# MOMENTUM TRADING ASSISTANT (MTA)

# USER MANUAL

**Document revision:** 0.1

**Date:** 11.04.2025

This code is making use of the API as published by Interactive Brokers.

## CONTENTS

# 1 Introduction

Momentum-Trading-Assistant (MTA) is a python program designed to replace a Momentum Trader using the Interactive Brokers Trader Workstation (TWS) in front of the screen during market opening hours, respectively to assist him during trading hours through order executions and the application of further individualized trading rules. Herein, MTA makes use of the API as provided by Interactive Brokers. This program is especially designed for Momentum Traders following the trading style of Mark Minervini, William O'Neal, Mark Richie II, Oliver Kell, Quallamaggie and many more.

This code offers a way to hard-wire your trading rules, execute them consequently and adjust these rules periodically after your personal post analysis. Momentum trading is directional trading and means that the entry prices defined in your daily trading plan must be above the current price, see more on order execution in chapter 6.2. Your trading strategy needs to be defined daily, see more on the required input file in chapter 4.1.6. The code is currently written for trading stocks long only and has not been tested on other assets.

MTA does not make any trading decisions which are not predefined by the user in advance. Therefore, the stock selection, definition of risk and profit potential, entry and exit scenarios fully rely on the Momentum Trader's judgement and decisions. This trading plan can also be updated when MTA is running so that it can be used to assist you while live trading.

MTA is currently set up to trade the markets of the US, Germany and Japan. With minor tweaks, also other markets can easily be added. MTA can only cover one market with assets in one currency at a time. If e.g. German and US markets need to be traded with overlapping market opening times, MTA needs to be run as two separate processes in parallel, one for Germany and one for the US.

The program can be started e.g. through the cmd prompt after you have opened and logged into your TWS. Make sure TWS is set up properly so that the API connection is enabled, you have subscribed to the relevant market data feeds and MTA has sufficient writing rights.

MTA is meant not only for trading-enthusiasts among the IT community, but also momentum traders with only limited IT and esp. coding know-how. Please therefore excuse that some explanations in the README as well as the User Manual are somewhat nitty gritty.

The `README.md` provides only a very first view on how to use MTA, but does not cover any further details about its many trading rules and functionalities. Therefore, please refer also to this more detailed User Manual as provided within this project.

# 2 Liabilities

This code is tested through my personal use for the US markets during the last several years, but still, it remains your responsibility to supervise MTA sufficiently during its application to avoid any unintended activities in your portfolio. The creator cannot be held liable for any financial damage occurring while using MTA.

# 3 Quick start guide

This section contains all necessary steps you need to get MTA up and running:

1. Ensure your IB TWS is set up properly to use the API with port 7496 see IB documentation at https://www.interactivebrokers.com/campus/ibkr-api-page/twsapi-doc/#tws-config-api

2. Make sure not to use IB API as it is available via pip as this package is outdated and no longer supported. It needs to be downloaded and installed manually, see https://interactivebrokers.github.io/#. Follow the instruction on the website for installation. Make sure to add ibapi to your PYTHONPATH in your environment variables (e.g. `C:\TWS API\source\pythonclient`).
3. Subscribe to the relevant market data on you IB-account
4. Save the project to a folder of your choice.
5. Go to the `Inputs` folder and fill columns A, B and D to M of `DailyTradingPlan.xlsx`
6. Log into TWS
7. Open your CMD prompt for windows and, navigate to the folder where you saved the files (e.g. through `cd documents\foldername`)
8. Start MTA through `python main.py`

It is recommended but not necessary to start MTA before the market opens.

# 4 MTA input and output files

As detailed in this chapter, MTA requires six files as inputs for startup, which are located in the same folder and will return three files as outputs to this same folder. Only `DailyTradingPlan.xlsx` contains your daily trading plan and needs to be updated daily (see chapter 4.1.6). The input files are:

- `main.py`
- `Functionalities/`
  - `MyFunctionalities.py`
- `Utilities/`
  - `MyOrders.py`
  - `MyUtilities.py`
- `Rules/`
  - `ConstantsAndRules.py`
- `Inputs/`
  - `tickDataTemplate.xlsx`
  - `DailyTradingPlan.xlsx` (resp. `DailyTradingPlan_DE.xlsx` or `DailyTradingPlan_JP.xlsx`)

The program will return the following three files to the `Outputs` folder:

- `yymmdd_fetch_open_positions.xlsx`
- `yymmdd_fetch_new_positions.xlsx`
- `yymmdd_trading_plan.xlsx`

## 4.1 Input files

### 4.1.1 main.py

`main.py` is a python file containing the main code of MTA which will access `MyOrders.py`, `MyUtilities.py` and `ConstantsAndRules.py`. `main.py` can be started in a cmd prompt or your favorite IDE. It is not required to do any changes to `main.py`. To read more about how to start MTA properly, please refer to chapter 5.

