robin_stocks Documentation

Release 1.0.0

Joshua M Fernandes

Oct 10, 2020

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

When writing code for this project, you want other developers to be able to follow all function calls. A lot of times in C++ it can be confusing when trying to figure out if a function is built-in, defined in the same file, defined in another object, or an alias for another function. In Python, it's a lot easier to see where a function comes from, but care must still be taken to make code as readable as possible. 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. However users do not have to explicity call functions because of the following reason…

- 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. As an example of this, I created the `robin_stocks.helper.request_get()` function, and then provided input parameters to handle different use cases. This means that although functions I write may have very different logic for how they handle the get requests from Robinhood, none of this logic is contained in the functions themselves. It's all been abstracted away to a single function which means the code is easier to debug, easier to propagate changes, and easier to read.

- 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`, the names of input parameters and local function variables as `camelCase`, the names of class names and enum names as `PascalCase`, and the names of global variables as `UPPER_SNAKE_CASE`.

In addition, the naming of each function is standardized in order to make searching for functions easier. Functions that load user account information begin with "load", functions that place orders begin with "order", functions that cancel orders begin with "cancel", functions that query begin with "find", and so on. If you are using a text editor/IDE with auto-complete (which I highly recommend!), then this naming convention makes it even easier to find the function you want. As long as you know what you want the function to do, then you know what word it starts with.

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.

Basic

```
>>> import robin_stocks as r
>>> login = r.login(<username>,<password>)
```

You will be prompted for your MFA token if you have MFA enabled and choose to do the above basic example.

With MFA entered programmatically from Time-based One-Time Password (TOTP)

NOTE: to use this feature, you will have to sign into your robinhood account and turn on two factor authentication. Robinhood will ask you which two factor authorization app you want to use. Select "other". Robinhood will present you with an alphanumeric code. This code is what you will use for "My2factorAppHere" in the code below. Run the following code and put the resulting MFA code into the prompt on your robinhood app.

```
>>> import pyotp
>>> totp  = pyotp.TOTP("My2factorAppHere").now()
>>> print("Current OTP:", totp)
```

Once you have entered the above MFA code (the totp variable that is printed out) into your Robinhood account, it will give you a backup code. Make sure you do not lose this code or you may be locked out of your account!!! You can also take the exact same "My2factorAppHere" from above and enter it into your phone's authentication app, such as Google Authenticator. This will cause the exact same MFA code to be generated on your phone as well as your python code. This is important to do if you plan on being away from your computer and need to access your Robinhood account from your phone.

Now you should be able to login with the following code,

```
>>> import pyotp
>>> import robin_stocks as r
>>> totp  = pyotp.TOTP("My2factorAppHere").now()
>>> login = r.login('joshsmith@email.com','password', mfa_code=totp)
```

Not all of the functions contained in the module need the user to be authenticated. A lot of the functions contained in the modules 'stocks' and 'options' do not require authentication, but it's still good practice to log into Robinhood at the start of each 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 stock 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

Trading stocks, options, and crypto-currencies is one of the most powerful features of Robin Stocks. There is the ability to submit market orders, limit orders, and stop orders as long as Robinhood supports it. Here is a list of possible trades you can make

```
>>> #Buy 10 shares of Apple at market price
>>> robin_stocks.order_buy_market('AAPL',10)
>>> #Sell half a Bitcoin is price reaches 10,000
>>> robin_stocks.order_sell_crypto_limit('BTC',0.5,10000)
>>> #Buy $500 worth of Bitcoin
>>> robin_stocks.order_buy_crypto_by_price('BTC',500)
>>> #Buy 5 $150 May 1st, 2020 SPY puts if the price per contract is $1.00. Good until␣
↪cancelled.
>>> robin_stocks.order_buy_option_limit('open','debit',1.00,'SPY',5,'2020-05-01',150,
↪'put','gtc')
```

Now let's try a slightly more complex example. Let's say you wanted to sell half your Tesla stock if it fell to 200.00. To do this you would type

```
>>> positions_data = robin_stocks.get_current_positions()
>>> ## Note: This for loop adds the stock ticker to every order, since Robinhood
>>> ## does not provide that information in the stock orders.
```

(continues on next page)

(continued from previous page)

```
>>> ## This process is very slow since it is making a GET request for each order.
>>> for item in positions_data:
>>>     item['symbol'] = robin_stocks.get_symbol_by_url(item['instrument'])
>>> TSLAData = [item for item in positions_data if item['symbol'] == 'TSLA']
>>> 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')
>>> for item in optionData:
>>>     print(' price -',item['strike_price'],' exp - ',item['expiration_date'],'␣
↪symbol - ',
>>>           item['chain_symbol'],' delta - ',item['delta'],' theta - ',item['theta
↪'])
```

1.3.5 Working With Orders

You can also view all orders you have made. This includes filled orders, cancelled orders, and open orders. Stocks, options, and cryptocurrencies are separated into three different locations. For example, let's say that you have some limit orders to buy and sell Bitcoin and those orders have yet to be filled. If you want to cancel all your limit sells, you would type

```
>>> positions_data = robin_stocks.get_all_open_crypto_orders()
>>> ## Note: Again we are adding symbol to our list of orders because Robinhood
>>> ## does not include this with the order information.
>>> for item in positions_data:
>>>    item['symbol'] = robin_stocks.get_crypto_quote_from_id(item['currency_pair_id
↪'], 'symbol')
>>> btcOrders = [item for item in positions_data if item['symbol'] == 'BTCUSD' and␣
↪item['side'] == 'sell']
>>> for item in btcOrders:
>>>    robin_stocks.cancel_crypto_order(item['id'])
```

1.3.6 Saving to CSV File

Users can also export a list of all orders to a CSV file. There is a function for stocks and options. Each function takes a directory path and an optional filename. If no filename is provided, a date stamped filename will be generated. The directory path can be either absolute or relative. To save the file in the current directory, simply pass in "." as the directory. Note that ".csv" is the only valid file extension. If it is missing it will be added, and any other file extension will be automatically changed. Below are example calls.

```
>>> # let's say that I am running code from C:/Users/josh/documents/
>>> r.export_completed_stock_orders(".") # saves at C:/Users/josh/documents/stock_
↪orders_Jun-28-2020.csv
>>> r.export_completed_option_orders("../", "toplevel") # save at C:/Users/josh/
↪toplevel.csv
```

1.3.7 Using Option Spreads

When viewing a spread in the robinhood app, it incorrectly identifies both legs as either “buy” or “sell” when closing a position. The “direction” has to reverse when you try to close a spread position.

I.e. direction=”credit” when “action”:”sell”,”effect”:”close”

in the case of a long call or put spread.

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

Contains decorator functions and functions for interacting with global data.

Functions

- request_document
- request_get

- request_post
- update_session

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

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 to filter 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. Will append the requests url as url/?key1=value1&key2=value2.
- **jsonify_data** (*bool*) – If this is true, will return requests.post().json(), otherwise will return response from requests.post().

Returns Returns the data from the get request. If jsonify_data=True and requests returns an http code other than <200> then either '[None]' or 'None' will be returned based on what the dataType parameter was set as.

robin_stocks.helper.**request_post**(*url*, *payload=None*, *timeout=16*, *json=False*, *jsonify_data=True*)

For a given url and payload, makes a post request and returns the response. Allows for responses other than 200.

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.
- **json** (*bool*) – This will set the 'content-type' parameter of the session header to 'application/json'
- **jsonify_data** (*bool*) – If this is true, will return requests.post().json(), otherwise will return response from requests.post().

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

Contains all functions for the purpose of logging in and out to Robinhood.

