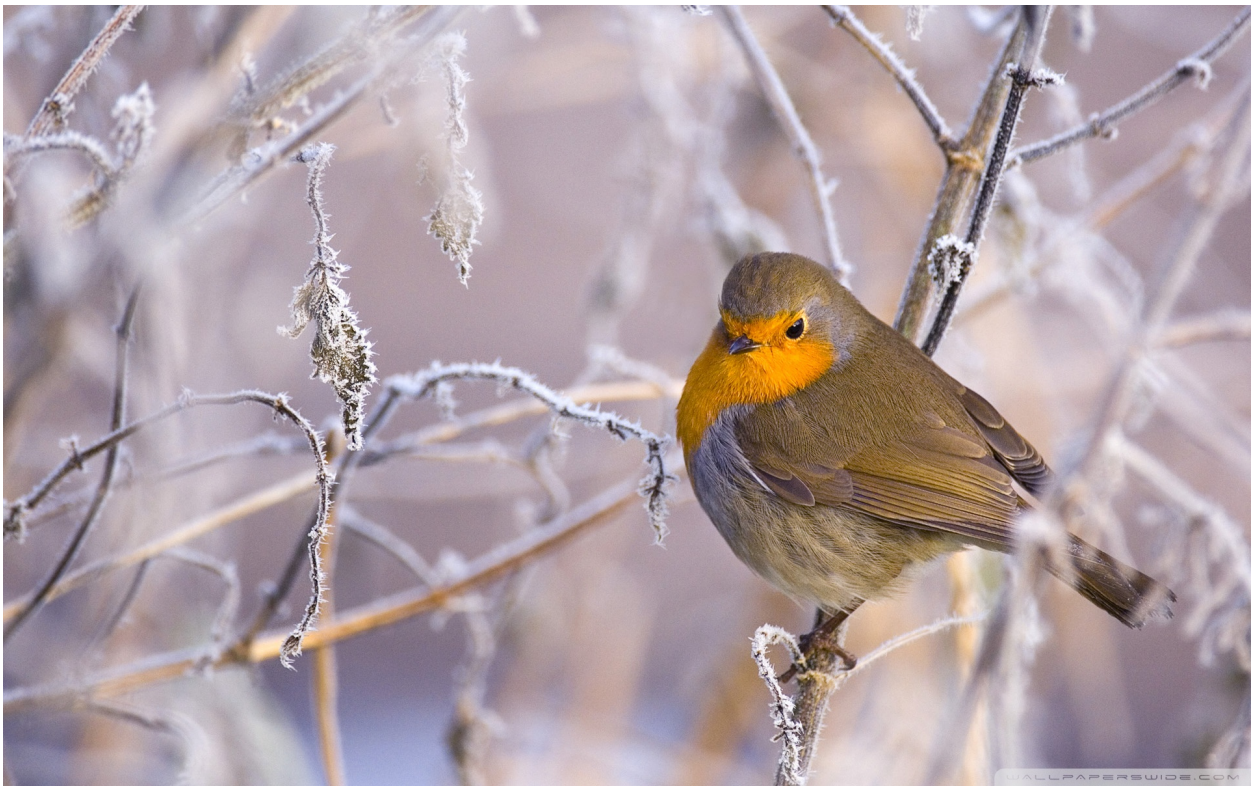
robin_stocks Documentation

Release 1.0.0

Joshua M Fernandes

Mar 22, 2019

Contents



This library aims to create simple to use functions to interact with the Robinhood API. This is a pure python interface and it requires Python 3. The purpose of this library is to allow people to make their own robo-investors or to view stock information in real time.

Note: These functions make real time calls to your Robinhood account. Unlike in the app, there are no warnings when you are about to buy, sell, or cancel an order. It is up to **YOU** to use these commands responsibly.

CHAPTER 1

User Guide

Below is the table of contents for Robin Stocks. Use it to find example code or to scroll through the list of all the callable functions.

1.1 Introduction

1.1.1 Philosophy

I've written the code in accordance with what I consider the best coding practices. Some of these are part of PEP 20 standards and some are my own. They are as follows:

- Explicit is better than implicit

This is the reason why my code uses `import robin_stocks.module as module` instead of `from module import *`. This means that calls to a function from the module must be written as `module.function` instead of the simply `function`. When viewing the code, it's easy to see which functions come from which modules.