### 4.1.2 MyOrders.py

Contains all order types that are relevant for `main.py`. It is not required to change anything in this file.

### 4.1.3 MyUtilities.py

Contains all utility functions used in `main.py`. It is not required to change anything in this file.

### 4.1.4 ConstantsAndRules.py

This file contains constants and variables used for MTA. It is the only `.py` file which should be changed to suite your personal trading requirements. On how to do this, please refer to chapter 6.1.

### 4.1.5 tickDataTemplate.xlsx

MTA fetches the bid, ask and last price as well as the related volumes every second during market opening hours (see also chapters 4.2.1 and 4.2.2). `tickDataTemplate.xlsx` is used to generate an empty dataframe to be filled by MTA. It is required for every startup of MTA. You can find the structure of the file also in Table 1.

*Table 1 - tickDataTemplate.xlsx columns*

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| | | timeStamp | Symbol | LAST price [$] | BID price [$] | ASK price [$] | BID size | ASK size | CLOSE price [$] | Volume |
| 0 | | | | | | | | | | |

Do not change this file.

### 4.1.6 DailyTradingPlan.xlsx

This file contains your trading plan for the day and needs to be updated before market opening hours but can also be updated during market opening hours (see chapter 6.11). The structure of `DailyTradingPlan.xlsx` is shown in Table 2 and an example file is given as part of this project. Please be aware again that MTA can only be used for one market with one currency at a time. If you seek to trade e.g. the German and US market with overlapping opening hours, MTA needs to be started two times as separate processes.
Each market needs therefore one dedicated trading plan as follows:

- US market → `DailyTradingPlan.xlsx`
- German market → `DailyTradingPlan_DE.xlsx`
- Japanese market → `DailyTradingPlan_JP.xlsx`

*Table 2 - DailyTradingPlan columns*

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Symbol | Company name | Security Type | Currency | Exchange | Primary Exchange | Entry price [$] | Stop price [$] | Quantity [#] | Buy limit price [$] | Profit taker price [$] | Open position | Add and reduce | Sell on close | Stop low of day | Sell bellow SMA [$] |
| 0 | | BTC | | CRYPTO | USD | SMART | PAXOS | 9 | 11 | 1 | 9.01 | 11.25 | | | | | |
| 1 | | OLLI | | STK | USD | SMART | NASDAQ | 99.25 | 96.60 | 20 | 99.75 | 109.18 | TRUE | | | | |
| 2 | | NFLX | | STK | USD | SMART | NASDAQ | 708.91 | 695.00 | 3 | 716 | 779.8 | TRUE | | | | |
| 3 | | COST | | STK | USD | SMART | NASDAQ | 909.00 | 890.00 | 2 | 918.09 | 999.9 | TRUE | | | | |
| 4 | | NVDA | | STK | USD | SMART | NASDAQ | 117.20 | 117.20 | 8 | 118.37 | 128.92 | TRUE | | | | |
| 5 | | NVDA | | STK | USD | SMART | NASDAQ | 117.20 | 117.20 | 7 | 118.37 | 134.78 | TRUE | | | | |
| 6 | | AAPL | | STK | USD | SMART | NASDAQ | 227.71 | 225.40 | 15 | 229.41 | 245.92 | FALSE | | | | |
| 7 | | TRAK | | STK | USD | SMART | NYSE | 20.34 | 19.80 | 50 | 20.54 | 22.37 | FALSE | | | | |

Columns A to M despite C, must be filled. Columns N to Q can be filled and all subsequent columns shall not be changed since this will cause MTA to error out. Further, it is recommended to leave BTC in the first row since this is used to keep the algo alive also when starting MTA hours before the market opens (BTC is trading 24/7). You can also replace this first row entry, but never delete it, since MTA will not consider it for some functions (e.g. comparison of open portfolio positions with open orders defined in `DailyTradingPlan.xlsx`, see chapter 5). As long as the entry price is 9 and the stop price 11, this row will not be considered for trading.

