Trader without Adaptive Trailing

1. Prerequisites:-
   Data connection feed from Database (MySQL/ MariaDb).
   Python modules – MySQLdb, numpy, math, time.

2. The Input data table should have the following fields for each tick:- Tick_Date
   - Tick_Time
   - Open
   - High
   - Low
   - Close
3. It creates 3 Output tables in the DB to which input data table belongs.
   - OHLC table
   - Position/Results table
   - Signal table

   It also creates a Rina CSV file in the same directory of the code.

4. The code creates the required tables if they don't exist. The code has all sql commands included in it for creating the tables if they don't exist and delete the data from the table if they exist previously.
   Also, the Rina CSV file is created if it does not exists else the previous content is flushed.

5. The Output tables are as follows:-
   - OHLC table: It stores the OHLC values with corresponding Date and Time for each bar formed for the period for which the code runs.
   - Results Table: It contains the Date, Time and Position for each bar. The position can be 1(Long), -1(Short) and 0(Neutral).
   - Signal Table: It contains the Signals to be executed. Each entry stores the information - Date, Time, Price, Type of Trade, Quantity and Remarks.
   - The Rina CSV file is generated in the specified format at the end of trading session.

6. The code reads from a configuration file named 'ConfFile.txt'. The file should contain the following as specified :-

- PriceDBName (Name of the data database)
- PriceTable (Name of the data table in data database)
- PriceDBUserName (Username to access the data database)
- PriceDBPassword (Password to access the data database)
- TradeDBName (Name of the result database)
- TradeDBUserName (Username to access the result database)
- TradeDBPassword (Password to access the result database)
- ResultTable (Table to store the positions)
- SignalTable (Table to store the trading signals)
- OHLCTable (Table to store OHLC bars)
- RinaFileName (Name of Rina CSV)
- BarSize (Size of the bar to trade with)
- SessionBegin (Session Begin Time)
- SessionEnd (Session End Time)
- LiveMode (True to run code in live mode)
- RunStartDate (Date to start the code from)
- RunStopDate: (Date to run the code till)
- RestartFlag (True to restart the code after crash)
- RunRestartDate (Date to restart the code from)
- RunRestartTime (Time to restart the code from)

NOTE:

1. All dates and times are needed to be specified in the format of date and time of the data table.
2. RunStopDate needs to be specified only in testing mode. Its value, if specified, has no significance during live mode. It is better to leave it blank during live mode.
3. RunRestartDate and RunRestartTime need to be specified if the code has to be rerun after a crash. Their values have no significance if restart flag is false. It is better to leave them blank when the code is started afresh.
4. RunStartDate and RunRestartDate are to be specified when the code is rerun. RunStartDate is the date that the code had started from, before it crashed. RunRestartDate is the date that you will resume from. Also RunRestartTime is the time that you will resume from.
   For e.g.: If we started the code from 26 Jan 1950 and it crashed on 15 Aug 1950 at 1015 hours and it has to be resumed from 16 Aug 1950, 1016 then
   RunStartDate: 26 Jan 1950
   RunRestartDate: 16 Aug 1950

RunRestartTime: 1016
And, this configuration will give me next position for 16 Aug 1950 1030 hours (Assuming 15 as BarSize).