- Flat is better than nested

The __init__.py file contains an import of all the functions I want to be made public to the user. This allows the user to call `robin_stocks.function` for all functions. Without the imports, the user would have to call `robin_stocks.module.function` and be sure to use the correct module name every single time. This may seem contradictory to the first standard, but the difference is that whereas I the developer must make explicit calls, for the end user it is unnecessary.

- Three strikes and you refactor

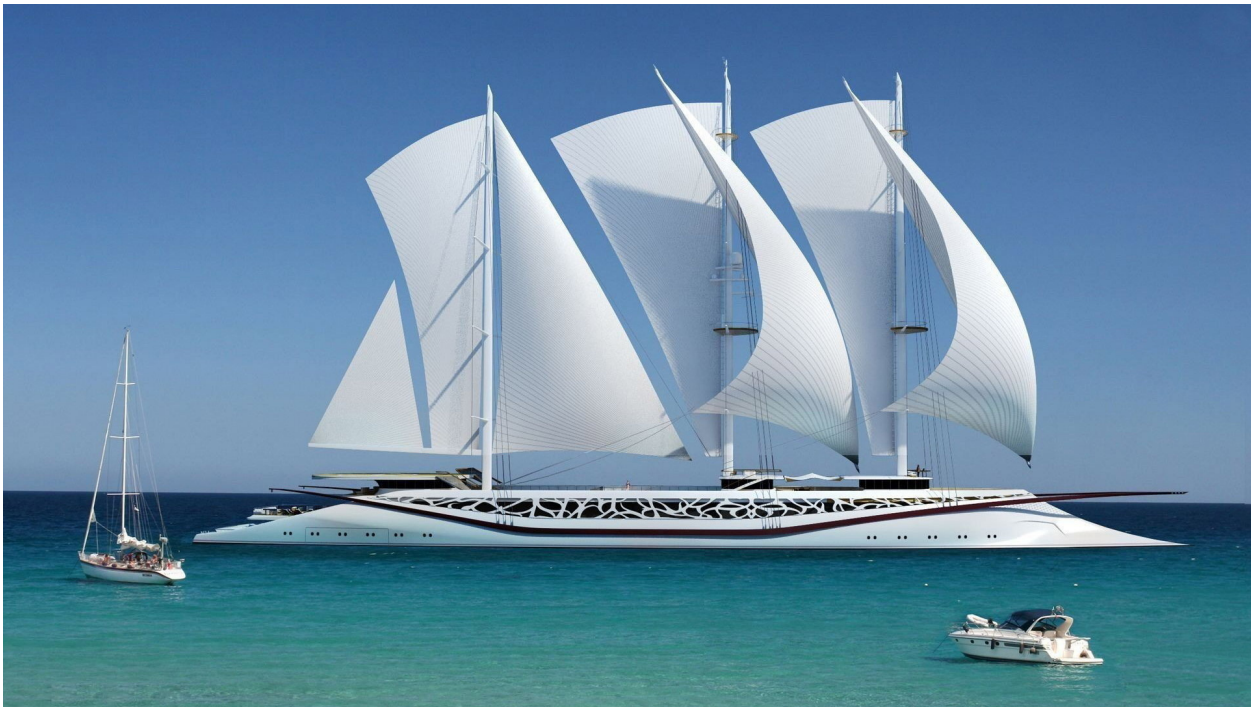
If you find yourself copying and pasting the same code 3 or more times, then it means you should put that code in its own function. Thus, I created the *`robin_stocks.helper.request_get()`* function, and then provided input parameters to handle different use cases.

- Type is in the name

A person should be able to look at the code and know the purpose of all the names they see. For this reason I have written names of functions as `snake_case`, names of input parameters and local function variables as `camelCase`, and names of global variables, class names, and enum names as `PascalCase`

1.1.2 License

1.2 Installing



1.2.1 Using Pip

This is the simplest method. To install Robin Stocks globally or inside a virtual environment, open terminal and run the command:

```
$ pip install robin_stocks
```