A short description of each of the required inputs is given below:

- (Column A) is a running index: Integers from 0 to infinity.
- Symbol (Column B) - is the stock symbol as string.
- Company name (Column C) - will be filled with the company name automatically and need not be changed (see output file in chapter 4.2.3).
- Security Type (Column D) - is currently always STK – no other assets have been tested so far.
- Currency (Column E) - is the currency of the traded asset (currently only USD, EUR and JPN tested).
- Exchange (Column F) - is the Exchange and can remain on SMART for all tested markets.
- Primary Exchange (Column G) - is the Primary Exchange related to your stock symbol (tested is NASDAQ, NYSE, IBIS and TSEJ).
- Entry price [$] (Column H) - is your entry price in the local currency (note again that the entry price must be above the current price, otherwise the order will be executed on the open, see chapter 6.2).
- Stop price [$] (Column I) - is your stop price.
- Quantity [#] (Column J) - is the quantity you seek to buy.
- Buy limit price [$] (Column K) - is your buy limit price.
- Profit taker price [$] (Column L) - is your profit taker price.
- Open position (Column M) – is a boolean to define if it is an open position (TRUE) or if you intend to buy this as a new position (empty or FALSE).
- Add and reduce (Column N) (optional) - is a boolean TRUE or left empty as this is interpreted as FALSE. For a description of the functionality, please see chapter 6.7.
- Sell on close (Column O) (optional) - is a boolean TRUE or left empty as this is interpreted as FALSE. If TRUE, the stock is sold within the last minutes of the trading day. For a description of the functionality, please see chapter 6.8.
- Stop low of day (Column P) (optional) - is a boolean TRUE or left empty as this is interpreted as FALSE. If TRUE, when the buy price is reached, a bracket order is defined with the low of the day used as stop price. Please note that the low of day price is determined only since the position was added to `DailyTradingPlan.xlsx`. If you update `DailyTradingPlan.xlsx` to add a position during the

trading session, MTA's low of day price might not reflect the correct value.

- Sell below SMA [$] (Column Q) (optional) - is a numerical entry as decimal. It is only active when also TRUE is entered for Sell on close (Column O). See chapter 6.9 for the detailed functionality.

The currency values entered (buy price etc.) must always be in the currency defined in Currency (Column E). When markets other than the US are traded, the column designations e.g. Entry price [$] is therefore technically wrong, apologies then.
Similar positions should be listed directly below each other, since the data fetch function will then only consider each stock symbol once. If the stocks are not listed directly below each other, they will be doubled in the data fetch files requiring unnecessary memory.

## 4.2 Output files

This chapter contains a short summary of the output files produced for each trading sessions. They will be saved automatically to the file path of `main.py` shortly after the market closed.
For all output files, the same naming convention as used in chapter 4.1.6 is applied incl. "_JP" for Japanese markets and "_DE" for German markets.

### 4.2.1 yymmdd_fetch_open_positions

Based on the template of `tickDataTemplate.xlsx` (see chapter 4.1.5), all relevant datapoints are fetched once per second from market opening to market close for all positions marked as open positions in `DailyTradingPlan.xlsx` (Open position in Column M = TRUE).

### 4.2.2 yymmdd_fetch_new_positions

Based on the template of `tickDataTemplate.xlsx` (see chapter 4.1.5), all relevant datapoints are fetched once per second from market opening to market close for all positions marked as new positions in `DailyTradingPlan.xlsx` (Open position in Column M = FALSE or empty).

### 4.2.3 yymmdd_trading_plan

This file is the output version of your `DailyTradingPlan.xlsx` and provides an overview of all activities which happened during the trading day incl. buys and sells as well as the application of further trading rules. Below, you can find an overview of the columns subsequent to Column Q "Sell below SMA [$]" as previously described in chapter 4.1.6:

- (Column A) – As outlined in chapter 4.1.6
- Symbol - As outlined in chapter 4.1.6
- Company name – Company name
- Security Type - As outlined in chapter 4.1.6
- Currency - As outlined in chapter 4.1.6
- Exchange - As outlined in chapter 4.1.6
- Primary Exchange - As outlined in chapter 4.1.6
- Entry price [$] (Column H) – In case of an executed new order, this shows the actual fill price.
- Stop price [$] (Column I) – In case the stop is triggered for a new, filled position or an open position,

this shows the actual fill price.

