40.317 Homework Assignment

This homework assignment is due at 11:59 p.m. on Monday 28th June. You may complete it in groups of 3 – 5 members per group (4 is ideal size). Remember to cc yourself and all your groupmates. You will be severely penalized for late work. There are three parts to this homework assignment

Part 1

You are given two Python scripts, a client and a server, which communicate with each other via ZeroMQ to form a simple distributed application. The client code is complete; the server code is not complete. Finish the server code to form a complete, working application.

To receive credit for this part of the assignment, you must submit:

- Your completed server script.
- One screen shot showing your server script generates zero warnings when run through the `pycodestyle` style-checking utility.
- One or more screen shots showing your client/server work as expected. Your screen shots must demonstrate that your server rejects invalid inputs. Examples of invalid inputs include:
    - An unsupported command
    - Too many inputs
    - Too few inputs
    - Inputs of the wrong type, e.g. a string instead of a number
    - A negative number when only a positive number is OK
    - A purchase attempt without enough cash on hand
    - A sale attempt without enough shares on hand

Part 2

Using the "holdings manager" you implemented in Part 1 above:

1. Modify your holdings manager to compute two quantities every 10 seconds: the Volume-Weighted Average Price ("VWAP") of all *purchases* so far, and the VWAP of all *sales* so far.
   You can find the definition of VWAP online – for instance, here.
    a. Add a global instance of collections.deque which stores the history of all purchases (the quantity and price of each). Modify your buy command logic to append the current buy information into this deque.
    b. Add another global instance of collections.deque which stores the history of all sales. Modify your sell command logic to append the current sell information into this deque.

c. Construct a "daemon thread" and launch it at startup. This thread
    i. computes the buy VWAP (if any) and sell VWAP (if any) from the contents of these two deques;
    ii. prints out these two VWAPs to the console;
    iii. sleeps for 10 seconds;
    iv. repeats step i.

2. Pass your computed VWAPs from the daemon thread back to the main thread, safely, instead of simply printing them out.
    a. Add a third global collections.deque which contains the computed (buy VWAP, sell VWAP) pairs.
    b. Modify the daemon thread to append the buy and sell VWAPs to this deque, instead of printing them out.

3. Extend your asset manager by adding a command which returns the *latest* buy and sell VWAPs (if any).
    a. Implement a new command "get_latest_vwaps". When run, this command retrieves the most recent buy and sell VWAPs from the third deque, and returns them in a string.
        - You can remove older entries from the VWAP deque, but be careful not to remove them all.
    b. On success, this new command should return the string "[OK] <latest buy VWAP> <latest sell VWAP>"
    c. If either or both of these VWAPs are missing, it should return "N/A" in place of a number.

To receive credit for this part of the assignment, you must submit:

- Your completed server script.
- One screen shot showing your server script generates zero warnings when run through the `pycodestyle` style-checking utility.
- One or more screen shots showing your client/server work as expected. Your screen shots must demonstrate that your server can be shut down cleanly, i.e. you do not need to explicitly shut down the daemon thread in order to shut down the server.

Part 3

Use the PySimpleGUI package to create a working graphical client for the holdings manager server you wrote for Part 2 above. Its appearance should closely resemble the screen shot below. Its behaviour should include the following:

- Clicking on "Deposit" button should increment the cash holdings by the amount input under "Deposit Cash"
- Clicking on "Buy" button should increment the shares holdings by the amount input under "Buy Shares" and decrement cash holdings by "Quantity × Price per Share"

- Clicking on “Sell” button should decrement the shares holdings by the amount input under “Sell Shares” and increment cash holdings by “Quantity × Price per Share”
- Clicking on “Close Server & Quit” button should, as the name suggests, close the server and quit.
- At least 3 types of error messages:
    - When you input a string instead of a number, an error message pops up.
    - When you try to buy shares without sufficient cash holdings, an error message pops up.
    - When you try to sell shares without sufficient shares holdings, an error message pops up.
    - **Bonus marks** will be given for additional error messages that you can think of.
- Use buttons to send commands to the server
    - Note that although the holdings manager is exactly the same as that you wrote in Part 2 above, you need to make some minor changes, i.e., you are expected to map the buttons to the various commands for the holdings manager. You are not expected to make any other changes to the holdings manager.

Perform the following setup steps before you start working:

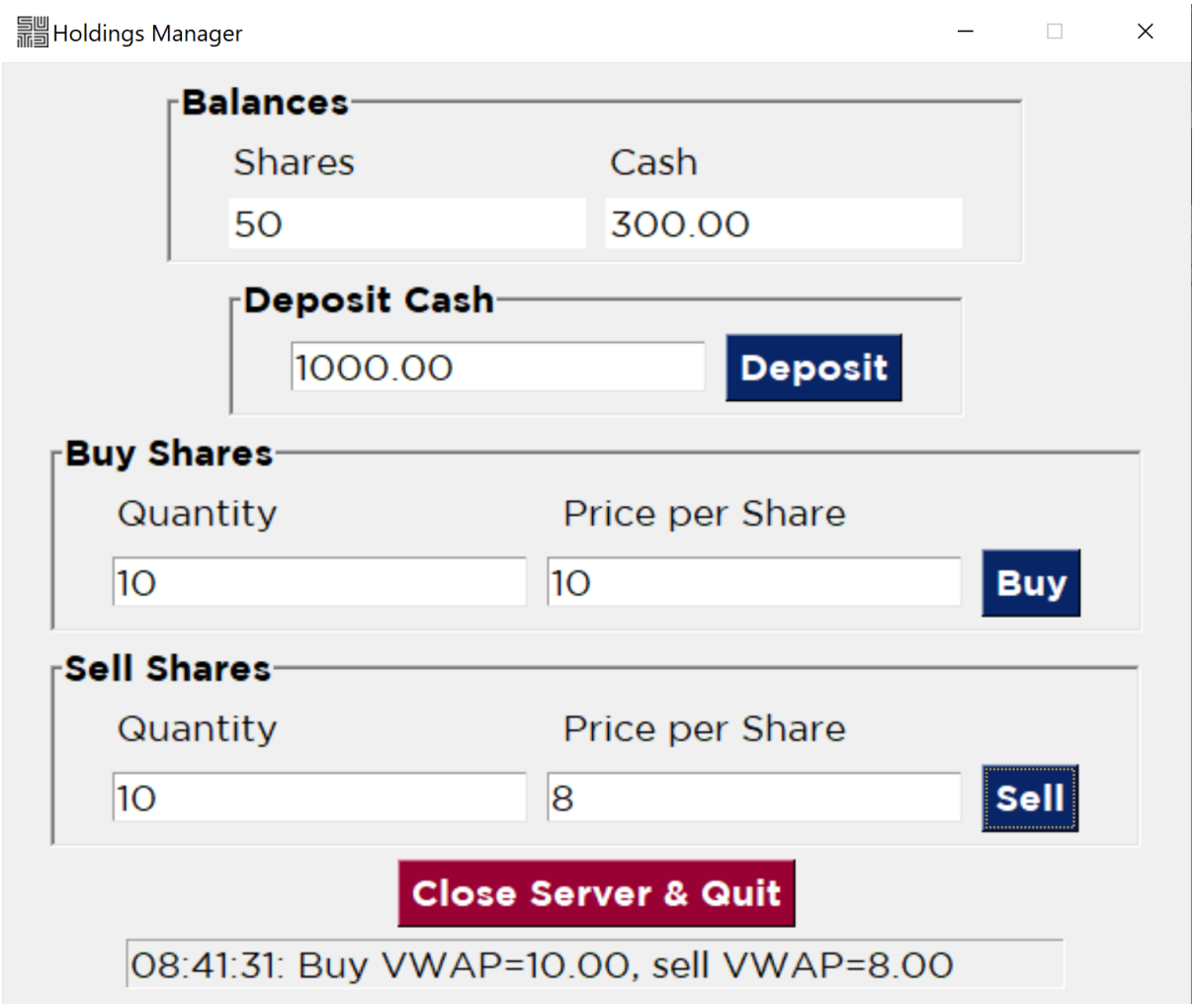
1) Install PySimpleGUI by running
   `pip install pysimplegui`
   in a command prompt.
2) Add the two provided .ttf files (Gotham Book.ttf and Gotham Bold.ttf) to your laptop’s set of available fonts. The specific steps are of course different for Windows versus MacOS. You can find the instructions for your operating system online.

To receive credit for this assignment, you must submit:

- Your completed GUI client script.
- Your holdings manager server from Homework 2, for testing your GUI client.
- A screen recording of your GUI client in action, showing it works as expected.
    - This recording should be at most 1 minute long.
    - Instructions for making a screen recording on Windows 10 and MacOS are here and here. If making this recording turns out to be difficult, a series of screen shots is also acceptable.
- (There is no requirement that your code have zero warnings from pycodestyle.)

Additional suggestions:

- There are many clearly written demo programs which can help you.
- Do not hesitate to read through the PySimpleGUI source code itself.


Holdings Manager
Balances
Shares
Cash
50
300.00
Deposit Cash
1000.00
Deposit
Buy Shares
Quantity
Price per Share
10
10
Buy
Sell Shares
Quantity
Price per Share
10
8
Sell
Close Server & Quit
08:41:31: Buy VWAP=10.00, sell VWAP=8.00

Appendix: Python scripts via a Windows prompt

You can freely ignore this appendix if you use MacOS, or if you intend to always run your Python scripts via the Anaconda Prompt.

1. During the installation of Anaconda 3, tick the box to add the path to python.exe to your PATH environment variable.
2. The remaining instructions will allow you to run a Python script from a Windows command prompt by simply typing "<script name> <argument(s)>" as well as "python <script name> <argument(s)>". They summarise the contents of this Stack Overflow article: https://stackoverflow.com/questions/29540541/executable-python-script-not-take-sys-argv-in-windows
    a. The Registry key
    `HKCR\Applications\python.exe\shell\open\command`
    must contain the string
    `"<Path to python.exe>\python.exe" "%1" %*`
    b. The Registry key
    `HKCR\py_auto_file\shell\open\command`
    must contain the string
    `"<Path to python.exe>\python.exe" "%1" %*`
    c. In either of these Registry keys, the trailing `%*` *might* be missing, in which case you should (carefully) add it.