`robin_stocks.authentication.`**`login`**(*username=None, password=None, expiresIn=86400, scope='internal', by_sms=True, store_session=True, mfa_code=None*)

This function will effectively log the user into robinhood by getting an authentication token and saving it to the session header. By default, it will store the authentication token in a pickle file and load that value on subsequent logins.

Parameters

- **`username`** (*Optional[str]*) – The username for your robinhood account, usually your email. Not required if credentials are already cached and valid.
- **`password`** (*Optional[str]*) – The password for your robinhood account. Not required if credentials are already cached and valid.
- **`expiresIn`** (*Optional[int]*) – The time until your login session expires. This is in seconds.
- **`scope`** (*Optional[str]*) – Specifies the scope of the authentication.
- **`by_sms`** (*Optional[boolean]*) – Specifies whether to send an email(False) or an sms(True)
- **`store_session`** (*Optional[boolean]*) – Specifies whether to save the log in authorization for future log ins.
- **`mfa_code`** (*Optional[str]*) – MFA token if enabled.

Returns A dictionary with log in information. The 'access_token' keyword contains the access token, and the 'detail' keyword contains information on whether the access token was generated or loaded from pickle file.

`robin_stocks.authentication.`**`logout`**()

Removes authorization from the session header.

Returns None

1.5.3 Loading Profiles

Contains functions for getting all the information tied to a user account.

`robin_stocks.profiles.`**`load_account_profile`**(*info=None*)

Gets the information associated with the accounts profile,including day trading information and cash being held by Robinhood.

Parameters **`info`** (*Optional[str]*) – The name of the key whose value is to be returned from the function.

Returns The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- url
- portfolio_cash
- can_downgrade_to_cash

- user
- account_number
- type
- created_at
- updated_at
- deactivated
- deposit_halted
- only_position_closing_trades
- buying_power
- cash_available_for_withdrawal
- cash
- cash_held_for_orders
- uncleared_deposits
- sma
- sma_held_for_orders
- unsettled_funds
- unsettled_debit
- crypto_buying_power
- max_ach_early_access_amount
- cash_balances
- margin_balances
- sweep_enabled
- instant_eligibility
- option_level
- is_pinnacle_account
- rhs_account_number
- state
- active_subscription_id
- locked
- permanently_deactivated
- received_ach_debit_locked
- drip_enabled
- eligible_for_fractionals
- eligible_for_drip
- eligible_for_cash_management
- cash_management_enabled

- option_trading_on_expiration_enabled
- cash_held_for_options_collateral
- fractional_position_closing_only
- user_id
- rhs_stock_loan_consent_status

`robin_stocks.profiles.`**`load_basic_profile`**`(`*info=None*`)`

Gets the information associated with the personal profile, such as phone number, city, marital status, and date of birth.

Parameters **`info`** (`Optional[str]`) – The name of the key whose value is to be returned from the function.

Returns The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- user
- address
- city
- state
- zipcode
- phone_number
- marital_status
- date_of_birth
- citizenship
- country_of_residence
- number_dependents
- signup_as_rhs
- tax_id_ssn
- updated_at

`robin_stocks.profiles.`**`load_investment_profile`**`(`*info=None*`)`

Gets the information associated with the investment profile. These are the answers to the questionaire you filled out when you made your profile.

Parameters **`info`** (`Optional[str]`) – The name of the key whose value is to be returned from the function.

Returns The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- user
- total_net_worth

- annual_income
- source_of_funds
- investment_objective
- investment_experience
- liquid_net_worth
- risk_tolerance
- tax_bracket
- time_horizon
- liquidity_needs
- investment_experience_collected
- suitability_verified
- option_trading_experience
- professional_trader
- understand_option_spreads
- interested_in_options
- updated_at

`robin_stocks.profiles.`**`load_portfolio_profile`**`(`*info=None*`)`

Gets the information associated with the portfolios profile, such as withdrawable amount, market value of account, and excess margin.

Parameters **`info`** (`Optional[str]`) – The name of the key whose value is to be returned from the function.

Returns The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- url
- account
- start_date
- market_value
- equity
- extended_hours_market_value
- extended_hours_equity
- extended_hours_portfolio_equity
- last_core_market_value
- last_core_equity
- last_core_portfolio_equity
- excess_margin
- excess_maintenance

- excess_margin_with_uncleared_deposits
- excess_maintenance_with_uncleared_deposits
- equity_previous_close
- portfolio_equity_previous_close
- adjusted_equity_previous_close
- adjusted_portfolio_equity_previous_close
- withdrawable_amount
- unwithdrawable_deposits
- unwithdrawable_grants

`robin_stocks.profiles.`**`load_security_profile`**`(`*info=None*`)`

Gets the information associated with the security profile.

Parameters **`info`** (*`Optional[str]`*) – The name of the key whose value is to be returned from the function.

Returns The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- user
- object_to_disclosure
- sweep_consent
- control_person
- control_person_security_symbol
- security_affiliated_employee
- security_affiliated_firm_relationship
- security_affiliated_firm_name
- security_affiliated_person_name
- security_affiliated_address
- security_affiliated_address_subject
- security_affiliated_requires_duplicates
- stock_loan_consent_status
- agreed_to_rhs
- agreed_to_rhs_margin
- rhs_stock_loan_consent_status
- updated_at

`robin_stocks.profiles.`**`load_user_profile`**`(`*info=None*`)`

Gets the information associated with the user profile, such as username, email, and links to the urls for other profiles.

Parameters **`info`** (*`Optional[str]`*) – The name of the key whose value is to be returned from the function.

Returns The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- url
- id
- id_info
- username
- email
- email_verified
- first_name
- last_name
- origin
- profile_name
- created_at

1.5.4 Getting Stock Information

Contains information in regards to stocks.

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

Dictionary Keys

- id
- url
- quote
- fundamentals
- splits
- state
- market
- simple_name
- name
- tradeable

- tradability
- symbol
- bloomberg_unique
- margin_initial_ratio
- maintenance_ratio
- country
- day_trade_ratio
- list_date
- min_tick_size
- type
- tradable_chain_id
- rhs_tradability
- fractional_tradability
- default_collar_fraction

`robin_stocks.stocks.`**`get_earnings`**(*symbol*, *info=None*)
Returns the earnings for the different financial quarters.

Parameters

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

Returns [list] 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.

Dictionary Keys

- symbol
- instrument
- year
- quarter
- eps
- report
- call

`robin_stocks.stocks.`**`get_events`**(*symbol*, *info=None*)
Returns the events related to a stock that the user owns. For example, if you owned options for USO and that stock underwent a stock split resulting in you owning shares of newly created USO1, then that event will be returned when calling get_events('uso1')

Parameters

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

Returns [list] 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.

Dictionary Keys

- account
- cash_component
- chain_id
- created_at
- direction
- equity_components
- event_date
- id
- option
- position
- quantity
- state
- total_cash_amount
- type
- underlying_price
- updated_at

`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 [list] 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.

Dictionary Keys

- open
- high
- low
- volume
- average_volume_2_weeks
- average_volume
- high_52_weeks
- dividend_yield
- float
- low_52_weeks

- market_cap
- pb_ratio
- pe_ratio
- shares_outstanding
- description
- instrument
- ceo
- headquarters_city
- headquarters_state
- sector
- industry
- num_employees
- year_founded
- symbol

`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 [dict or str] If info parameter is left as None then will return a dictionary of key/value pairs for a specific url. Otherwise, it will be the string value of the key that corresponds to info.

Dictionary Keys

- id
- url
- quote
- fundamentals
- splits
- state
- market
- simple_name
- name
- tradeable
- tradability
- symbol
- bloomberg_unique

- margin_initial_ratio
- maintenance_ratio
- country
- day_trade_ratio
- list_date
- min_tick_size
- type
- tradable_chain_id
- rhs_tradability
- fractional_tradability
- default_collar_fraction

`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 [list] If info parameter is left as None then the list will 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.

