Documentation for AlpacaFarmP program

Built for strategies with trading times greater than 10 minutes, less than six hours.

* Overall
  + *Portfolio* class contains all function for buying, selling, or otherwise interacting with the Alpaca API
  + Portfolio keeps track of holdings with an internally-referenced list of positions and periodically syncs to the actual positions held in the program
  + Entire portfolio class object is regularly pickled, can be dropped and used again at any time without desynchronization or other problems
  + RUN block calls sync, buy and sell functions of portfolio object in a cycle
* Buy/Sell rules
  + Takes dataframe from Alpaca barset call as an input, returns True or False to buy/sell or not
* Other functions
  + liquidate: closes all positions with requests.delete. Very important function.
  + Sends delete request twice to be EXTRA thorough
* Portfolio objects
  + Pos
    - Dataframe with each row representing a held position
    - SYM: ticker symbol
    - Qty: number of shares
    - Side: 'long' or 'short'. Only long for now, but might as well build it in
    - Entry: base price when bought,
      * 'avg\_entry\_price' in alpaca
    - Base: highest price reached by asset yet since bought.
      * Constantly updates. Used for step-up sell floor calculations
    - Current: current price
      * 'current\_price' in Alpaca
    - UPLPC: unrealized percent gain/lost.
      * 'unrealized plpc' in Alpaca
      * Constnatly updates. Used to record sells
    - EntryTime: datetime of time bought.
      * Filled by SyncPositions function, assumes the time that this function is called is more or less equal to the time bought
    - Favor:
      * Favor value from WatchList
  + Watch
    - Dataframe with each row representing an asset tradable with the Portfolio
    - Symbols in Pos are also in Watch
    - SYM: ticker symbol
    - Cooldown: time at which the asset become buyable again
      * Default is 04-09-1996
    - Ban: True/False whether to touch the asset or not. Use to "shut down" parts of the watch if we're trying to restrict to a certain trade list
    - Favor: integer, sum of all total UPLPC's for trades with this symbol.
      * Default is 1 (Because they can't sum to 0)
      * Used with dataframe.sample to weight random selection for the buy rule
  + Various internal time parameters
* Portfolio functions
  + addToWatch
    - Puts symbols from a SYM.txt file into the watch list with default values
  + SyncPositions
    - Calls position list from alpaca API, compares to internal positions list object
    - adds any external positions not internalized
    - Removes any internal positions not externalized
    - Updates values for each position object that are expected to change (current price, mainly)
    - Finds favor from watch, adds to watch if not in watch
  + BuyWatch
    - Iterates through *N* assets in the watch list
      * 100 is probably safe
    - Uses *BuyStrat* on the bartset data received from alapaca to decide if it should buy
    - Decides on the target amount of cash to use with *NPos*, the desired number of positions in the account
    - Prints each buy move if *PrintTrades* = True
    - Doesn't actually send a buy order to Alpaca if *Loaded* = False
    - Does not buy is actual cash available is less than the target cash
    - Uses ordinary market orders
      * No need for fancy stuff unless you're at the micro scale (<5 min)
    - Does not actually update portfolio positions list; that's left to the SyncPositions function
  + SellPos
    - Iterates through each position in position list, applying *SellStrat* to decide if it should sell
    - Simultaneously, applies a sell floor if *Floor* = True.
      * Means if the current price is *FloorPC* percent below the base price, it sells.
    - Also closes the position if it is held longer than *MaxHoldTime*, if *MaxHold* = True
    - Will not sell by any condition if *MinHold* is True and the position has been held a shorter time than *MinHoldTime*
    - Logs the following string if *Log* = True in *SellLogCSV*
      * Sell,SellStrat,Symbol,EntryTime,SellTime,PLPC
        + Sell can be "sell" for normal sell triggers or fsell for "hard" sells, or tsell for expiration time sell
        + PLPC is last UPLPC of position, is approximate
    - Makes assets in the watch list untradable until *CoolTime* into the future if they are sold by the floor condition
    - Increases/decreases Favor value of symbol in Watchlist by UPLPC value at time sold if *ChangeFavor* = True
    - Also uses market orders
    - Also relies on SyncPositions to actually reflect changes in the position list.
    - Sometimes tries to sell a position that doesn't exist because the sell order has yet to run through Alpaca. Except block avoids the resulting crash, SyncPositions will eventually correct for the discrepancy
* RUN block
  + Checks account status, breaks loop if can't trade
  + Checks to see cash in account is greater than zero. If not, sells worst position in the positions list by UPLPC.
    - Cash can go less than zero if the program buys without real cash because the previous buy order hasn't gone through yet
  + SellPos, then BuyWatch because it's more important sell positons update before the next sync
  + Waits at least 60 seconds before looping to avoid overloading the API. This step is placed before synpositions() to allow for maximum time for Alpaca to update
  + SyncPositions
  + Will wait without breaking the loop if the market is not open, but only after doing one cycle and queueing up some orders