- Quantity [#] (Column J) – The quantity as per the market close considering any applied trading rules or filled stop, respectively filled profit orders.
- Buy limit price [$] (Column K) - As outlined in chapter 4.1.6
- Profit taker price [$] (Column L) – In case the profit target was reached, this shows the actual fill price.
- Open position (Column M) – As outlined in chapter 4.1.6
- Add and reduce (Column N) (optional) - As outlined in chapter 4.1.6
- Sell on close (Column O) (optional) - As outlined in chapter 4.1.6
- Stop low of day (Column P) (optional) - As outlined in chapter 4.1.6
- Sell below SMA [$] (Column Q) (optional) – In case stock was sold on close below defined value, this shows the actual fill price.
- Stop undercut [time] – timestamp when the defined stop price was undercut.
- Stop undercut – TRUE if stop was undercut.
- Crossed buy price [time] – timestamp when the buy price was crossed.
- Crossed buy price – TRUE if buy price was crossed.
- Order executed [time] – timestamp when order was executed.
- Order executed – TRUE if order was executed. See further details on order execution in chapter 6.2.
- parentOrderId – order ID of the parent order if order was executed.
- Order filled – TRUE if parent order was filled.
- profitOrderId – order ID of profit order.
- Profit order filled – TRUE if profit order was filled.
- stopOrderId – order ID of stop order.
- Stop order filled – TRUE if stop order was filled.
- sellOnCloseOrderId – order ID of sell-on-close order.
- SOC order filled – TRUE if sell-on-close order was filled.
- Spread at execution [%] – self explanatory 😊
- 2% above buy point – TRUE if buy order was filled and price increased above a certain variable percent value. See further information in chapter 6.1 for the adjustment of this value in `ConstantsAndRules.py` and 6.4 for its further functionality (“2%” is a legacy nomenclature).
- New OCA bracket [time] – timestamp if new OCA bracket is defined. For further information please see chapter 6.4.
- New OCA bracket – TRUE if new OCA bracket was defined. For further information please see chapter 6.4.
- 5% above buy point [time] – Timestamp if buy order was filled and price increases above a certain variable percent value above the buy price. For further information please see chapter 6.1 for the adjustment of this value in `ConstantsAndRules.py` and 6.5 for its further functionality (“5%” is a legacy nomenclature).
- 5% above buy point – TRUE if buy order was filled and price increases above a certain variable percent value above the buy price. For further information please see chapter 6.1 for the adjustment of this value in `ConstantsAndRules.py` and 6.5 for its further functionality (“5%” is a legacy nomenclature).

- Bad close rule [time] – Timestamp if stock attempts a bad close in the bottom x percent of the daily price range. For further information please see chapter 6.1 for the adjustment of this value in `ConstantsAndRules.py` and 6.6 for its further functionality.
- Bad close rule – TRUE if the Bad Close Rule was executed. For further information please see chapter 6.1 for the adjustment of this value in `ConstantsAndRules.py` and 6.6 for its further functionality.
- Stock sold [time] – Timestamp if the stock was sold.
- Stock sold – TRUE if the stock was sold.
- Spread above limit – TRUE if the stock crossed the buy price but the spread was above the defined limit value. For the adjustment of the maximum allowable spread, please see chapter 6.1.
- Price above limit – TRUE if price gapped up through the buy price so that the buy limit price was also crossed too fast for MTA to execute the order.
- Stock looped – TRUE in two cases:
    A. For new positions TRUE if buy price is crossed, but either the spread or price is above the limit in the first minutes of the trading day. For further information please see chapter 5.1 for the adjustment of the spread limit in `ConstantsAndRules.py` and 6.2 for its further functionality.
    B. For open positions TRUE if e.g. stock gaps down on the open. See chapter 6.3 for further details on the functionality.
- Open position bracket submitted – TRUE if bracket order is transmitted for the open position.
- Add and reduce executed – TRUE if add and reduce was executed. Please see chapter 6.7 for further information on the functionality.
- Open position updated – TRUE if open position was updated. Please see chapter 6.11.1 for further information on the functionality.
- Open position updated [time] – Timestamp if open position was updated. Please see chapter 6.11.1 for further information on the functionality.
- New position updated - TRUE if new position was updated. Please see chapter 6.11.2 for further information on the functionality.
- New position updated [time] - Timestamp if new position was updated. Please see chapter 6.11.2 for further information on the functionality.
- New position added - TRUE if new position was added. Please see chapter 6.11.3 for further information on the functionality.
- New position added [time] - Timestamp if new position was added. Please see chapter 6.11.3 for further information on the functionality.
- Stop timestamp – timestamp used for two cases:
    A. For new positions used if buy price is crossed and stocks loops as described in chapter 6.2
    B. For open positions used if stock gaps through stop as described in chapter 6.3
- Last stop price – last stop price used if stock gaps through stop as described in chapter 6.3
- Invest limit reached [time] – timestamp when investment limit is reached. For further information see chapter 5 on how to define this limit and chapter 6.10 for the description of the functionality.
- Invest limit reached – TRUE if investment limit is reached. For further information see chapter 5 on how to define this limit and chapter 6.10 for the description of the functionality.
- Position below limit – TRUE if position is below limit. For further information please see chapter 6.1

for the adjustment of this value in `ConstantsAndRules.py` and chapter 6.10 for further information on the functionality.