Dictionary Keys

- id
- url
- quote
- fundamentals
- splits
- state
- market
- simple_name
- name
- tradeable
- tradability
- symbol
- bloomberg_unique
- margin_initial_ratio
- maintenance_ratio
- country

- day_trade_ratio
- list_date
- min_tick_size
- type
- tradable_chain_id
- rhs_tradability
- fractional_tradability
- default_collar_fraction

`robin_stocks.stocks.`**`get_latest_price`**(*inputSymbols*, *priceType=None*, *includeExtendedHours=True*)

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.
- **`priceType`** (*`str`*) – Can either be 'ask_price' or 'bid_price'. If this parameter is set, then includeExtendedHours is ignored.
- **`includeExtendedHours`** (*`bool`*) – Leave as True if you want to get extendedhours price if available. False if you only want regular hours price, even after hours.

Returns [list] 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 [str] 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 **`url`** (*`str`*) – The url of the stock as a string.

Returns [str] 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 [list] 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.

Dictionary Keys

- api_source
- author
- num_clicks

- preview_image_url
- published_at
- relay_url
- source
- summary
- title
- updated_at
- url
- uuid
- related_instruments
- preview_text
- currency_id

`robin_stocks.stocks.`**`get_pricebook_by_id`**(*stock_id*, *info=None*)

Represents Level II Market Data provided for Gold subscribers

Parameters

- **`stock_id`** (*`str`*) – robinhood stock id
- **`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 dictionary of asks and bids.

`robin_stocks.stocks.`**`get_pricebook_by_symbol`**(*symbol*, *info=None*)

Represents Level II Market Data provided for Gold subscribers

Parameters

- **`symbol`** (*`str`*) – symbol id
- **`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 dictionary of asks and bids.

`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 [list] 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.

Dictionary Keys

- ask_price
- ask_size
- bid_price

- bid_size
- last_trade_price
- last_extended_hours_trade_price
- previous_close
- adjusted_previous_close
- previous_close_date
- symbol
- trading_halted
- has_traded
- last_trade_price_source
- updated_at
- instrument

`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 [dict] 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.

Dictionary Keys

- summary - value is a dictionary
- ratings - value is a list of dictionaries
- instrument_id - value is a string
- ratings_published_at - value is a string

`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 [list] 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.

Dictionary Keys

- url
- instrument
- execution_date

- multiplier
- divisor

robin_stocks.stocks.**get_stock_historicals**(*inputSymbols*, *interval='hour'*, *span='week'*, *bounds='regular'*, *info=None*)

Represents the historicl data for a stock.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **interval** (*Optional[str]*) – Interval to retrieve data for. Values are '5minute', '10minute', 'hour', 'day', 'week'. Default is 'hour'.
- **span** (*Optional[str]*) – Sets the range of the data to be either 'day', 'week', 'month', '3month', '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', 'trading', or 'regular'. Default is 'regular'
- **info** (*Optional[str]*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns [list] 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.

Dictionary Keys

- begins_at
- open_price
- close_price
- high_price
- low_price
- volume
- session
- interpolated
- symbol

robin_stocks.stocks.**get_stock_quote_by_id**(*stock_id*, *info=None*)

Represents basic stock quote information

Parameters

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

Returns [dict] 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.

Dictionary Keys

- ask_price
- ask_size
- bid_price
- bid_size

- last_trade_price
- last_extended_hours_trade_price
- previous_close
- adjusted_previous_close
- previous_close_date
- symbol
- trading_halted
- has_traded
- last_trade_price_source
- updated_at
- instrument

`robin_stocks.stocks.`**`get_stock_quote_by_symbol`**`(`*symbol*, *info=None*`)`

Represents basic stock quote information

Parameters

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

Returns [dict] 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.

Dictionary Keys

- ask_price
- ask_size
- bid_price
- bid_size
- last_trade_price
- last_extended_hours_trade_price
- previous_close
- adjusted_previous_close
- previous_close_date
- symbol
- trading_halted
- has_traded
- last_trade_price_source
- updated_at
- instrument

`robin_stocks.stocks.`**`get_symbol_by_url`**`(`*url*`)`

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

Returns [str] Returns the ticker symbol of the stock.

1.5.5 Getting Option Information

Contains functions for getting information about options.

`robin_stocks.options.`**`find_options_by_expiration`**(*inputSymbols*, *expirationDate*, *optionType=None*, *info=None*)

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

Parameters

- **inputSymbols** (*str*) – The ticker of either a single stock or a list of stocks.
- **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_by_expiration_and_strike`**(*inputSymbols*, *expirationDate*, *strikePrice*, *optionType=None*, *info=None*)

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

Parameters

- **inputSymbols** (*str*) – The ticker of either a single stock or a list of stocks.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strikePrice** (*str*) – Represents the strike price to filter for.
- **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_by_specific_profitability`**(*inputSymbols*, *expirationDate=None*, *strikePrice=None*, *optionType=None*, *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. Leave as None to get all available dates.
- **strikePrice** (*str*) – Represents the price of the option. Leave as None to get all available strike prices.
- **optionType** (*Optional[str]*) – Can be either 'call' or 'put' or leave blank to get both.
- **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.**find_options_by_strike**(*inputSymbols*, *strikePrice*, *optionType=None*, *info=None*)

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

Parameters

- **inputSymbols** (*str*) – The ticker of either a single stock or a list of stocks.
- **strikePrice** (*str*) – Represents the strike price to filter for.
- **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**(*symbol*, *expirationDate=None*, *strikePrice=None*, *optionType=None*, *info=None*)

Returns a list of all available options for a stock.

Parameters

- **symbol** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strikePrice** (*str*) – Represents the strike price of the option.
- **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_aggregate_positions`**(*info=None*)

Collapses all option orders for a stock into a single dictionary.

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 order. 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_all_option_positions`**(*info=None*)

Returns all option positions ever held for the account.

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 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_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_market_options`**(*info=None*)

Returns a list of all options.

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 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_open_option_positions`**(*info=None*)

Returns all open option positions for the account.

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 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_option_historicals`**(*symbol*, *expirationDate*, *strikePrice*, *optionType*, *interval='hour'*, *span='week'*, *bounds='regular'*, *info=None*)

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.
- **`strikePrice`** (*`str`*) – Represents the price of the option.
- **`optionType`** (*`str`*) – Can be either ‘call’ or ‘put’.
- **`interval`** (*`Optional[str]`*) – Interval to retrieve data for. Values are ‘5minute’, ‘10minute’, ‘hour’, ‘day’, ‘week’. Default is ‘hour’.
- **`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 'regular', 'trading', and 'extended'. regular hours are 6 hours long, trading hours are 9 hours long, and extended hours are 16 hours long. Default is 'regular'
- **info** (*Optional[str]*) – Will filter the results to have a list of the values that correspond to key that matches info.

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*, *strikePrice*, *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.
- **strikePrice** (*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**(*inputSymbols*, *expirationDate*, *strikePrice*, *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

- **inputSymbols** (*str*) – The ticker of the stock.
- **expirationDate** (*str*) – Represents the expiration date in the format YYYY-MM-DD.
- **strikePrice** (*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.

`robin_stocks.options.spinning_cursor()`
This is a generator function to yield a character.

`robin_stocks.options.write_spinner()`
Function to create a spinning cursor to tell user that the code is working on getting market data.

1.5.6 Getting Market Information

Contains functions for getting market level data.

