Brain Mind and Markets Lab

THE UNIVERSITY OF MELBOURNE

FACULTY OF BUSINESS & ECONOMICS

FNCE30010

Algorithmic Trading

Semester 2, 2018

Project 1 - Task 2 (CAPM)

Due **11:59 pm on Friday, 21 September 2018**

Nitin Yadav and Peter Bossaerts

Brain, Mind & Markets Lab.

Department of Finance.

Faculty of Business and Economics.

The University of Melbourne.

Administrative Arrangements

Due date:	**11:59 pm on Friday, 21 September 2018**
Late assignments:	Late submission assignments may attract a penalty unless an extension has been granted. All extension requests must be made prior to the assessment due date and supported with appropriate documentation. Unless an extension has been granted, penalties to the assessment will be applied. For assignments submitted after the due date, the mark awarded will be reduced by 10% for each day the work is late. For assignments submitted later than 5 working days after the due date will not be marked and will receive no marks.
Where to submit:	Assignment submission is via the LMS Assignment Submission link for all written assignments. Please refer to the LMS Student Guide: Turnitin Assignments for detailed submission instructions, if needed. **Only one member of the group has to upload** (see Group Size below).
Group size:	For undergraduates the maximum number in each group is 3. For postgraduates this is an individual assessment.
Submission Metadata:	You should include your details as part of your submission. **Subject Code and Name** **Student Numbers of all members in group** **Full Names** **Assignment Name**
Word limit:	FBE usually as a word limit. This assignment is for computer code, however, and hence word limits do not apply.
Marks:	This assignment counts 15% towards the final mark in this subject.

The assignment must be your group's own work. Students are encouraged to discuss the assignment and to share information sources. However, the writing of each group's assignment must be conducted separately and independently.

Assignment Instructions

Trading Setting (Identical to "Instructions" for CAPM Manual Experiment)

You will be endowed with certain amount of cash and a certain amount of securities (upto 3 risky and 1 risk-free security). You will be able to trade these for 20 minutes. Short sales is NOT allowed.

After markets close, the securities pay a liquidating dividend, depending on the drawing of one of the possible four states, W, X, Y, or Z. These states are equally likely to occur. Liquidating dividends depend on the states (an example of possible values is shown below). The value of liquidating dividends for a security in particular state will be made available in the market's description.

If state is	W	X	Y	Z
Stock A	10	0	7.5	2.5
Stock B	0	2.5	7.5	10
Stock C	0	7.5	2.5	10
Note	5	5	5	5

Your performance for this trading session will be based on the final composition of your portfolio. We will evaluate how high the expected payoff is, while penalising risk (payoff variance), as follows:

*Performance = Expected Payoff - b * Payoff Variance*, where b (penalty for risk) = 0.01

Task Specification

For this task, please use the robot class template CAPMBot provided on the LMS. The template has two predefined variables to model the risk penalty and the total trading time in one session. The provided template also provides code that populates the variable `_payoffs` with the respective dividends (in cents) of securities in each possible state. If you add new functions in the given file, please make sure you prefix your functions with an underscore ("_"). You should not remove any function or variables from the provided template.

1. Trading and bot parameters .... [1 mark]
    a. Trading account details. Change the variables that start with `FM_` prefix (e.g., `FM_ACCOUNT`) to the Adhocmarket details of the submitting member.
    b. Group details. Change the `GROUP_MEMBERS` variable to the details of the group members.
    c. Market ids. You should extract market ids and save them in the variable `self._market_ids`. This variable should be a dictionary with key as the market item and value as the market id. You should use this variable to get market_id when required (e.g., sending an order).
2. Trading style .... [2 marks]
   The CAPMBot is free to proactively send orders or react to existing orders. The bot should dynamically switch its trading style based on the time left in the market and/or based on the existing orders in the order book.
3. Performance measurement .... [4 marks]
    a. The template CAPMBot has an (empty) method called `get_potential_performance`. This method takes a list of orders as an argument and is meant to return the portfolio performance if the given orders should trade. You should complete this method and use it to evaluate orders that are profitable to send to the marketplace.
    b. Complete the method `is_portfolio_optimal`. This method should return True if the current holdings are optimal (i.e., the bot has an optimal portfolio with respect to the performance formula). If the holdings are not optimal, this method should return False.
4. Order management .... [5 marks]
    a. Order generation. Based on the current holdings, the bot should only send orders that improve its portfolio performance. You can use any valid approach to generate such orders. There are no restrictions on the number of units in an order.
    b. Order management. The bot should be able to participate in multiple markets simultaneously (e.g., by sending a buy order for stockA and/or sell order for

stockB). If one of the orders get partially or fully filled, then the bot should re-evaluate the next set of profitable orders and manage the pending orders accordingly.

   c. Order validity. Before sending an order the bot should check for order validity (e.g., does it have enough units to sell, or enough cash to buy).

5. Code quality .... [3 marks]

   Your submitted code must be clear to understand and well commented. Please read the following links for suggestions:

   – https://docs.python-guide.org/writing/style/
   – https://www.python.org/dev/peps/pep-0008/

Note: You are allowed to reuse your own code from your previous induced-demand supply task. You can also use numpy library to help with implementation of some of the functions.