MC2-Project-1

link:

http://quantsoftware.gatech.edu/MC2-Project-1

Questions on MC2-project 1:

 "directory structure" -- [Here](http://quantsoftware.gatech.edu/MC2-Project-1#Directory_structure.2Ftemplate)it is on wiki

"Will the Udacity enviorment be set up to test this project": We are looking into it but DO NOT REPLY ON THEM.

"Stuck on how to start": Watch Prof explain it on Google hangout : Here it is

# MC2-Project-1

## Contents

 [[hide](http://quantsoftware.gatech.edu/MC2-Project-1)]

* [1 Updates](http://quantsoftware.gatech.edu/MC2-Project-1#Updates)
* [2 Overview](http://quantsoftware.gatech.edu/MC2-Project-1#Overview)
* [3 Template](http://quantsoftware.gatech.edu/MC2-Project-1#Template)
* [4 Part 1: Basic simulator (95%)](http://quantsoftware.gatech.edu/MC2-Project-1#Part_1:_Basic_simulator_.2895.25.29)
* [5 Part 2: Leverage (5%)](http://quantsoftware.gatech.edu/MC2-Project-1#Part_2:_Leverage_.285.25.29)
* [6 Orders files to run your code on](http://quantsoftware.gatech.edu/MC2-Project-1#Orders_files_to_run_your_code_on)
* [7 Short example to check your code](http://quantsoftware.gatech.edu/MC2-Project-1#Short_example_to_check_your_code)
* [8 More comprehensive examples](http://quantsoftware.gatech.edu/MC2-Project-1#More_comprehensive_examples)
  + [8.1 orders.csv](http://quantsoftware.gatech.edu/MC2-Project-1#orders.csv)
  + [8.2 orders2.csv](http://quantsoftware.gatech.edu/MC2-Project-1#orders2.csv)
* [9 Implementation suggestions & assumptions](http://quantsoftware.gatech.edu/MC2-Project-1#Implementation_suggestions_.26_assumptions)
* [10 What to turn in](http://quantsoftware.gatech.edu/MC2-Project-1#What_to_turn_in)
* [11 Rubric](http://quantsoftware.gatech.edu/MC2-Project-1#Rubric)

## Updates

**09/28/2015**: Directory structure/template - please refer to [Template](http://quantsoftware.gatech.edu/MC2-Project-1#Template) section below. Note that order files are available in ./orders/ within mc2\_p1. Stock price data is going to be in ../data/ as usual. The includedutil.py should read data correctly.

**09/30/2015**: Several changes have been made to the input and output requirements for compute\_portvals(), and the format of the CSV order files. The template has been updated accordingly.

* Fixed example output for orders-short.csv, orders.csv and orders2.csv shown on this wiki page.
* Collapsed the separate year, month, day columns in order CSVs into a single Date column in yyyy-mm-ddformat so that it is easy to read in directly with pandas.
* Added a header row to CSV files with column names: Date, Symbol, Order, Shares (see details below).
* Changed all Order column values to be UPPERCASE: Buy/Sell -> BUY/SELL
* Clarified input parameters: start\_date and end\_date will be supplied as strings in yyyy-mm-dd format; however, if you have already implemented compute\_portvals() to accept datetime objects, we will evaluate it accordingly when grading.
* Clarified output type: The return value of compute\_portvals() should ideally be a pandas.Series (since there is only one column). But you can also return a single-column pandas.DataFrame, and the first column will be extracted for grading (the name of the column does not matter).

## Overview

In this project you will create a market simulator that accepts trading orders and keeps track of a portfolio's value over time and then assesses the performance of that portfolio.

## Template

Instructions:

* Download [**mc2\_p1.zip**](http://quantsoftware.gatech.edu/images/6/63/Mc2_p1.zip), unzip inside ml4t/
* Copy analysis.py (your solution for MC1-Project-1) to mc2\_p1/portfolio/.
* Implement the compute\_portvals() function in mc2\_p1/marketsim.py.
* To execute, run **python -m marketsim** from mc2\_p1/ directory.

## Part 1: Basic simulator (95%)

Open the provided template file: marketsim.py. Your job is to implement a function, your market simulator,compute\_portvals() that returns a dataframe with one column.

It should adhere to the following API:

def compute\_portvals(start\_date, end\_date, orders\_file, start\_val):

# TODO: Your code here

return portvals

where start\_date and end\_date are the first date and last date to track, respectively (specified as 'yyyy-mm-dd' strings\*), orders\_file is the name of a file from which to read orders, and start\_val is the starting value of the portfolio (initial cash available). Return the result (portvals) as a pandas.Series or a single-columnpandas.DataFrame (column name does not matter), containing the value of the portfolio for each trading day from start\_date to end\_date, inclusive.