`robin_stocks.markets.get_all_stocks_from_market_tag(tag, info=None)`
Returns all the stock quote information that matches a tag category.

Parameters

- **tag** (*str*) – The category to filter for. Examples include 'biopharmaceutical', 'upcoming-earnings', 'most-popular-under-25', and 'technology'.
- **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.

Dictionary Keys

- ask_price
- ask_size
- bid_price
- bid_size
- last_trade_price
- last_extended_hours_trade_price
- previous_close
- adjusted_previous_close
- previous_close_date
- symbol
- trading_halted
- has_traded
- last_trade_price_source
- updated_at
- instrument

`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.

Dictionary Keys

- asset_currency
- display_only
- id
- max_order_size
- min_order_size
- min_order_price_increment
- min_order_quantity_increment
- name
- quote_currency
- symbol
- tradability

`robin_stocks.markets.`**`get_market_hours`**(*market*, *date*, *info=None*)

Returns the opening and closing hours of a specific market on a specific date. Also will tell you if market is market is open on that date. Can be used with past or future dates.

Parameters

- **market** (*str*) – The 'mic' value for the market. Can be found using get_markets().
- **date** (*str*) – The date you want to get information for. format is YYYY-MM-DD.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the specific market. If info parameter is provided, the string value for the corresponding key will be provided.

Dictionary Keys

- date
- is_open
- opens_at
- closes_at
- extended_opens_at
- extended_closes_at
- previous_open_hours
- next_open_hours

`robin_stocks.markets.`**`get_market_next_open_hours`**(*market*, *info=None*)

Returns the opening and closing hours for the next open trading day after today. Also will tell you if market is market is open on that date.

Parameters

- **market** (*str*) – The 'mic' value for the market. Can be found using get_markets().

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

Returns Returns a dictionary of key/value pairs for the specific market. If info parameter is provided, the string value for the corresponding key will be provided.

Dictionary Keys

- date
- is_open
- opens_at
- closes_at
- extended_opens_at
- extended_closes_at
- previous_open_hours
- next_open_hours

robin_stocks.markets.**get_market_next_open_hours_after_date**(*market*, *date*, *info=None*)

Returns the opening and closing hours for the next open trading day after a date that is specified. Also will tell you if market is market is open on that date.

Parameters

- **market** (*str*) – The 'mic' value for the market. Can be found using get_markets().
- **date** (*str*) – The date you want to find the next available trading day after. format is YYYY-MM-DD.
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the specific market. If info parameter is provided, the string value for the corresponding key will be provided.

Dictionary Keys

- date
- is_open
- opens_at
- closes_at
- extended_opens_at
- extended_closes_at
- previous_open_hours
- next_open_hours

robin_stocks.markets.**get_market_today_hours**(*market*, *info=None*)

Returns the opening and closing hours of a specific market for today. Also will tell you if market is market is open on that date.

Parameters

- **market** (*str*) – The 'mic' value for the market. Can be found using get_markets().
- **info** (*Optional[str]*) – Will filter the results to get a specific value.

Returns Returns a dictionary of key/value pairs for the specific market. If info parameter is provided, the string value for the corresponding key will be provided.

Dictionary Keys

- date
- is_open
- opens_at
- closes_at
- extended_opens_at
- extended_closes_at
- previous_open_hours
- next_open_hours

`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.

Dictionary Keys

- url
- todays_hours
- mic
- operating_mic
- acronym
- name
- city
- country
- timezone
- website

`robin_stocks.markets.`**`get_top_100`**`(`*info=None*`)`

Returns a list of the Top 100 stocks on Robin Hood.

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

Dictionary Keys

- ask_price
- ask_size
- bid_price
- bid_size
- last_trade_price

- last_extended_hours_trade_price
- previous_close
- adjusted_previous_close
- previous_close_date
- symbol
- trading_halted
- has_traded
- last_trade_price_source
- updated_at
- instrument

`robin_stocks.markets.`**`get_top_movers`**(*info=None*)

Returns a list of the Top 20 movers on Robin Hood.

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

Dictionary Keys

- ask_price
- ask_size
- bid_price
- bid_size
- last_trade_price
- last_extended_hours_trade_price
- previous_close
- adjusted_previous_close
- previous_close_date
- symbol
- trading_halted
- has_traded
- last_trade_price_source
- updated_at
- instrument

`robin_stocks.markets.`**`get_top_movers_sp500`**(*direction*, *info=None*)

Returns a list of the top S&P500 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.

Dictionary Keys

- instrument_url
- symbol
- updated_at
- price_movement
- description

1.5.7 Getting Positions and Account Information

Contains functions for getting information related to the user account.

`robin_stocks.account.`**`build_holdings`**(*with_dividends=False*)

Builds a dictionary of important information regarding the stocks and positions owned by the user.

Parameters **`with_dividends`** (*`bool`*) – True if you want to include divident information.

Returns Returns a dictionary where the keys are the stock tickers and the value is another dictionary that has the stock price, quantity held, equity, percent change, equity change, type, name, id, pe ratio, percentage of portfolio, and average buy price.

`robin_stocks.account.`**`build_user_profile`**()

Builds a dictionary of important information regarding the user account.

Returns Returns a dictionary that has total equity, extended hours equity, cash, and divendend total.

`robin_stocks.account.`**`delete_symbols_from_watchlist`**(*inputSymbols*, *name='My First List'*)

Deletes multiple stock tickers from a watchlist.

Parameters

- **`inputSymbols`** (*`str or list`*) – May be a single stock ticker or a list of stock tickers.
- **`name`** (*`Optional[str]`*) – The name of the watchlist to delete data from.

Returns Returns result of the delete request.

`robin_stocks.account.`**`download_all_documents`**(*doctype=None*, *dirpath=None*)

Downloads all the documents associated with an account and saves them as a PDF. If no name is given, document is saved as a combination of the data of creation, type, and id. If no directory is given, document is saved in the root directory of code.

Parameters

- **`doctype`** (*`Optional[str]`*) – The type of document to download, such as account_statement.
- **`dirpath`** (*`Optional[str]`*) – The directory of where to save the documents.

Returns Returns the list of documents from get_documents(info=None)

`robin_stocks.account.`**`download_document`**(*url*, *name=None*, *dirpath=None*)

Downloads a document and saves as it as a PDF. If no name is given, document is saved as the name that Robinhood has for the document. If no directory is given, document is saved in the root directory of code.

Parameters

- **url** (*str*) – The url of the document. Can be found by using get_documents(info='download_url').
- **name** (*Optional[str]*) – The name to save the document as.
- **dirpath** (*Optional[str]*) – The directory of where to save the document.

Returns Returns the data from the get request.

`robin_stocks.account.`**`get_all_positions`**(*info=None*)

Returns a list containing every position ever traded.

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

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

Dictionary Keys

- url
- instrument
- account
- account_number
- average_buy_price
- pending_average_buy_price
- quantity
- intraday_average_buy_price
- intraday_quantity
- shares_held_for_buys
- shares_held_for_sells
- shares_held_for_stock_grants
- shares_held_for_options_collateral
- shares_held_for_options_events
- shares_pending_from_options_events
- updated_at
- created_at

`robin_stocks.account.`**`get_all_watchlists`**(*info=None*)

Returns a list of all watchlists that have been created. Everyone has a 'My First List' watchlist.

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

Returns Returns a list of the watchlists. Keywords are 'url', 'user', and 'name'.

`robin_stocks.account.`**`get_bank_account_info`**(*id*, *info=None*)

Returns a single dictionary of bank information

Parameters

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

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

`robin_stocks.account.`**`get_bank_transfers`**`(`*info=None*`)`

Returns all bank transfers made for the account.