- Max. daily loss reached [time] – Timestamp when maximum daily loss is reached. For further information also on the adjustment of this value please see chapter 6.1.
- Max. daily loss reached – TRUE if maximum daily loss is reached. For further information also on the adjustment of this value please see chapter 6.1.
- LAST price [$] – Last price continuously updated during the trading day.
- BID price [$]– Bid price continuously updated during the trading day.
- ASK price [$]– Ask price continuously updated during the trading day.
- BID size – Bid size continuously updated during the trading day.
- ASK size – Ask size continuously updated during the trading day.
- HIGH price [$] – Daily high price.
- LOW price [$]– Daily low price.
- CLOSE price [$]– Daily close price.
- Volume – Volume continuously updated during the trading day.
- MAX_STOCK_SPREAD – Max. allowable stock spread. For further information please see chapter 6.1.
- SELL_HALF_REVERSAL_RULE – Value for Sell Half Rule. For further information please see chapter 6.1 and chapter 6.4.
- SELL_FULL_REVERSAL_RULE - Value for Sell Squat Rule. For further information please see chapter 6.1 and chapter 6.5.
- BAD_CLOSE_RULE – Value for Bad Close Rule. For further information please see chapter 6.1 and chapter 6.6.
- MAX_ALLOWED_DAILY_PNL_LOSS – Max. allowable daily loss. For further information please see chapter 6.1.
- MIN_POSITION_SIZE – Min. position size. For further information please see chapter 6.1 and chapter 6.10.
- Bad close checked – TRUE if Bad Close Rule was checked. For further information on the Bad Close Rule, please see chapter 6.6.
- liquidHours – Trading hrs of the asset (not used by MTA so far – update in development).
- timeZoneId – Timezone of the asset (not used by MTA so far – update in development).
- local opening time – Local opening time of the asset (not used by MTA so far – update in development).
- local closing time – Local closing time of the asset (not used by MTA so far – update in development).

# 5 Starting MTA

It is recommended but not required to start MTA at any time before the market opening. The first row BTC of `DailyTradingPlan.xlsx` is recommended not to be changed to assure program stability (see also chapter 4.1.6 and chapter 7).

Open the cmd prompt or your favorite IDE and locate the folder where the program is saved e.g. `through cd documents\foldername` on windows.

Start the program through its file name e.g. `python main.py`.

Define which market you want to trade, see Figure 1. Since MTA can only trade one market at a time, if you

seek to trade e.g. the German and US market in parallel, you need to prepare two `DailyTradingPlan.xlsx` and run MTA twice in parallel. For further information see also chapter 4.1.6.

```
Do you want to trade New York [NY], Japan [JP] or Germany [DE]?
NY
```

*Figure 1 - Which market do you want to trade?*

Afterwards, an extract of your `DailyTradingPlan.xlsx` will be printed and you will be asked to confirm that the `DailyTradingPlan.xlsx` is correct with "y" in case, see Figure 2.

```
   Symbol  Entry price [$]  Stop price [$]  Quantity [#]  Profit taker price [$]  Open position
0     BTC             9.00           11.00             1                   11.25          False
1    AAPL           227.71          223.95            15                  245.92           True
2    NFLX           708.91          695.00             3                  779.80           True
3    NVDA           117.20          117.20             8                  128.92           True
4    NVDA           117.20          113.00             7                  134.78           True
5    LDOS           160.35          156.90            10                  176.39          False

Have you updated the DailyTradingPlan on the server? [y/n]
y
```

*Figure 2 - Is DailyTradingPlan.xlsx up-to-date?*

Then define what %-invested you would like to go as a maximum for the day, see Figure 3. The entered value can be $0 < x < \infty$ and therefore can also be used for a margin account. Please see further information on the significance of this value in chapter 6.10.

```
What is your maximum you want to go in the market today [%]?
100
```

*Figure 3 - Maximum %-invested for the day*

Finally, chose the indices of the correct market opening and closing hours while considering that indices start counting at 0, see Figure 4.

```
 ['20240930:0930', '20240930:1600', '20241001:0930', '20241001:1600', '20241002:0930', '20241002:1600',
'20241003:0930', '20241003:1600', '20241004:0930', '20241004:1600']

Enter the index of the next market OPEN: 0

Enter the index of the next market CLOSE: 1
```

*Figure 4 - Choose next market opening and closing time*

The program then starts its work without any further inputs required.

To assure a flawless start, it is recommended to check also the following based on the given prints in the cmd (or similar):

- Are the company names correct based on the entered stock data?
- Is the printed Account Information e.g. in terms of current %-invested correct?
- Are the market opening hours correctly defined for the day?
- Are there any other noticeable deviations to normal operation?

As a further support of your trading in case of US-market stocks, the code provides an information about the days to the next Earnings Calls of the companies mentioned in your `DailyTradingPlan.xlsx`, see Figure 5 as an example.

```
The earnings dates for your focus stocks are:
  Symbol      Earnings Date Days to Earnings
2   NFLX   October 17, 2024              17
1   LDOS   October 29, 2024              29
3   AAPL   October 31, 2024              31
0   NVDA  November 27, 2024              58
```

*Figure 5 - Next Earnings Calls*

The code gives you a warning if one earnings call is only three days or less away. Further, MTA checks if the open positions as defined in your `DailyTradingPlan.xlsx` are matching the open positions in your portfolio for the given currency (and therefore market) of the assets. In case this matches, the confirming message is shown in Figure 6. If this is not the case, MTA gives you a warning. This shall assure that all positions will be covered by a bracket order.

```
DailyTradingPlan matches current open portfolio positions for currency: USD
```

*Figure 6 – DailyTradingPlan.xlsx matches open portfolio positions*

If there are any issues during this starting phase of MTA, consider interrupting the code e.g. through STRG + C and restarting it with updated information.