1.2.2 Get The Source Code

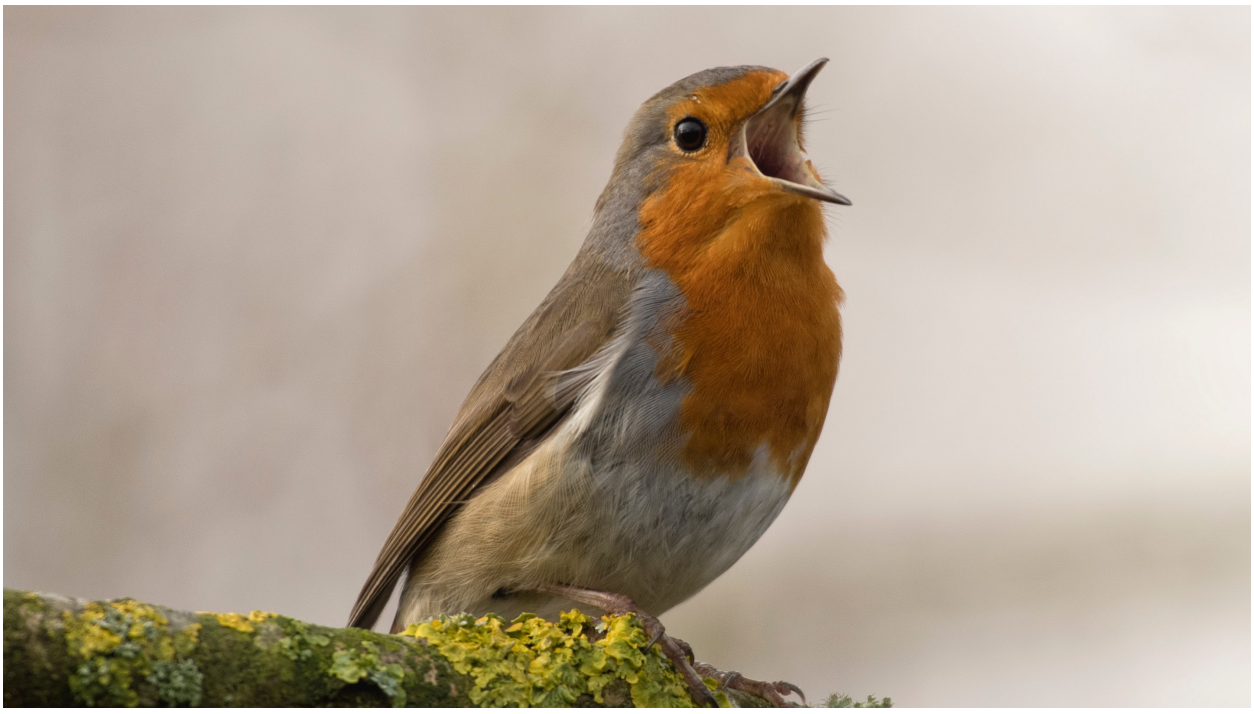
If you prefer to install the source code directly, it can be found here, or you can clone the repository using:

```
$ git clone https://github.com/jmfernandes/robin_stocks.git
```

Once the file has been downloaded or cloned, cd into the directory that contains the `setup.py` file and install using:

```
$ pip install .
```

1.3 Quick Start



1.3.1 Importing and Logging In

The first thing you will need to do is to import Robin Stocks by typing:

```
>>> import robin_stocks
```

robin_stocks will need to added as a preface to every function call in the form of `robin_stocks.function`. If you don't want to have to type robin_stocks at the beginning of every call, then import Robin Stocks by typing

```
>>> from robin_stocks import *
```

Keep in mind that this method is not considered good practice as it obfuscates the distinction between Robin Stocks' functions and other functions. For the rest of the documentation, I will assume that Robin Stocks was imported as `import robin_stocks`.

Once you have imported Robin Stocks, you will need to login in order to store an authentication token using

```
>>> robin_stocks.login(<username>,<password>)
```

Not all functions require authentication, but its good practice to log in to Robinhood at the beginning of your script.

1.3.2 Building Profile and User Data

The two most useful functions are `build_holdings` and `build_user_profile`. These condense information from several functions into a single dictionary. If you wanted to view all your holdings then type:

```
>>> my_stocks = robin_stocks.build_holdings()
>>> for key,value in my_stocks.items():
>>>     print(key,value)
```

1.3.3 Buying and Selling Stock

Buying and selling stocks is one of the most powerful features of Robin Stocks. For example, if you wanted to buy 10 shares of Apple, you would type

```
>>> robin_stocks.order_buy_market('AAPL',10)
```

and if you wanted to sell half your Tesla stock if it fell to 200.00 you would type

```
>>> positions_data = robin_stocks.get_current_positions()
>>> TSLAData = [item for item in positions_data if
>>>             robin_stocks.get_name_by_url(item['instrument']) == robin_stocks.get_
↪name_by_symbol('TSLA')][0]
>>> sellQuantity = float(TSLAData['quantity'])//2.0
>>> robin_stocks.order_sell_limit('TSLA',sellQuantity,200.00)
```

Also be aware that all the order functions default to 'gtc' or 'good until cancelled'. To change this, pass one of the following in as the last parameter in the function: 'gfd'(good for the day), 'ioc'(immediate or cancel), or 'opg'(execute at opening).