Parameters **`info`** (*`Optional[str]`*) – Will filter the results to get a specific value. 'direction' gives if it was deposit or withdrawl.

Returns Returns a list of dictionaries of key/value pairs for each transfer. 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.account.`**`get_card_transactions`**`(`*cardType=None*, *info=None*`)`

Returns all debit card transactions made on the account

Parameters

- **`cardType`** (*`Optional[str]`*) – Will filter the card transaction types. Can be 'pending' or 'settled'.
- **`info`** (*`Optional[str]`*) – Will filter the results to get a specific value. 'direction' gives if it was debit or credit.

Returns Returns a list of dictionaries of key/value pairs for each transfer. 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.account.`**`get_day_trades`**`(`*info=None*`)`

Returns recent day trades.

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 day trade. 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.account.`**`get_dividends`**`(`*info=None*`)`

Returns a list of dividend trasactions that include information such as the percentage rate, amount, shares of held stock, and date paid.

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

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

Dictionary Keys

- id
- url
- account
- instrument
- amount
- rate
- position
- withholding
- record_date
- payable_date
- paid_at

- state
- nra_withholding
- drip_enabled

`robin_stocks.account.`**`get_dividends_by_instrument`**(*instrument*, *dividend_data*)

Returns a dictionary with three fields when given the instrument value for a stock

Parameters

- **instrument** (*str*) – The instrument to get the dividend data.
- **dividend_data** (*list*) – The information returned by get_dividends().

Returns dividend_rate – the rate paid for a single share of a specified stock total_dividend – the total dividend paid based on total shares for a specified stock amount_paid_to_date – total amount earned by account for this particular stock

`robin_stocks.account.`**`get_documents`**(*info=None*)

Returns a list of documents that have been released by Robinhood to the account.

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 document. 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.account.`**`get_latest_notification`**()

Returns the time of the latest notification.

Returns Returns a dictionary of key/value pairs. But there is only one key, 'last_viewed_at'

`robin_stocks.account.`**`get_linked_bank_accounts`**(*info=None*)

Returns all linked bank accounts.

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 bank.

`robin_stocks.account.`**`get_margin_calls`**(*symbol=None*)

Returns either all margin calls or margin calls for a specific stock.

Parameters **symbol** (*Optional[str]*) – Will determine which stock to get margin calls for.

Returns Returns a list of dictionaries of key/value pairs for each margin call.

`robin_stocks.account.`**`get_margin_interest`**(*info=None*)

Returns a list of margin interest.

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 interest. 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.account.`**`get_notifications`**(*info=None*)

Returns a list of notifications.

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 notification. 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.account.`**`get_open_stock_positions`**(*info=None*)

Returns a list of stocks that are currently held.

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

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

Dictionary Keys

- url
- instrument
- account
- account_number
- average_buy_price
- pending_average_buy_price
- quantity
- intraday_average_buy_price
- intraday_quantity
- shares_held_for_buys
- shares_held_for_sells
- shares_held_for_stock_grants
- shares_held_for_options_collateral
- shares_held_for_options_events
- shares_pending_from_options_events
- updated_at
- created_at

`robin_stocks.account.`**`get_referrals`**`(`*info=None*`)`
Returns a list of referrals.

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 referral. 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.account.`**`get_stock_loan_payments`**`(`*info=None*`)`
Returns a list of loan payments.

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 payment. 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.account.`**`get_subscription_fees`**`(`*info=None*`)`
Returns a list of subscription fees.

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 fee. 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.account.`**`get_total_dividends`**`()`
Returns a float number representing the total amount of dividends paid to the account.

Returns Total dollar amount of dividends paid to the account as a 2 precision float.

`robin_stocks.account.`**`get_watchlist_by_name`**(*name='My First List'*, *info=None*)

Returns a list of information related to the stocks in a single watchlist.

Parameters

- **`name`** (`Optional[str]`) – The name of the watchlist to get data from.
- **`info`** (`Optional[str]`) – Will filter the results to get a specific value.

Returns Returns a list of dictionaries that contain the instrument urls and a url that references itself.

`robin_stocks.account.`**`get_wire_transfers`**(*info=None*)

Returns a list of wire transfers.

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 wire transfer. 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.account.`**`load_phoenix_account`**(*info=None*)

Returns unified information about your account.

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

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

Dictionary Keys

- account_buying_power
- cash_available_from_instant_deposits
- cash_held_for_currency_orders
- cash_held_for_dividends
- cash_held_for_equity_orders
- cash_held_for_options_collateral
- cash_held_for_orders
- crypto
- crypto_buying_power
- equities
- extended_hours_portfolio_equity
- instant_allocated
- levered_amount
- near_margin_call
- options_buying_power
- portfolio_equity
- portfolio_previous_close
- previous_close
- regular_hours_portfolio_equity
- total_equity
- total_extended_hours_equity

- total_extended_hours_market_value
- total_market_value
- total_regular_hours_equity
- total_regular_hours_market_value
- uninvested_cash
- withdrawable_cash

`robin_stocks.account.`**`post_symbols_to_watchlist`**(*inputSymbols*, *name='My First List'*)
Posts multiple stock tickers to a watchlist.

Parameters

- **inputSymbols** (*str or list*) – May be a single stock ticker or a list of stock tickers.
- **name** (*Optional[str]*) – The name of the watchlist to post data to.

Returns Returns result of the post request.

`robin_stocks.account.`**`unlink_bank_account`**(*id*)
Unlinks a bank account.

Parameters **id** (*str*) – The bank id.

Returns Information returned from post request.

1.5.8 Placing and Cancelling Orders

Contains all functions for placing orders for stocks, options, and crypto.

`robin_stocks.orders.`**`cancel_all_crypto_orders`**()
Cancels all crypto orders.

Returns Returns the order information for the orders that were cancelled.

`robin_stocks.orders.`**`cancel_all_option_orders`**()
Cancels all option orders.

Returns Returns the order information for the orders that were cancelled.

`robin_stocks.orders.`**`cancel_all_stock_orders`**()
Cancels all stock orders.

Returns The list of orders that were cancelled.

`robin_stocks.orders.`**`cancel_crypto_order`**(*orderID*)
Cancels a specific crypto order.

Parameters **orderID** (*str*) – The ID associated with the order. Can be found using get_all_crypto_orders(info=None).

Returns Returns the order information for the order that was cancelled.

`robin_stocks.orders.`**`cancel_option_order`**(*orderID*)
Cancels a specific option order.

Parameters **orderID** (*str*) – The ID associated with the order. Can be found using get_all_option_orders(info=None).

Returns Returns the order information for the order that was cancelled.

robin_stocks.orders.**cancel_stock_order**(*orderID*)
Cancels a specific order.

Parameters **orderID** (*str*) – The ID associated with the order. Can be found using get_all_stock_orders(info=None).

Returns Returns the order information for the order that was cancelled.

robin_stocks.orders.**find_stock_orders**(***arguments*)
Returns a list of orders that match the keyword parameters.

Parameters **arguments** (*str*) – Variable length of keyword arguments. EX. find_orders(symbol='FB',cancel=None,quantity=1)

Returns Returns a list of orders.

robin_stocks.orders.**get_all_crypto_orders**(*info=None*)
Returns a list of all the crypto orders that have been processed for the account.

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 option order. 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.orders.**get_all_open_crypto_orders**(*info=None*)
Returns a list of all the crypto orders that have been processed for the account.

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 option order. 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.orders.**get_all_open_option_orders**(*info=None*)
Returns a list of all the orders that are currently open.

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 order. 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.orders.**get_all_open_stock_orders**(*info=None*)
Returns a list of all the orders that are currently open.

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 order. 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.orders.**get_all_option_orders**(*info=None*)
Returns a list of all the option orders that have been processed for the account.