If you start MTA the first time on your account, you should delete all open orders related to your stock holdings for this market. MTA can only receive order IDs of open orders which were previously sent with the same client ID. If this is not the case, it will lead to the definition of unnecessary bracket orders and impair also others of MTA's functionalities.

# 6 Functionalities of MTA

MTA has a lot of functionalities which are automated to support your trading as per predefined rules. These rules are hard-wired, but can be deactivated respectively be adjusted as per your preferences for each trading day. This chapter describes how such adjustments can be done and what they are. For deactivation, just define the constants outside the usual range, e.g. the allowable spread at 1 (= 100%).

## 6.1 Constants to adjust in ConstantsAndRules.py

As described in chapter 1, this code allows trading as per your pre-defined trading rules and adjustments to the same based on your trading results and post-analysis.

The possible adjustments can be found in `ConstantsAndRules.py` as shown in Figure 7. Further, for documentation reasons they are saved in your `DailyTradingPlan.xlsx` output file (see chapter 4.2.3).

```
# Constants and rule are defined here
port = 7497   # for paper trading
# port = 7496
max_stock_spread = 0.0125
sell_half_reversal_rule = 0.06
sell_full_reversal_rule = 0.1
bad_close_rule = 0.15
max_allowed_daily_pnl_loss = -0.05
min_position_size = 0.001
portfolio_update_prints = 0.1
```

*Figure 7 - Adjustment of trading rules*

The possible adjustments are:

- MAX_STOCK_SPREAD – Max. allowable stock spread as decimal value. When the stock crosses the defined buy-price, the trade will not be executed if the spread is above this limit. For further information see also chapter 6.2.
- SELL_HALF_REVERSAL_RULE – Decimal value for application of Sell Half Rule as further outlined in chapter 6.4.
- SELL_FULL_REVERSAL_RULE - Decimal value for application of Sell Squat Rule as further outlined in chapter 6.5.
- BAD_CLOSE_RULE – Decimal value for application of Bad Close Rule as further outlined in chapter 6.6.
- MAX_ALLOWED_DAILY_PNL_LOSS – Decimal value for maximum allowable daily loss as percentage of your total account value. If this value is reached, MTA will stop any further buying.
- MIN_POSITION_SIZE – Decimal value for minimum position size to be bought as percentage of your total account value. For further information please see chapter 6.10.

MTA's base currency is USD. In case German or Japanese markets are traded with MTA, from time to time the exchange rates need to be manually adjusted in `ConstantsAndRules.py`. These can be found directly below the code snipped as shown in Figure 7 as EXR_RATE. They are currently defined as 150 YEN/USD and 0,91 EUR/USD.

## 6.2 Order execution for new positions

MTA offers several checks and balances before a buy order is transmitted to TWS. A buy order is only executed if:

- The maximum daily loss has not been reached, see chapter 6.1
- The daily investment limit has not been reached, see chapter 6.10

- The stop price has not already been undercut
- The stock crosses the buy price for the first time
- The spread is below the defined limit, see chapter 6.1
- The stock price is below the defined buy limit price

In the first minute of the trading day, the stock will loop if the spread and/ or the buy limit price is exceeded.

As soon as all of the above is given, the order will be send to TWS and "Order executed" will be marked as TRUE in `DailyTradingPlan.xlsx`. If TWS confirms that the order has been filled, also "Order filled" is market TRUE and the new position is covered by a bracket order with the defined stop loss and profit taker prices.

## 6.3 Transmission of bracket orders for open positions

MTA defines all open orders new for each trading day. Therefore, 15 minutes before the market opens, MTA deletes all orders currently defined in TWS if they have previously been defined by the same client (see further information on connecting MTA to TWS for the first time in chapter 5).
As soon as the market opens, bracket orders are defined for all open positions as per the `DailyTradingPlan.xlsx`.
MTA checks if the current stock price for open positions is max. -1% below the defined stop price. If this is the case, the bracket order is placed immediately. If not, e.g. when the stock price gaps down on the open, MTA waits for 4 seconds. If the price falls any further, the stock is sold immediately via a market order. If the price remains the same or increases, MTA will wait another 4 seconds. This loop continues until the price falls further again and the position is closed.
This function seeks a more favorable exit compared to a simple sell stop order by profiting from an immediate recovery of the price in case. Further, the stock spread tends to decrease within the first seconds of market opening which may also lead to a more favorable exit price.