* **Note**: If you have already implemented this project assuming start\_date and end\_date are datetimeobjects (as specified in a previous version of the requirements), that is fine - your submission will be graded appropriately if you inform us (instructions on this coming soon!).

The files containing orders are CSV files with the following columns:

* Date (yyyy-mm-dd)
* Symbol (e.g. AAPL, GOOG)
* Order (BUY or SELL)
* Shares (no. of shares to trade)

For example:

Date,Symbol,Order,Shares

2008-12-3,AAPL,BUY,130

2008-12-8,AAPL,SELL,130

2008-12-5,IBM,BUY,50

note- dates are not in order, be sure the program can handle dates not in order.

apply some sort of sort.

Your simulator should calculate the total value of the portfolio for each day using **adjusted closing prices**.

The value for each day is cash plus the current value of equities.

Return The result (dataframe) should contain values like this:

2008-12-3 1000000

2008-12-4 1000010

2008-12-5 1000250

...

We will evaluate your code by calling compute\_portvals() with multiple test cases.

For debugging purposes, you should write your own additional helper function to call compute\_portvals() with your own test cases. We suggest that you report the following factors:

* Plot the price history over the trading period.
* Sharpe ratio (Always assume you have 252 trading days in an year. And risk free rate = 0) of the total portfolio
* Cumulative return of the total portfolio
* Standard deviation of daily returns of the total portfolio
* Average daily return of the total portfolio
* Ending value of the portfolio

## Part 2: Leverage (5%)

Many brokers allow "leverage" which is to say that you can borrow money from them in order to buy (or sell) more assets. As an example, suppose you deposit $100,000 with your broker; You might then buy $100,000 worth of stocks. At that point you would have a cash position of $0 and a sum of long positions (Stock values) of $100,000. This situation is 1.0 leverage. However, many brokers allow up to 2.0 leverage. So, you could borrow $100,000, to buy more stocks. If you did that, you'd have long positions of $200,000 and a cash position of -$100,000 due to the loan. Here's how to calculate leverage:

leverage = (sum(longs) + sum(abs(shorts))) / ((sum(longs) - sum(abs(shorts)) + cash)

Here are a few examples:

* You deposit $100,000, then short $50K worth of stock and buy $50K worth of stock. You would then have $100K of cash, $50K of longs, -$50K of shorts, so your leverage would be 1.0.
* You deposit $100,000 then short $200K worth of stock. You have $300K of cash and -$200K in shorts. So your leverage is 2.0.
* You deposit $100,000 then buy $50K of stock. Your leverage is 0.5.

To receive credit: Your simulator should prohibit trades that would cause portfolio leverage to exceed 2.0.

FAQs:

* Q: What if the portfolio becomes levered after the trades have been entered?
* A: It is OK if the trades are entered and then later, due to stock price changes, leverage exceeds 2.0.
* Q: Should I allow a partial order to be filled so it gets just right up to 2.0
* A: No reject the order entirely.
* Q: What if the portfolio is levered at 2.0 already, should I accept orders that reduce leverage?
* A: Yes.

## Orders files to run your code on

Example orders files are available in the orders subdirectory.

## Short example to check your code

Here is a very very short example that you can use to check your code. Starting conditions:

start\_date = '2011-1-05'

end\_date = '2011-1-20'

start\_val = 1000000

For the orders file orders-short.csv, the orders are:

Date,Symbol,Order,Shares

2011-01-05,AAPL,BUY,1500

2011-01-20,AAPL,SELL,1500

The daily value of the portfolio (spaces added to help things line up):

2011-01-05 1000000

2011-01-06 999595

2011-01-07 1003165

2011-01-10 1012630

2011-01-11 1011415

2011-01-12 1015570

2011-01-13 1017445

2011-01-14 1021630

2011-01-18 1009930

2011-01-19 1007230

2011-01-20 998035

For reference, here are the **adjusted close** values for AAPL on the relevant days:

AAPL

2011-01-05 332.57

2011-01-06 332.30

2011-01-07 334.68

2011-01-10 340.99

2011-01-11 340.18

2011-01-12 342.95

2011-01-13 344.20

2011-01-14 346.99

2011-01-18 339.19

2011-01-19 337.39

2011-01-20 331.26

The full results:

Data Range: 2011-01-05 to 2011-01-20

Sharpe Ratio of Fund: -0.446948390642

Sharpe Ratio of $SPX: 0.882168679776

Cumulative Return of Fund: -0.001965

Cumulative Return of $SPX: 0.00289841448894

Standard Deviation of Fund: 0.00634128215394

Standard Deviation of $SPX: 0.00544933521991

Average Daily Return of Fund: -0.000178539446839

Average Daily Return of $SPX: 0.000302827205547

Final Portfolio Value: 998035.0

## More comprehensive examples

### orders.csv

We provide an example, orders.csv that you can use to test your code, and compare with others. All of these runs assume a starting portfolio of 1000000 ($1M).

