***“Live\_Traing\_App.py*” – Documentation**

# Description:

This file contains a Live\_Trade\_App class that handles communication with the MT4 platform including requesting and receiving current position holdings, subscribe and collect live market data, trade executions. Live Trading App connects with Risk Advisor and Performance Advisor to give a live-updating demonstration of the portfolio risk metrics and return attributions. It also connects with the database to add latest price data into the database.

# Dependencies:

* pandas
* matplotlib
* threading
* time
* IPython
* Random
* DWX\_ZeroMQ\_Connector
* Risk\_Advisor
* Performance\_Advisor
* DB\_Operator
* Any further dependencies of Risk and Performance advisors or DB\_Operator

# Class Function Descriptions:

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Inputs | Output | Description |
| Constructor | Pool\_Limit (int),  Update\_frequency(int),  \_Time\_Out (int),  \_Refresh (double),  \_TimeZone (int) | None | Upon constructing the live\_traing\_app class I set several parameters that will be applied through different functions.  (Pool Limit) means the size limit of live data pool, before it is recorded and cleared.  (Update\_Frequency) is the frequency of requesting portfolio holding and recalculating all risk/performance metrics.  When requests are sent these requests are refreshed per (\_Refresh) frequency if we didn’t hear back. Then they would be timed out at (\_Time\_Out) limit to prevent infinite waiting. Price information heard back will be adjusted to our local (\_TimeZone) before they are used and recorded to the historical database |
| \_Current\_Holding | Every (int) | None | This function takes care of requesting portfolio holding per (every) seconds, and record these holding, as well as request a lots/units translator and request portfolio equity level. These information are recorded and used by Risk/Performance Advisor.  This is an internal function. |
| \_Risk\_Performance\_Monitor | Every (int) | Print (str)  And  DataFrame (pd.df)  And  Pie Chart (plt)  Or  Bar Chart (plt) | This function calls the Risk and Performance Advisors to be constructed and the metrics to be shown. Details of what these metrics are is documented in other documentations.  All live demonstrations are then updated per (Every) seconds. |
| \_Baby\_Strategy | Every (int) | None | This Function is a toy trading strategy that’s there to trade randomly. I wrote it to show all other parts of Live\_Trading\_App works smoothly. |
| Trade\_Execute | Action (str),  Symbol (str),  \_type (int),  Lots (double, optional),  Price (double, optional),  Strategy\_Number (int, optional),  Tkt (int, optional),  SL(double, optional),  TP(double, optional) | None | This function executes trades of the given (Action : OPEN/CLOSE) on the given (Symbol), at given (\_type: long = 0/short = 1), with any other optional specifications. (lots, take profit, stop loss, limit price etc.)  Then this function feeds back whether the execution has been successful. |
| \_Price\_Buffering | Symbol (list),  Stop (bool) | None | This is an internal function that send request to either start live price subscription or terminate subscription (Stop), of Assets we’re interested in (Symbol). |
| \_Snapshots | symbol (list),  every (int) | None | This function takes a snapshot of streaming price and volume per (every) seconds, add it to the price buffer. When buffer reach the (Pool\_Limit) size it is added to the database. |
| Print\_Price\_Buffer | None | Pd.df | This function returns the price buffer |
| \_Check\_Connection\_status | None | None | This is an internal function that’s called when the Live\_Trading\_App was constructed to ensure all three sockets are successfully connected to the MT4 trading platform |
| Run | None | None | This is the main user function. This function spins off four threads to serve four purposes: 1. Constantly updating portfolio holding information. 2. Constantly outputting the updated risk and performance metrics as the holdings change. 3. A trading bot that trades with random strategy. 4. Constantly hearing the live streaming price and add it to the database. |