1.3.4 Finding Options

Manually clicking on stocks and viewing available options can be a chore. Especially, when you also want to view additional information like the greeks. Robin Stocks gives you the ability to view all the options for a specific expiration date by typing

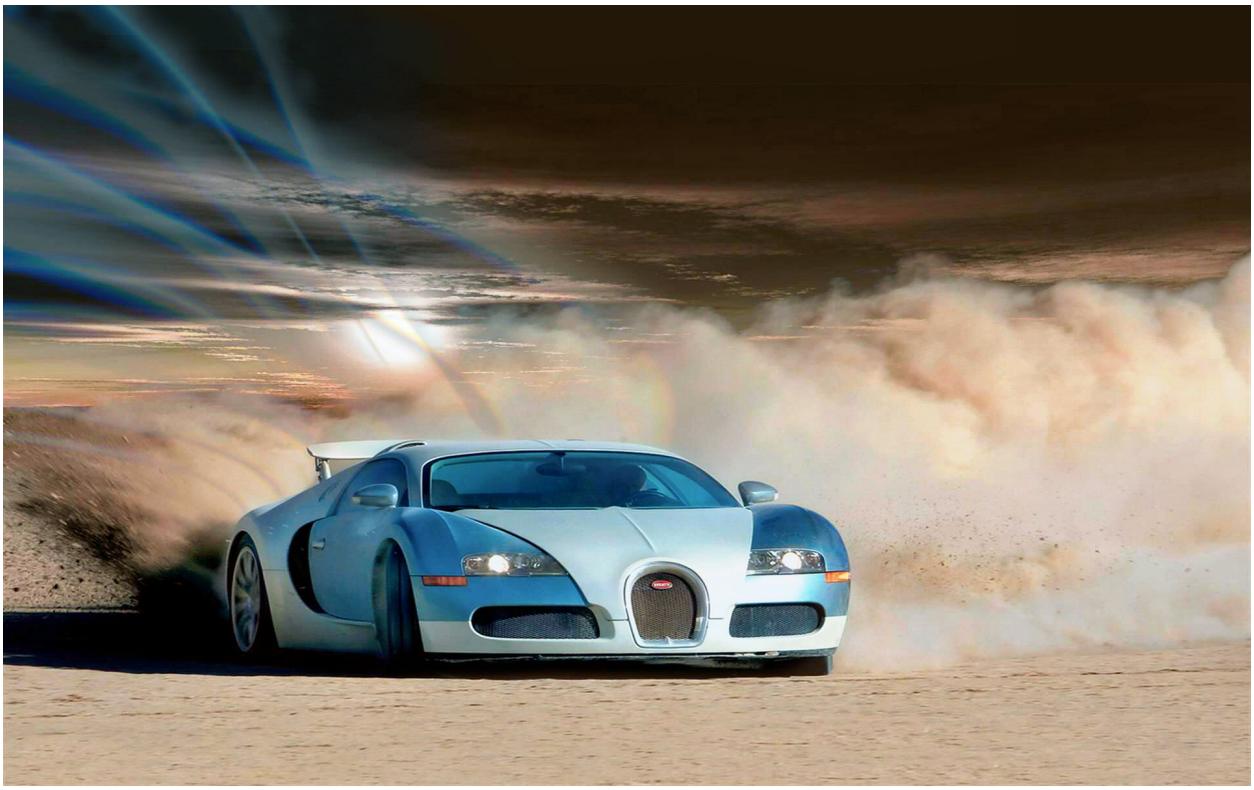
```
>>> optionData = robin_stocks.find_options_for_list_of_stocks_by_expiration_date(['fb
↪','aapl','tsla','nflx'],
>>>              expirationDate='2018-11-16',optionType='call')
```

(continues on next page)

(continued from previous page)

```
>>> for item in optionData:
>>>     print(' price -',item['strike_price'],' exp - ',item['expiration_date'],'␣
↪symbol - ',
>>>           item['chain_symbol'],' delta - ',item['delta'],' theta - ',item['theta
↪'])
```

