh.Robinhood method*), 17
pyrh.exceptions (*module*), 20

PyrhException, 21
PyrhValueError, 21

Q

quote_data() (*pyrh.Robinhood method*), 18
quotes_data() (*pyrh.Robinhood method*), 18

R

Robinhood (*class in pyrh*), 7

S

securities_owned() (*pyrh.Robinhood method*), 18
submit_buy_order() (*pyrh.Robinhood method*), 18
submit_sell_order() (*pyrh.Robinhood method*), 19
symbol() (*pyrh.Robinhood method*), 19