Data Range: 2011-01-10 to 2011-12-20

Sharpe Ratio of Fund: 1.21540888742

Sharpe Ratio of $SPX: 0.0183389807443

Cumulative Return of Fund: 0.13386

Cumulative Return of $SPX: -0.0224059854302

Standard Deviation of Fund: 0.00720514136323

Standard Deviation of $SPX: 0.0149716091522

Average Daily Return of Fund: 0.000551651296638

Average Daily Return of $SPX: 1.7295909534e-05

Final Portfolio Value: 1133860.0

### orders2.csv

The other sample file is orders2.csv that you can use to test your code, and compare with others.

Data Range: 2011-01-14 to 2011-12-14

Sharpe Ratio of Fund: 0.788982285751

Sharpe Ratio of $SPX: -0.177203019906

Cumulative Return of Fund: 0.0787526

Cumulative Return of $SPX: -0.0629581516192

Standard Deviation of Fund: 0.00711102080156

Standard Deviation of $SPX: 0.0150564855724

Average Daily Return of Fund: 0.000353426354584

Average Daily Return of $SPX: -0.000168071648902

Final Portfolio Value: 1078752.6

## Implementation suggestions & assumptions

In terms of execution prices, you should assume you get the **adjusted close** price for the day of the trade.

Here is a video outlining an approach to solving this problem [[youtube video](https://www.youtube.com/watch?v=E1GTOSUoDpE)].

25:52 mins

1. read CSV file:

read start and end dates

boils down to reading date range and symbols

2. read adjusted closing prices

## What to turn in

Be sure to follow these instructions diligently!

Via T-Square, submit as attachment (no zip files; refer to schedule for deadline):

* Your code as marketsim.py (only the function compute\_portvals() will be tested)

Unlimited resubmissions are allowed up to the deadline for the project.

## Rubric

* Basic simulator: 10 test cases: We will test your code against 10 cases (9.5% per case). Each case will be deemed "correct" if:
  + For each day, abs(reference portval - your portval) < $0.01
* Leverage: 2 test cases (2.5% per case). Each case will be deemed "correct" if:
  + For each day, abs(reference portval - your portval) < $0.01

https://piazza.com/class/idadrtx18nie1?cid=610

# Answers to questions about MC2-Project-1

Hi Folks,

Sorry this is coming after the deadline.  I've been sick and not able to be online. I thought still I should share this because it may put some questions to rest.

Q: Is it worth the effort to implement the leverage part?  It is only worth 5%. So it seems that if we attempt the leverage portion of the project, we could mess up the regular test cases for the non leverage portion, and end up with a very low grade if things go bad. How should we go about this? Can we have some more information on how this will be handled?

A: I intended that getting 100% on this assignment should be a bit of a challenge (not trivial).  Note though that you can still get an A on the project without completing the leverage part.

Q: Two orders with the same date are received. Accepting both breaks the leverage threshold, rejecting one and accepting the other does not. Now, depending on which order is accepted, the portfolio will have different values, and we have no way of delineating between the two.

A: For the purpose of this assignment, assume that all orders for a day occur simultaneously.  Reject or accept all of them at once.

Q: I have a working solution but it doesn't work if the dates are not in chronological order (which is the case with both orders and orders2, one of the date combinations is reversed chronologically).

A: Sort.

Q: What is required as a submission? More specifically, how would you want us to submit our code so its easy for the auto-grader to grade the general part separately? What should the leverage part return?

A: Just submit your code that solves the complete problem (including leverage). For part 1 we will test your code with examples that do not exceed 2.0 leverage.

Q: how do we handle scenarios where there are two or more trades in a day?  Will this scenario come up?

A: Treat all trades on one day as occurring simultaneously.  Accept them all or reject them all. No, this scenario will not come up with regard to the leverage component.

Q: For part 1, can we assume that the test cases will not exceed a leverage of 2.0?

A: Yes.