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 option order. 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.orders.**get_all_stock_orders**(*info=None*)
Returns a list of all the orders that have been processed for the account.

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 order. 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.orders.**get_crypto_order_info**(*order_id*)
Returns the information for a single crypto order.

Parameters **order_id** (*str*) – The ID associated with the option order.

Returns Returns a list of dictionaries of key/value pairs for the order.

robin_stocks.orders.**get_option_order_info**(*order_id*)
Returns the information for a single option order.

Parameters **order_id** (*str*) – The ID associated with the option order.

Returns Returns a list of dictionaries of key/value pairs for the order.

robin_stocks.orders.**get_stock_order_info**(*orderID*)
Returns the information for a single order.

Parameters **orderID** (*str*) – The ID associated with the order. Can be found using get_all_orders(info=None) or get_all_orders(info=None).

Returns Returns a list of dictionaries of key/value pairs for the order.

robin_stocks.orders.**order**(*symbol*, *quantity*, *orderType*, *trigger*, *side*, *priceType=None*, *limitPrice=None*, *stopPrice=None*, *timeInForce='gtc'*, *extendedHours=False*)
A generic order function. All parameters must be supplied.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to sell.
- **quantity** (*int*) – The number of stocks to sell.
- **orderType** (*str*) – Either 'market' or 'limit'
- **trigger** (*str*) – Either 'immediate' or 'stop'
- **side** (*str*) – Either 'buy' or 'sell'
- **priceType** (*str*) – Can either be 'ask_price', 'bid_price', or 'None' for last_trade_price. If this parameter is not None, then extendedHours parameter is ignored.
- **limitPrice** (*float*) – The price to trigger the market order.
- **stopPrice** (*float*) – The price to trigger the limit or market order.
- **timeInForce** (*str*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase or selling of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_crypto_by_price**(*symbol*, *amountInDollars*, *priceType='ask_price'*, *timeInForce='gtc'*)
Submits a market order for a crypto by specifying the amount in dollars that you want to trade. Good for share fractions up to 8 decimal places.

Parameters

- **symbol** (*str*) – The crypto ticker of the crypto to trade.
- **amountInDollars** (*float*) – The amount in dollars of the crypto you want to buy.
- **priceType** (*str*) – The type of price to get. Can be 'ask_price', 'bid_price', or 'mark_price'

- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the buying of crypto, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_crypto_by_quantity**(*symbol*, *quantity*, *priceType='ask_price'*, *timeInForce='gtc'*)

Submits a market order for a crypto by specifying the decimal amount of shares to buy. Good for share fractions up to 8 decimal places.

Parameters

- **symbol** (*str*) – The crypto ticker of the crypto to trade.
- **quantity** (*float*) – The decimal amount of shares to buy.
- **priceType** (*str*) – The type of price to get. Can be 'ask_price', 'bid_price', or 'mark_price'
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the buying of crypto, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_crypto_limit**(*symbol*, *quantity*, *price*, *timeInForce='gtc'*)

Submits a limit order for a crypto by specifying the decimal amount of shares to buy. Good for share fractions up to 8 decimal places.

Parameters

- **symbol** (*str*) – The crypto ticker of the crypto to trade.
- **quantity** (*float*) – The decimal amount of shares to buy.
- **price** (*float*) – The limit price to set for the crypto.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the buying of crypto, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_fractional_by_price**(*symbol*, *amountInDollars*, *timeInForce='gtc'*, *priceType='ask_price'*, *extendedHours=False*)

Submits a market order to be executed immediately for fractional shares by specifying the amount in dollars that you want to trade. Good for share fractions up to 6 decimal places.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to purchase.
- **amountInDollars** (*float*) – The amount in dollars of the fractional shares you want to buy.

- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **priceType** (*str*) – Can either be 'ask_price', 'bid_price', or 'None' for last_trade_price. If this parameter is not None, then extendedHours parameter is ignored.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_fractional_by_quantity**(*symbol*, *quantity*, *timeInForce='gtc'*, *priceType='ask_price'*, *extendedHours=False*)

Submits a market order to be executed immediately for fractional shares by specifying the amount that you want to trade. Good for share fractions up to 6 decimal places.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to purchase.
- **quantity** (*float*) – The amount of the fractional shares you want to buy.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **priceType** (*str*) – Can either be 'ask_price', 'bid_price', or 'None' for last_trade_price. If this parameter is not None, then extendedHours parameter is ignored.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_limit**(*symbol*, *quantity*, *limitPrice*, *timeInForce='gtc'*, *extendedHours=False*)

Submits a limit order to be executed once a certain price is reached.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to purchase.
- **quantity** (*int*) – The number of stocks to buy.
- **limitPrice** (*float*) – The price to trigger the buy order.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_market**(*symbol*, *quantity*, *timeInForce='gtc'*, *priceType='ask_price'*, *extendedHours=False*)

Submits a market order to be executed immediately.

Parameters

- **`symbol`** (*`str`*) – The stock ticker of the stock to purchase.
- **`quantity`** (*`int`*) – The number of stocks to buy.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **`priceType`** (*`str`*) – Can either be 'ask_price', 'bid_price', or 'None' for last_trade_price. If this parameter is not None, then extendedHours parameter is ignored.
- **`extendedHours`** (*`Optional[str]`*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_buy_option_limit`**(*positionEffect*, *creditOrDebit*, *price*, *symbol*, *quantity*, *expirationDate*, *strike*, *optionType='both'*, *timeInForce='gtc'*)

Submits a limit order for an option. i.e. place a long call or a long put.

Parameters

- **`positionEffect`** (*`str`*) – Either 'open' for a buy to open effect or 'close' for a buy to close effect.
- **`creditOrDebit`** (*`str`*) – Either 'debit' or 'credit'.
- **`price`** (*`float`*) – The limit price to trigger a buy of the option.
- **`symbol`** (*`str`*) – The stock ticker of the stock to trade.
- **`quantity`** (*`int`*) – The number of options to buy.
- **`expirationDate`** (*`str`*) – The expiration date of the option in 'YYYY-MM-DD' format.
- **`strike`** (*`float`*) – The strike price of the option.
- **`optionType`** (*`str`*) – This should be 'call' or 'put'
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the buying of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_buy_option_stop_limit`**(*positionEffect*, *creditOrDebit*, *limitPrice*, *stopPrice*, *symbol*, *quantity*, *expirationDate*, *strike*, *optionType='both'*, *timeInForce='gtc'*)

Submits a stop order to be turned into a limit order once a certain stop price is reached.

Parameters

- **`positionEffect`** (*`str`*) – Either 'open' for a buy to open effect or 'close' for a buy to close effect.
- **`creditOrDebit`** (*`str`*) – Either 'debit' or 'credit'.
- **`limitPrice`** (*`float`*) – The limit price to trigger a buy of the option.
- **`stopPrice`** (*`float`*) – The price to trigger the limit order.

- **symbol** (*str*) – The stock ticker of the stock to trade.
- **quantity** (*int*) – The number of options to buy.
- **expirationDate** (*str*) – The expiration date of the option in 'YYYY-MM-DD' format.
- **strike** (*float*) – The strike price of the option.
- **optionType** (*str*) – This should be 'call' or 'put'
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the buying of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_stop_limit**(*symbol*, *quantity*, *limitPrice*, *stopPrice*, *timeInForce='gtc'*, *extendedHours=False*)

Submits a stop order to be turned into a limit order once a certain stop price is reached.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to purchase.
- **quantity** (*int*) – The number of stocks to buy.
- **limitPrice** (*float*) – The price to trigger the market order.
- **stopPrice** (*float*) – The price to trigger the limit order.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_buy_stop_loss**(*symbol*, *quantity*, *stopPrice*, *timeInForce='gtc'*, *extendedHours=False*)

Submits a stop order to be turned into a market order once a certain stop price is reached.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to purchase.
- **quantity** (*int*) – The number of stocks to buy.
- **stopPrice** (*float*) – The price to trigger the market order.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_option_credit_spread**(*price*, *symbol*, *quantity*, *spread*, *timeInForce='gtc'*)

Submits a limit order for an option credit spread.