1.4 Advanced Usage



1.4.1 Making Custom Get and Post Requests

Robin Stocks depends on Requests which you are free to call and use yourself, or you could use it within the Robin Stocks framework by using *robin_stocks.helper.request_get()*, *robin_stocks.helper.request_post()*, *robin_stocks.helper.request_document()*, and *robin_stocks.helper.request_delete()*. For example, if you wanted to make your own get request to the option instruments API endpoint in order to get all calls you would type:

```
>>> url = 'https://api.robinhood.com/options/instruments/'
>>> payload = { 'type' : 'call'}
>>> robin_stocks.request_get(url,'regular',payload)
```

Robinhood returns most data in the form:

```
{ 'previous' : None, 'results' : [], 'next' : None}
```

where 'results' is either a dictionary or a list of dictionaries. However, sometimes Robinhood returns the data in a different format. To compensate for this, I added the **dataType** parameter which defaults to return the entire dictionary listed above. There are four possible values for **dataType** and their uses are:

```
>>> robin_stocks.request_get(url,'regular')     # For when you want
>>>                                             # the whole dictionary
>>>                                             # to view 'next' or
>>>                                             # 'previous' values.
>>>
>>> robin_stocks.request_get(url,'results')     # For when results contains a
>>>                                             # list or single dictionary.
>>>
>>> robin_stocks.request_get(url,'pagination')  # For when results contains a
>>>                                             # list, but you also want to
>>>                                             # append any information in
>>>                                             # 'next' to the list.
>>>
>>> robin_stocks.request_get(url,'indexzero')   # For when results is a list
>>>                                             # of only one entry.
```

Also keep in mind that the results from the Robinhood API have been decoded using `.json()`. There are instances where the user does not want to decode the results (such as retrieving documents), so I added the *robin_stocks.helper.request_document()* function, which will always return the raw data, so there is no **dataType** parameter. *robin_stocks.helper.request_post()* is similar in that it only takes a url and payload parameter.

1.5 List of All Functions

Note: Even though the functions are written as `robin_stocks.module.function`, the module name is unimportant when calling a function. Simply use `robin_stocks.function` for all functions.

1.5.1 Sending Requests to API

---

robin_stocks.helper.**request_get**(*url*, *dataType='regular'*, *payload=None*)

For a given url and payload, makes a get request and returns the data.

Parameters

- **url** (*str*) – The url to send a get request to.
- **dataType** (*Optional[str]*) – Determines how far to drill down into the data. 'regular' returns the unfiltered data. 'results' will return data['results']. 'pagination' will return data['results'] and append it with any data that is in data['next']. 'indexzero' will return data['results'][0].
- **payload** (*Optional[dict]*) – Dictionary of parameters to pass to the url as url/?key1=value1&key2=value2.

Returns Returns the data from the get request.

robin_stocks.helper.**request_post**(*url*, *payload=None*, *timeout=16*)
For a given url and payload, makes a post request and returns the response.

Parameters

- **url** (*str*) – The url to send a post request to.
- **payload** (*Optional[dict]*) – Dictionary of parameters to pass to the url as url/?key1=value1&key2=value2.
- **timeout** (*Optional[int]*) – The time for the post to wait for a response. Should be slightly greater than multiples of 3.

Returns Returns the data from the post request.

robin_stocks.helper.**request_delete**(*url*)
For a given url and payload, makes a delete request and returns the response.

Parameters **url** (*str*) – The url to send a delete request to.

Returns Returns the data from the delete request.

robin_stocks.helper.**request_document**(*url*, *payload=None*)
Using a document url, makes a get request and returnes the session data.

Parameters **url** (*str*) – The url to send a get request to.

Returns Returns the session.get() data as opppose to session.get().json() data.

1.5.2 Logging In and Out

---

robin_stocks.authentication.**login**(*username*, *password*, *expiresIn=86400*, *scope='internal'*)
This function will effectivly log the user into robinhood by getting an authentication token and saving it to the session header.

Parameters

- **username** (*str*) – The username for your robinhood account. Usually your email.
- **password** (*str*) – The password for your robinhood account.
- **expiresIn** (*Optional[int]*) – The time until your login session expires. This is in seconds.
- **scope** (*Optional[str]*) – Specifies the scope of the authentication.

Returns A dictionary with log in information.

1.5.3 Loading Profiles

1.5.4 Getting Stock Information

---

robin_stocks.stocks.**find_instrument_data**(*query*)
Will search for stocks that contain the query keyword and return the instrument data.