## 6.4 The Sell Half Rule

The Sell Half Rule and the Sell Squat Rule give you the opportunity to sell half or all of your new purchase in case of a squat, which is a very Minervini-specific nomenclature. Both rules are only applied on the day of the purchase in case. Herein, a squat is defined as a price breakout above the pivot price which subsequently comes back in again.
With the constant SELL_HALF_REVERSAL_RULE you define the percentage move a stock needs to make above your buy price, so that half of your holding will be sold when the stock subsequently comes back to break-even on the same day. For example, you define SELL_HALF_REVERSAL_RULE as 0,06 as shown in Figure 7. You buy a stock at 100 USD, it continues higher above 106 USD and then comes back to 100 USD on the same day. MTA would sell half your position there and secure the remaining half with a new bracket order. If this rule triggers, your output file yymmdd_trading_plan (see chapter 4.2.3) will show the related timestamp in column "New OCA bracket [time]" and TRUE in columns "2% above buy point" and "New OCA bracket" ("2%" is a legacy nomenclature and does not have any further relevance).
In case you want to deactivate this rule, just define SELL_HALF_REVERSAL_RULE e.g. as 10.

## 6.5 The Sell Squat Rule

In analogy to the Sell Half Rule of chapter 6.4, the SELL_FULL_REVERSAL_RULE defines the price advance from your buy price the stock needs to make, so that your full position would be sold when the stock comes back to your break-even point on the same day.
For example, you define SELL_FULL_REVERSAL_RULE as 0,1 as shown in Figure 7. You buy a stock at 100 USD, it continues higher above 110 USD and then comes back to 100 USD on the same day. MTA would sell your position there since the sell stop order was already increased to break-even when the stock price hit 110 USD.
If the price increases above your SELL_FULL_REVERSAL_RULE, your output file yymmdd_trading_plan (see chapter 4.2.3) will show the related timestamp in column "5% above buy point [time]" and TRUE in column "5% above buy point" ("5%" is a legacy nomenclature and does not have any further relevance). If the stock came back to break-even and was sold, you will also see TRUE in column "Stock sold".
It makes therefore sense to define SELL_FULL_REVERSAL_RULE > SELL_HALF_REVERSAL_RULE.
In case you want to deactivate this rule, just define SELL_FULL_REVERSAL_RULE e.g. as 10.

## 6.6 Bad Close Rule

The Bad Close Rule sells half of your position at the end of the trading day when the stock attempts a bad close for a new position you bought that day. Herein, BAD_CLOSE_RULE defines the percentage of the daily price range, e.g. BAD_CLOSE_RULE of 0,15 would sell half of your position if the stock attempts to close in its bottom 15% of the daily range. This is checked 2 minutes before the end of the trading day. A new bracket order is immediately defined to secure your remaining position in case half was sold.
This rule will only be applied if it is a new position and the Sell Half Rule (see chapter 6.4) respectively the Sell Squat Rule (see chapter 6.5) have not been triggered.

## 6.7 Add and Reduce

Add and Reduce is a very Minervini-specific nomenclature. Let's say you bought 100 shares of XYZ company at 100 USD and your stop is at 95 USD. Some days later a new pivot is formed higher at 110 USD which allows you a technical stop of 105 USD. When you add to your existing position of XYZ company when Add and Reduce is activated (see chapter 4.1.6 on how to do that), the stop price for your original 100 shares will also be increased to 105 USD when the new buy order is executed. The name Add and Reduce stems from the fact that this technique allows you to add to your position while reducing your risk.

## 6.8 Sell On Close

If Sell On Close is activated (see chapter 4.1.6 on how to do that), your stock is sold 5 minutes before the end of the trading day. This may e.g. make sense if there is an Earnings Event after the market close. Sell On Close can also be applied for new positions which are opened the same day, which effectively allows you also to day trade and go flat over night.

## 6.9 Sell Below SMA

The Sell Below SMA (simple moving average) only works in conjunction with Sell On Close (see chapter 4.1.6 on how to activate it).
Sell Below SMA allows to sell the stock shortly before the market close if the stock price is below the defined

value, which could be a SMA but can also be any other value.
For example, you bought a stock at 100 USD and it increased to 110 USD during the next trading days. You now want to sell it when it attempts to close below the 10-day SMA, which is at 107 USD.

