LINC-STEM-Hackathon API Documentation

LINC

February 2025

Contents

1 Setup Instructions **2**
- 1.1 Bash Installation . . . 2
- 1.2 Jupyter Notebook Setup . . . 2
- 1.3 Importing packages . . . 2

2 Help and Recommendations **2**

3 Overview **3**

4 Account Functions **3**
- 4.1 `get_all_orders()` . . . 3
- 4.2 `get_completed_orders()` . . . 3
- 4.3 `get_pending_orders()` . . . 3
- 4.4 `get_stoploss_orders()` . . . 4
- 4.5 `get_balance()` . . . 4
- 4.6 `get_portfolio()` . . . 4

5 Order Functions **4**
- 5.1 `buy(ticker, amount, price, days_to_cancel)` . . . 4
- 5.2 `sell(ticker, amount, price, days_to_cancel)` . . . 5
- 5.3 `stoploss(ticker, amount, price, days_to_cancel)` . . . 5
- 5.4 `cancel(order_id, ticker)` . . . 5

6 Market Data Functions **6**
- 6.1 `get_all_tickers()` . . . 6
- 6.2 `get_current_price(ticker)` . . . 6
- 6.3 `get_historical_data(days_back, ticker)` . . . 6

7 Example Usage **6**

1 Setup Instructions

1.1 Bash Installation

To get started, open your terminal and run the following commands to install the required packages:

```
pip install --upgrade hackathon-linc
pip install --upgrade pandas
```

1.2 Jupyter Notebook Setup

If you rather want to run this in a notebook, install using:

```
%pip install --upgrade hackathon-linc
%pip install --upgrade pandas
```

1.3 Importing packages

After installing the packages, launch your Jupyter Notebook, or a new python script (example.py) and import the modules as follows:

```
import hackathon_linc as lh
import pandas as pd
lh.init('API-KEY-HERE')
```

Replace `'API-KEY-HERE'` with your actual API key.

2 Help and Recommendations

We recommend testing all the relevant functions provided by the API to understand the structure and contents of the returned dictionaries. Consider the following tips:

- **Explore the Functions:** Run individual functions in your Jupyter Notebook and inspect the output. This will help you understand what each function returns and how to work with the data.
- **Use `pandas`:** Convert dictionaries or lists into `pandas.DataFrame` objects for easier data manipulation and visualization.
- **Leverage `numpy`:** Utilize `numpy` for numerical operations, especially when working with large datasets or performing mathematical computations.
- **Experiment:** Try different combinations of functions and observe how they interact. This exploration is key to mastering the API and building robust strategies.

3 Overview

This document provides an overview of the available API functions for interacting with the trading system. The API is organized into three main categories:

- **Account Functions** – Functions to retrieve orders, portfolio, and account balance.
- **Order Functions** – Functions to place various types of orders (buy, sell, stoploss) and to cancel orders.
- **Market Data Functions** – Functions to retrieve current market data such as ticker lists, current prices, and historical data.

4 Account Functions

These functions help you query various aspects of your account such as orders, balance, and portfolio details.

4.1 `get_all_orders()`

Description: Returns a list of all orders, including both completed and pending orders.
Returns: `List[Dict[str, Union[str, int, float]]]`
Example:

```
orders = lh.get_all_orders()
```

4.2 `get_completed_orders()`

Description: Retrieves a list of all completed orders from the account.
Returns: `List[Dict[str, Union[str, int, float]]]`
Example:

```
completed = lh.get_completed_orders()
```

4.3 `get_pending_orders()`

Description: Retrieves a list of all pending orders from the account.
Returns: `List[Dict[str, Union[str, int, float]]]`
Example:

```
pending = lh.get_pending_orders()
```

4.4 get_stoploss_orders()

Description: Retrieves a list of all stoploss orders.
Returns: `List[Dict[str, Union[str, int, float]]]`
Example:

```
stoploss_orders = lh.get_stoploss_orders()
```

4.5 get_balance()

Description: Retrieves the current account balance.
Returns: `float`
Example:

```
balance = lh.get_balance()
```

4.6 get_portfolio()

Description: Retrieves a dictionary that details the quantity of securities held for each ticker.
Returns: `Dict[str, int]`
Example:

```
portfolio = lh.get_portfolio()
```

5 Order Functions

These functions are used to place orders on the market. They support buying, selling, setting stoploss orders, and cancelling orders.

5.1 buy(ticker, amount, price, days_to_cancel)

Description: Places a buy order for a given security.
Parameters:

- `ticker (str)`: The ticker symbol.
- `amount (int)`: Number of shares to buy.
- `price (int, optional)`: The price at which to execute the order. If not specified, uses the current market price.
- `days_to_cancel (int, optional)`: The number of days the order will remain active if not executed.

Returns: `Dict` – The server response.
Example:

```
buy_response = lh.buy('STOCK1', 10, 100)
```

5.2 sell(ticker, amount, price, days_to_cancel)

Description: Places a sell order for a given security.
Parameters: Same as the `buy` function.
Returns: `Dict` – The server response.
Example:

```
sell_response = lh.sell('STOCK1', 5, 110)
```

5.3 stoploss(ticker, amount, price, days_to_cancel)

Description: Places a stoploss order for a given security to limit potential losses.
Parameters:

- `ticker (str)`: The ticker symbol.
- `amount (int)`: Number of shares.
- `price (float)`: The stoploss price.
- `days_to_cancel (int, optional)`: Duration in days for which the order is valid.

Returns: `Dict` – The server response.
Example:

```
stoploss_response = lh.stoploss('STOCK1', 10, 95.5)
```

5.4 cancel(order_id, ticker)

Description: Cancels an order by specifying either an order ID or a ticker. One of the parameters must be provided. If only ticker is provided, it will cancel all orders with that ticker. The parameter order_id is provided in the dictionary returned after placing an order. NOTE: stoploss orders CANNOT be canceled.
Parameters:

- `order_id (str, optional)`: The ID of the order to cancel.
- `ticker (str, optional)`: The ticker symbol for which all orders should be cancelled.

Returns: `Dict` – The server response.
Example:

```
cancel_response = lh.cancel(order_id='123abc', ticker
   = 'STOCK1')
```

6 Market Data Functions

These functions provide market-related data such as available tickers, current stock prices, and historical price data.

6.1 `get_all_tickers()`

Description: Retrieves a list of all available ticker symbols from the market.
Returns: `List[str]`
Example:

```
tickers = lh.get_all_tickers()
```

6.2 `get_current_price(ticker)`

Description: Retrieves the current price of a specific ticker. If no ticker is provided, returns prices for all securities.
Parameters:

- `ticker (str, optional)`: The ticker symbol. If omitted, data for all tickers is returned.

Returns: `dict` – A dictionary containing the current price(s).
Example:

```
price_data = lh.get_current_price('STOCK1')
```

6.3 `get_historical_data(days_back, ticker)`

Description: Retrieves historical data for the specified ticker(s) for a given number of days in the past (up to 365 days).
Parameters:

- `days_back (int)`: The number of days to look back (must be between 0 and 365).
- `ticker (str, optional)`: The ticker symbol. If omitted, historical data for all tickers is returned.

Returns: `dict` – Historical market data.
Example:

```
historical_data = lh.get_historical_data(30, 'STOCK1')
```

7 Example Usage

Have a look at the github here. There is an example strategy and a corresponding flask app connected to it. GitHub Repository