Parameters **query** (*str*) – The keyword to search for.

Returns Returns a list of dictionaries that contain the instrument data for each stock that matches the query.

robin_stocks.stocks.**get_earnings**(*symbol*, *info=None*)

Returns the earnings for the differenct financial quarters.

Parameters

- **symbol** (*str*) – The stock ticker.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.stocks.**get_events**(*symbol*, *info=None*)

Returns the events related to a stock.

Parameters

- **symbol** (*str*) – The stock ticker.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns If the info parameter is provided, then the function will extract the value of the key that matches the info parameter. Otherwise, the whole dictionary is returned.

robin_stocks.stocks.**get_fundamentals**(*inputSymbols*, *info=None*)

Takes any number of stock tickers and returns fundamental information about the stock such as what sector it is in, a description of the company, dividend yield, and market cap.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **info** (*Optional[str]*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns If info parameter is left as None then the list will contain a dictionary of key/value pairs for each ticker. Otherwise, it will be a list of strings where the strings are the values of the key that corresponds to info.

robin_stocks.stocks.**get_historicals**(*inputSymbols*, *span='week'*, *bounds='regular'*)

Represents the data that is used to make the graphs.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **span** (*Optional[str]*) – Sets the range of the data to be either 'day', 'week', 'year', or '5year'. Default is 'week'.
- **bounds** (*Optional[str]*) – Represents if graph will include extended trading hours or just regular trading hours. Values are 'extended' or 'regular'.

Returns Returns a list of dictionaries where each dictionary is for a different time. If multiple stocks are provided the historical data is listed one after another.

robin_stocks.stocks.**get_instrument_by_url**(*url*, *info=None*)

Takes a single url for the stock. Should be located at `https://api.robinhood.com/instruments/<id>` where <id> is the id of the stock.

Parameters

- **url** (*str*) – The url of the stock. Can be found in several locations including in the dictionary returned from get_instruments_by_symbols(inputSymbols,info=None)
- **info** (*Optional[str]*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns If info parameter is left as None then the list will contain a dictionary of key/value pairs for each ticker. Otherwise, it will be a list of strings where the strings are the values of the key that corresponds to info.

robin_stocks.stocks.**get_instruments_by_symbols**(*inputSymbols*, *info=None*)

Takes any number of stock tickers and returns information held by the market such as ticker name, bloomberg id, and listing date.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **info** (*Optional[str]*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns If info parameter is left as None then the list will contain a dictionary of key/value pairs for each ticker. Otherwise, it will be a list of strings where the strings are the values of the key that corresponds to info.

robin_stocks.stocks.**get_latest_price**(*inputSymbols*)

Takes any number of stock tickers and returns the latest price of each one as a string.

Parameters **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.

Returns A list of prices as strings.

robin_stocks.stocks.**get_name_by_symbol**(*symbol*)

Returns the name of a stock from the stock ticker.

Parameters **symbol** (*str*) – The ticker of the stock as a string.

Returns Returns the simple name of the stock. If the simple name does not exist then returns the full name.

robin_stocks.stocks.**get_name_by_url**(*url*)

Returns the name of a stock from the instrument url. Should be located at `https://api.robinhood.com/instruments/<id>` where <id> is the id of the stock.

Parameters **symbol** (*str*) – The url of the stock as a string.

Returns Returns the simple name of the stock. If the simple name does not exist then returns the full name.

robin_stocks.stocks.**get_news**(*symbol*, *info=None*)

Returns news stories for a stock.

Parameters

- **symbol** (*str*) – The stock ticker.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.stocks.**get_popularity**(*symbol*, *info=None*)

Returns the number of open positions.

Parameters

- **symbol** (*str*) – The stock ticker.
- **info** (*Optional[str]*) – Will filter the results to be a string value.

Returns If the info parameter is provided, then the function will extract the value of the key that matches the info parameter. Otherwise, the whole dictionary is returned.

robin_stocks.stocks.**get_quotes**(*inputSymbols*, *info=None*)
Takes any number of stock tickers and returns information pertaining to its price.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **info** (*Optional[str]*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns If info parameter is left as None then the list will contain a dictionary of key/value pairs for each ticker. Otherwise, it will be a list of strings where the strings are the values of the key that corresponds to info.

robin_stocks.stocks.**get_ratings**(*symbol*, *info=None*)
Returns the ratings for a stock, including the number of buy, hold, and sell ratings.