Parameters

- **price** (*float*) – The limit price to trigger a sell of the option.
- **symbol** (*str*) – The stock ticker of the stock to trade.
- **quantity** (*int*) – The number of options to sell.
- **spread** (*dict*) – A dictionary of spread options with the following keys:
  - expirationDate: The expiration date of the option in 'YYYY-MM-DD' format.
  - strike: The strike price of the option.
  - optionType: This should be 'call' or 'put'.
  - effect: This should be 'open' or 'close'.
  - action: This should be 'buy' or 'sell'.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' = execute at opening.

Returns Dictionary that contains information regarding the trading of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_option_debit_spread**(*price*, *symbol*, *quantity*, *spread*, *timeInForce='gtc'*)

Submits a limit order for an option debit spread.

Parameters

- **price** (*float*) – The limit price to trigger a sell of the option.
- **symbol** (*str*) – The stock ticker of the stock to trade.
- **quantity** (*int*) – The number of options to sell.
- **spread** (*dict*) – A dictionary of spread options with the following keys:
  - expirationDate: The expiration date of the option in 'YYYY-MM-DD' format.
  - strike: The strike price of the option.
  - optionType: This should be 'call' or 'put'.
  - effect: This should be 'open' or 'close'.
  - action: This should be 'buy' or 'sell'.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the trading of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_option_spread**(*direction*, *price*, *symbol*, *quantity*, *spread*, *timeInForce='gtc'*)

Submits a limit order for an option spread. i.e. place a debit / credit spread

Parameters

- **direction** (*str*) – Can be "credit" or "debit".
- **price** (*float*) – The limit price to trigger a trade of the option.
- **symbol** (*str*) – The stock ticker of the stock to trade.

- **quantity** (*int*) – The number of options to trade.
- **spread** (*dict*) – A dictionary of spread options with the following keys:
  - expirationDate: The expiration date of the option in 'YYYY-MM-DD' format.
  - strike: The strike price of the option.
  - optionType: This should be 'call' or 'put'.
  - effect: This should be 'open' or 'close'.
  - action: This should be 'buy' or 'sell'.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the trading of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_crypto_by_price`**(*symbol*, *amountInDollars*, *priceType='bid_price'*, *timeInForce='gtc'*)

Submits a market order for a crypto by specifying the amount in dollars that you want to trade. Good for share fractions up to 8 decimal places.

Parameters

- **symbol** (*str*) – The crypto ticker of the crypto to trade.
- **amountInDollars** (*float*) – The amount in dollars of the crypto you want to sell.
- **priceType** (*str*) – The type of price to get. Can be 'ask_price', 'bid_price', or 'mark_price'
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the selling of crypto, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_crypto_by_quantity`**(*symbol*, *quantity*, *priceType='bid_price'*, *timeInForce='gtc'*)

Submits a market order for a crypto by specifying the decimal amount of shares to buy. Good for share fractions up to 8 decimal places.

Parameters

- **symbol** (*str*) – The crypto ticker of the crypto to trade.
- **quantity** (*float*) – The decimal amount of shares to sell.
- **priceType** (*str*) – The type of price to get. Can be 'ask_price', 'bid_price', or 'mark_price'
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the selling of crypto, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_crypto_limit`**(*symbol*, *quantity*, *price*, *timeInForce='gtc'*)

Submits a limit order for a crypto by specifying the decimal amount of shares to sell. Good for share fractions up to 8 decimal places.

Parameters

- **`symbol`** (*`str`*) – The crypto ticker of the crypto to trade.
- **`quantity`** (*`float`*) – The decimal amount of shares to sell.
- **`price`** (*`float`*) – The limit price to set for the crypto.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the selling of crypto, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_fractional_by_price`**(*symbol*, *amountInDollars*, *timeInForce='gtc'*, *priceType='bid_price'*, *extendedHours=False*)

Submits a market order to be executed immediately for fractional shares by specifying the amount in dollars that you want to trade. Good for share fractions up to 6 decimal places.

Parameters

- **`symbol`** (*`str`*) – The stock ticker of the stock to purchase.
- **`amountInDollars`** (*`float`*) – The amount in dollars of the fractional shares you want to buy.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **`extendedHours`** (*`Optional[str]`*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_fractional_by_quantity`**(*symbol*, *quantity*, *timeInForce='gtc'*, *priceType='bid_price'*, *extendedHours=False*)

Submits a market order to be executed immediately for fractional shares by specifying the amount that you want to trade. Good for share fractions up to 6 decimal places.

Parameters

- **`symbol`** (*`str`*) – The stock ticker of the stock to purchase.
- **`quantity`** (*`float`*) – The amount of the fractional shares you want to buy.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **`extendedHours`** (*`Optional[str]`*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the purchase of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_limit`**(*symbol*, *quantity*, *limitPrice*, *timeInForce='gtc'*, *extendedHours=False*)

Submits a limit order to be executed once a certain price is reached.

Parameters

- **`symbol`** (*`str`*) – The stock ticker of the stock to sell.
- **`quantity`** (*`int`*) – The number of stocks to sell.
- **`limitPrice`** (*`float`*) – The price to trigger the sell order.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **`extendedHours`** (*`Optional[str]`*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the selling of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_market`**(*symbol*, *quantity*, *timeInForce='gtc'*, *priceType='bid_price'*, *extendedHours=False*)

Submits a market order to be executed immediately.

Parameters

- **`symbol`** (*`str`*) – The stock ticker of the stock to sell.
- **`quantity`** (*`int`*) – The number of stocks to sell.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **`priceType`** (*`str`*) – Can either be 'ask_price', 'bid_price', or 'None' for last_trade_price. If this parameter is not None, then extendedHours parameter is ignored.
- **`extendedHours`** (*`Optional[str]`*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the selling of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_option_limit`**(*positionEffect*, *creditOrDebit*, *price*, *symbol*, *quantity*, *expirationDate*, *strike*, *optionType='both'*, *timeInForce='gtc'*)

Submits a limit order for an option. i.e. place a short call or a short put.

Parameters

- **`positionEffect`** (*`str`*) – Either 'open' for a sell to open effect or 'close' for a sell to close effect.
- **`creditOrDebit`** (*`str`*) – Either 'debit' or 'credit'.
- **`price`** (*`float`*) – The limit price to trigger a sell of the option.
- **`symbol`** (*`str`*) – The stock ticker of the stock to trade.
- **`quantity`** (*`int`*) – The number of options to sell.

- **expirationDate** (*str*) – The expiration date of the option in 'YYYY-MM-DD' format.
- **strike** (*float*) – The strike price of the option.
- **optionType** (*str*) – This should be 'call' or 'put'
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the selling of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_sell_option_stop_limit**(*positionEffect*, *creditOrDebit*, *limitPrice*, *stopPrice*, *symbol*, *quantity*, *expirationDate*, *strike*, *optionType='both'*, *timeInForce='gtc'*)

Submits a stop order to be turned into a limit order once a certain stop price is reached.