*Table 3 - Sell Below SMA correctly defined*

| | Symbol | Company name | Security Type | Currency | Exchange | Primary Exchange | Entry price [$] | Stop price [$] | Quantity [#] | Buy limit price [$] | Profit taker price [$] | Open position | Add and reduce | Sell on close | Stop low of day | Sell bellow SMA [$] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | BTC | | CRYPTO | USD | SMART | PAXOS | 9.00 | 11.00 | 1 | 9.01 | 11.25 | | | | | |
| 1 | XYZ | | STK | USD | SMART | NASDAQ | 100.00 | 95.00 | 100 | 101 | 120 | TRUE | | TRUE | | 107 |

Sell Below SMA checks 8 minutes before the end of the trading day if the stock is below 107 USD. If so, MTA will sell the stock 5 minutes before the market close. If the stock value is above 107 USD, the stock will be held.

## 6.10 Daily investment limit

The definition of the daily investment limit is especially useful when MTA is running when you are not in front of the screen. Before a new buy order is executed, MTA checks if this position would exceed the daily investment limit. The daily investment limit is defined within e.g. the cmd when MTA is started, see also chapter 5.
For example, you are 50% invested with your current holdings and you define five new positions in `DailyTradingPlan.xlsx`, each of which would be 10% of your account. You start MTA and define your daily investment limit at only 75% to cap your exposure and risk for this day. During the first trading hour, already 2 stocks cross their buy prices so that you are now 70% invested. As soon as a third stock crosses its buy price, your daily investment limit will be reached. In this case, the third position will be reduced to 5% of your account value so that you are at 75% invested as you defined. The MIN_POSITION_SIZE (see chapter 6.1) defines the smallest size of a new position to be entered. In the example of 0,001 as shown in Figure 7, a minimum of a 0,1% position would still be bought which might be only a couple shares depending on your account size.
In case another stock will cross the buy price, no buy order will be executed.

## 6.11 Updating the DailyTradingPlan during market opening hours

MTA can also be used during market opening hours as a live-trading assistant. This is possible until 5 minutes before the market close. The `DailyTradingPlan.xlsx` is being read by MTA every 15 seconds and compared for changes. Just copy an updated version of the `DailyTradingPlan.xlsx` into your designated file location.
It is only possible to change or add lines to the `DailyTradingPlan.xlsx`, not to delete lines. In case a new position shall not be purchased, just set the Quantity to 0. All possible updates t `DailyTradingPlan.xlsx` are outlined in the following sections.

### 6.11.1 Updating open positions or executed new positions

For open and executed new positions, the following data can be updated:

- Stop price (can be changed up and down from your original value)
- Profit taker price (can be changed up and down from your original value)
- Quantity (can only be changed down from your original value to trim the position)

- Sell on close (can be set from `False` to `True` and vice versa)
- Stop low of day (can be set from `False` to `True`)
- Sell bellow SMA price (can be added and changed up and down)

The code will detect those changes and set a new bracket order accordingly. In case you decrease your Quantity, a market sell order will be transmitted and a bracket order defined for the remaining position. Feedback when this is done can be seen in your cmd window.

### 6.11.2 Updating new positions that are not executed yet

This section only applies if the buy price has not been crossed yet. If so, please refer to chapter 6.11.1.
The following data can be updated in case the new position is still not executed:

- Entry price (can be changed up and down from your original value)
- Stop price (can be changed up and down from your original value)
- Quantity (can be changed up and down from your original value)
- Buy limit price (can be changed up and down from your original value)
- Profit taker price (can be changed up and down from your original value)
- Sell on close (can be set from `False` to `True` and vice versa)
- Stop low of day (can be set from `False` to `True` and vice versa)
- Sell bellow SMA price (can be added and changed up and down)

The code will detect those changes and consider them accordingly. Feedback when changes are recognized can be seen in your cmd window.

### 6.11.3 Adding new positions

For adding new positions to the `DailyTradingPlan.xlsx`, add a new row at the end of the file. The code will detect that there is a new row added, will request market and contract data for the stock and consider it accordingly. Feedback when this is done can be seen in your cmd window.

# 7 Program stability

It must be assured that MTA is running and has a continuous internet connection throughout its time of application. Therefore, please ensure that your computer does not go to sleep mode or shut down as well as that your internet connection is stable. I was able to increase MTA's stability to a maximum through shifting TWS and MTA to a cloud computer. Further, as previously described, it is recommended to leave the first row of `DailyTradingPlan.xlsx` (BTC data stream) unchanged to avoid the code "falling asleep".

# 8 Contribution to this project

Your feedback as well as contribution is most welcome. This project shall serve to support our trading and improve our trading results.
Proposed next improvements are:

- Slim down `main.py` through outsourcing of logical tests to `MyFunctionalities.py`
- Increase focus on OOP standards (to be pursued soon in separate fork)

Therefore, brains-on please 😊

# 9 Software support

If you encounter any errors or uncertainties resp. ambiguities towards MTA's usage, please feel free to come back to me any time in case I did not cover this topic in the user manual.