Parameters

- **symbol** (*str*) – The stock ticker.
- **info** (*Optional[str]*) – Will filter the results to contain a dictionary of values that correspond to the key that matches info. Possible values are summary, ratings, and instrument_id

Returns If info parameter is left as None then the list will contain a dictionary of key/value pairs for each ticker. Otherwise, it will contain the values that correspond to the keyword that matches info. In this case, the value will also be a dictionary.

robin_stocks.stocks.**get_splits**(*symbol*, *info=None*)
Returns the date, divisor, and multiplier for when a stock split occureed.

Parameters

- **symbol** (*str*) – The stock ticker.
- **info** (*Optional[str]*) – Will filter the results to get a specific value. Possible options are url, instrument, execution_date, divsor, and multiplier.

Returns Returns a list of dictionaries. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

1.5.5 Getting Option Information

robin_stocks.options.**find_options_for_list_of_stocks_by_expiration_date**(*inputSymbols*, *expirationDate*, *optionType='both'*, *info=None*)

Returns a list of all the option orders that match the seach parameters

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **optionType** (*Optional[str]*) – Can be either 'call' or 'put' or leave blank to get both.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all options of the stock that match the search parameters. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**find_options_for_stock_by_expiration**(*symbol*, *expirationDate*, *optionType='both'*, *info=None*)

Returns a list of all the option orders that match the seach parameters

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **optionType** (*Optional[str]*) – Can be either 'call' or 'put' or leave blank to get both.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all options of the stock that match the search parameters. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**find_options_for_stock_by_expiration_and_strike**(*symbol*, *expirationDate*, *strike*, *optionType='both'*, *info=None*)

Returns a list of all the option orders that match the seach parameters

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strike** (*str*) – Represents the price of the option.

- **optionType** (*Optional[str]*) – Can be either 'call' or 'put' or leave blank to get both.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all options of the stock that match the search parameters. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**find_options_for_stock_by_strike**(*symbol*, *strike*, *optionType='both'*, *info=None*)

Returns a list of all the option orders that match the seach parameters

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **strike** (*str*) – Represents the price of the option.
- **optionType** (*Optional[str]*) – Can be either 'call' or 'put' or leave blank to get both.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all options of the stock that match the search parameters. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**find_tradable_options_for_stock**(*symbol*, *optionType='both'*, *info=None*)

Returns a list of all available options for a stock.

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **optionType** (*Optional[str]*) – Can be either 'call' or 'put' or left blank to get both.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all calls of the stock. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**get_chains**(*symbol*, *info=None*)

Returns the chain information of an option.

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the option. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**get_list_market_data**(*inputSymbols*, *expirationDate*, *info=None*)

Returns a list of option market data for several stock tickers.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all stock option market data. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**get_list_options_of_specific_profitability**(*inputSymbols*, *expirationDate*, *typeProfit='chance_of_profit_short'*, *profitFloor=0.0*, *profitCeiling=1.0*, *info=None*)

Returns a list of option market data for several stock tickers that match a range of profitability.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **typeProfit** (*str*) – Will either be "chance_of_profit_short" or "chance_of_profit_long".
- **profitFloor** (*int*) – The lower percentage on scale 0 to 1.
- **profitCeiling** (*int*) – The higher percentage on scale 0 to 1.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for all stock option market data. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.options.**get_option_historicals**(*symbol*, *expirationDate*, *strike*, *optionType*, *span='week'*)

Returns the data that is used to make the graphs.

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strike** (*str*) – Represents the price of the option.
- **optionType** (*str*) – Can be either 'call' or 'put'.
- **span** (*Optional[str]*) – Sets the range of the data to be either 'day', 'week', 'year', or '5year'. Default is 'week'.

Returns Returns a list that contains a list for each symbol. Each list contains a dictionary where each dictionary is for a different time.

robin_stocks.options.**get_option_instrument_data**(*symbol*, *expirationDate*, *strike*, *optionType*, *info=None*)

Returns the option instrument data for the stock option.

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strike** (*str*) – Represents the price of the option.

- **optionType** (*str*) – Can be either 'call' or 'put'.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the stock. If info parameter is provided, the value of the key that matches info is extracted.

robin_stocks.options.**get_option_instrument_data_by_id**(*id*, *info=None*)
Returns the option instrument information.