Parameters

- **positionEffect** (*str*) – Either 'open' for a buy to open effect or 'close' for a buy to close effect.
- **creditOrDebit** (*str*) – Either 'debit' or 'credit'.
- **limitPrice** (*float*) – The limit price to trigger a buy of the option.
- **stopPrice** (*float*) – The price to trigger the limit order.
- **symbol** (*str*) – The stock ticker of the stock to trade.
- **quantity** (*int*) – The number of options to buy.
- **expirationDate** (*str*) – The expiration date of the option in 'YYYY-MM-DD' format.
- **strike** (*float*) – The strike price of the option.
- **optionType** (*str*) – This should be 'call' or 'put'
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.

Returns Dictionary that contains information regarding the buying of options, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

robin_stocks.orders.**order_sell_stop_limit**(*symbol*, *quantity*, *limitPrice*, *stopPrice*, *timeInForce='gtc'*, *extendedHours=False*)

Submits a stop order to be turned into a limit order once a certain stop price is reached.

Parameters

- **symbol** (*str*) – The stock ticker of the stock to sell.
- **quantity** (*int*) – The number of stocks to sell.
- **limitPrice** (*float*) – The price to trigger the market order.
- **stopPrice** (*float*) – The price to trigger the limit order.
- **timeInForce** (*Optional[str]*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **extendedHours** (*Optional[str]*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the selling of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

`robin_stocks.orders.`**`order_sell_stop_loss`**(*symbol*, *quantity*, *stopPrice*, *timeInForce='gtc'*, *extendedHours=False*)

Submits a stop order to be turned into a market order once a certain stop price is reached.

Parameters

- **`symbol`** (*`str`*) – The stock ticker of the stock to sell.
- **`quantity`** (*`int`*) – The number of stocks to sell.
- **`stopPrice`** (*`float`*) – The price to trigger the market order.
- **`timeInForce`** (*`Optional[str]`*) – Changes how long the order will be in effect for. 'gtc' = good until cancelled. 'gfd' = good for the day. 'ioc' = immediate or cancel. 'opg' execute at opening.
- **`extendedHours`** (*`Optional[str]`*) – Premium users only. Allows trading during extended hours. Should be true or false.

Returns Dictionary that contains information regarding the selling of stocks, such as the order id, the state of order (queued, confired, filled, failed, canceled, etc.), the price, and the quantity.

1.5.9 Getting Crypto Information

Contains functions to get information about crypto-currencies.

`robin_stocks.crypto.`**`get_crypto_currency_pairs`**(*info=None*)

Gets a list of all the cypto currencies that you can trade.

Parameters **`info`** (*`Optional[str]`*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns [list] 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.

Dictionary Keys

- asset_currency
- display_only
- id
- max_order_size
- min_order_size
- min_order_price_increment
- min_order_quantity_increment
- name
- quote_currency
- symbol
- tradability

`robin_stocks.crypto.`**`get_crypto_historicals`**(*symbol*, *interval='hour'*, *span='week'*, *bounds='24_7'*, *info=None*)

Gets historical information about a crypto including open price, close price, high price, and low price.

Parameters

- **`symbol`** (*`str`*) – The crypto ticker.
- **`interval`** (*`str`*) – The time between data points. Can be '15second', '5minute', '10minute', 'hour', 'day', or 'week'. Default is 'hour'.
- **`span`** (*`str`*) – The entire time frame to collect data points. Can be 'hour', 'day', 'week', 'month', '3month', 'year', or '5year'. Default is 'week'
- **`bound`** (*`str`*) – The times of day to collect data points. 'Regular' is 6 hours a day, 'trading' is 9 hours a day, 'extended' is 16 hours a day, '24_7' is 24 hours a day. Default is '24_7'
- **`info`** (*`Optional[str]`*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns [list] 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.

Dictionary Keys

- begins_at
- open_price
- close_price
- high_price
- low_price
- volume
- session
- interpolated
- symbol

`robin_stocks.crypto.`**`get_crypto_info`**(*symbol*, *info=None*)

Gets information about a crpyto currency.

Parameters

- **`symbol`** (*`str`*) – The crypto ticker.
- **`info`** (*`Optional[str]`*) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns [dict] If info parameter is left as None then will return a dictionary of key/value pairs for each ticker. Otherwise, it will be a strings representing the value of the key.

Dictionary Keys

- asset_currency
- display_only
- id
- max_order_size
- min_order_size

- min_order_price_increment
- min_order_quantity_increment
- name
- quote_currency
- symbol
- tradability

`robin_stocks.crypto.`**`get_crypto_positions`**(*info=None*)

Returns crypto positions for the account.

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

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

Dictionary Keys

- account_id
- cost_basis
- created_at
- currency
- id
- quantity
- quantity_available
- quantity_held_for_buy
- quantity_held_for_sell
- updated_at

`robin_stocks.crypto.`**`get_crypto_quote`**(*symbol*, *info=None*)

Gets information about a crypto including low price, high price, and open price

Parameters

- **`symbol`** (`str`) – The crypto ticker.
- **`info`** (`Optional[str]`) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns [dict] 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.

Dictionary Keys

- asset_currency
- display_only
- id
- max_order_size
- min_order_size
- min_order_price_increment

- min_order_quantity_increment
- name
- quote_currency
- symbol
- tradability

`robin_stocks.crypto.`**`get_crypto_quote_from_id`**`(`*id*, *info=None*`)`

Gets information about a crypto including low price, high price, and open price. Uses the id instead of crypto ticker.

Parameters

- **`id`** (`str`) – The id of a crypto.
- **`info`** (`Optional[str]`) – Will filter the results to have a list of the values that correspond to key that matches info.

Returns [dict] 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.

Dictionary Keys

- asset_currency
- display_only
- id
- max_order_size
- min_order_size
- min_order_price_increment
- min_order_quantity_increment
- name
- quote_currency
- symbol
- tradability

`robin_stocks.crypto.`**`load_crypto_profile`**`(`*info=None*`)`

Gets the information associated with the crypto account.

Parameters **`info`** (`Optional[str]`) – The name of the key whose value is to be returned from the function.

Returns [dict] The function returns a dictionary of key/value pairs. If a string is passed in to the info parameter, then the function will return a string corresponding to the value of the key whose name matches the info parameter.

Dictionary Keys

- apex_account_number
- created_at
- id
- rhs_account_number

- status
- status_reason_code
- updated_at
- user_id

1.5.10 Export Information

`robin_stocks.export.`**`create_absolute_csv`**(*dir_path*, *file_name*, *order_type*)

Creates a filepath given a directory and file name.

Parameters

- **`dir_path`** (*`str`*) – Absolute or relative path to the directory the file will be written.
- **`file_name`** (*`str`*) – An optional argument for the name of the file. If not defined, filename will be stock_orders_{current date}
- **`file_name`** – Will be 'stock', 'option', or 'crypto'

Returns An absolute file path as a string.

`robin_stocks.export.`**`export_completed_option_orders`**(*dir_path*, *file_name=None*)

Write all completed option orders to a csv

Parameters

- **`dir_path`** (*`str`*) – Absolute or relative path to the directory the file will be written.
- **`file_name`** (*`Optional[str]`*) – An optional argument for the name of the file. If not defined, filename will be option_orders_{current date}

`robin_stocks.export.`**`export_completed_stock_orders`**(*dir_path*, *file_name=None*)

Write all completed orders to a csv file

Parameters

- **`dir_path`** (*`str`*) – Absolute or relative path to the directory the file will be written.
- **`file_name`** (*`Optional[str]`*) – An optional argument for the name of the file. If not defined, filename will be stock_orders_{current date}

`robin_stocks.export.`**`fix_file_extension`**(*file_name*)

Takes a file extension and makes it end with .csv

Parameters **`file_name`** (*`str`*) – Name of the file.

Returns Adds or replaces the file suffix with .csv and returns it as a string.

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

Index