Parameters

- **id** (*str*) – The id of the stock.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the stock. If info parameter is provided, the value of the key that matches info is extracted.

robin_stocks.options.**get_option_market_data**(*symbol*, *expirationDate*, *strike*, *optionType*, *info=None*)
Returns the option market data for the stock option, including the greeks, open interest, change of profit, and adjusted mark price.

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strike** (*str*) – Represents the price of the option.
- **optionType** (*str*) – Can be either 'call' or 'put'.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the stock. If info parameter is provided, the value of the key that matches info is extracted.

robin_stocks.options.**get_option_market_data_by_id**(*id*, *info=None*)
Returns the option market data for a stock, including the greeks, open interest, change of profit, and adjusted mark price.

Parameters

- **id** (*str*) – The id of the stock.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the stock. If info parameter is provided, the value of the key that matches info is extracted.

1.5.6 Getting Market Information

---

robin_stocks.markets.**get_currency_pairs**(*info=None*)
Returns currency pairs

Parameters **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for each currency pair. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.markets.**get_markets**(*info=None*)
Returns a list of available markets.

Parameters **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for each market. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

robin_stocks.markets.**get_top_movers**(*direction*, *info=None*)
Returns a list of the top movers up or down for the day.

Parameters

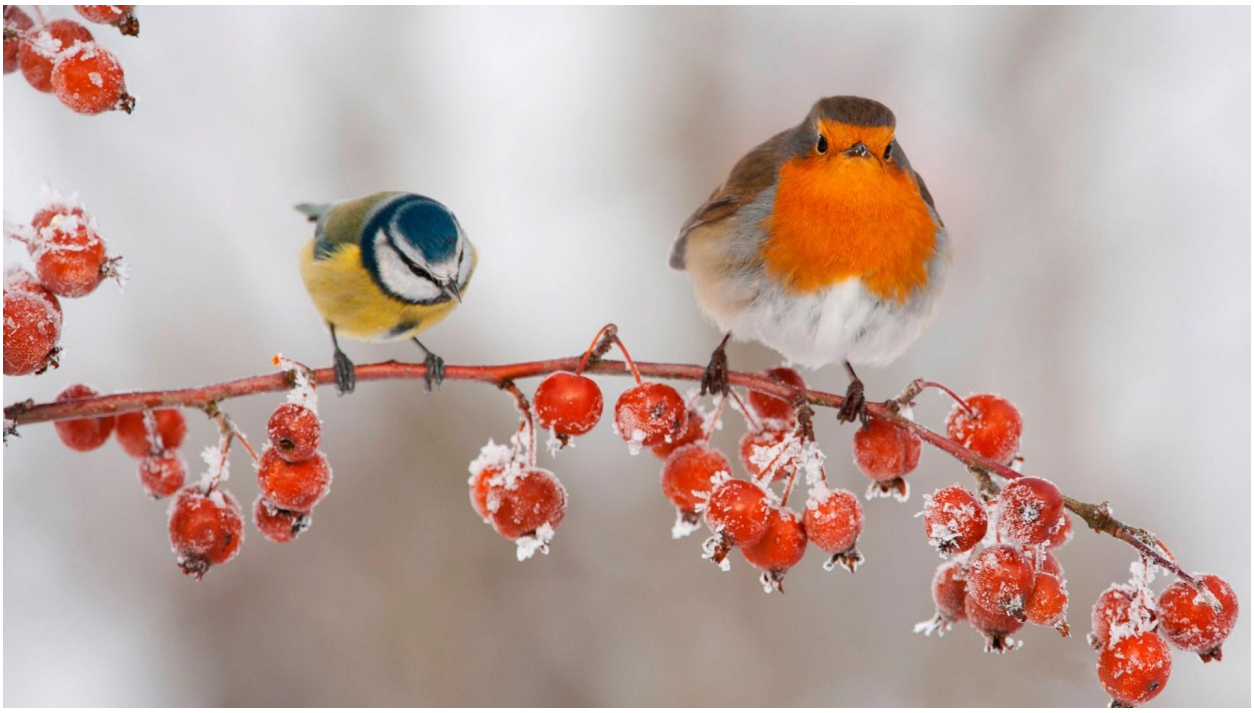
- **direction** (*str*) – The direction of movement either 'up' or 'down'
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries of key/value pairs for each mover. If info parameter is provided, a list of strings is returned where the strings are the value of the key that matches info.

1.5.7 Getting Positions and Account Information

1.5.8 Placing and Cancelling Orders

1.6 Example Scripts



Example python scripts can be found at https://github.com/jmfernandes/robin_stocks

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index