DSP 2019 – Assignment 01: A simple stock manager

An investor buys and sells shares of stock in various companies listed in the stock market. What if the investor sells 120 shares of a company for $90 each at a time when the investor owns 150 shares, of which 50 shares were bought in January at $60 each, 50 more in February at $70 each, and 50 more in March at $80 each? Which of the shares have to be sold?

Tax laws require the investor to compute the profit (or loss) on a LIFO basis, i.e, the most recently purchased shares are sold first. So the profit on the sale is $500 on the 50 shares purchased in March, $1000 on the 50 shares purchased in February, and $600 (20 shares at $30 each profit) on shares purchased in January.

Therefore, an appropriate data structure for the investor is one stack of share transactions for each of the companies that the investor owns stock in.

Your assignment is to implement a software “Stock Manager” for the investor. In doing so, you have to complete the class “StockManager” provided in the PyCharm code skeleton under the package “assignment01”.

The following requirements should be satisfied by your implementation:

- The “StockManager” can manage shares of the following 3 companies: “Google”, “Apple”, and “BMW”

Your “StockManager” allows the investor do to the following:

1. buy_shares(company, number, buy_price) : buy shares of a company at a certain price.
   An error message is printed on screen if we try to buy shares of a company that the StockManager does not manage.
2. sell_shares(company, number, sell_price) : sell shares of a company at a certain price.
   An error message is printed on screen if we try to sell shares that we do not own (i.e., of a company not managed or shares of a managed company not in the portfolio).
3. sell_multiple(company_list, number_list, sell_price_list) : sell shares of different companies in one go, e.g., sell_multiple([‘Google’, ‘Apple’], [20, 10], [30, 67]) sells 20 shares of Google at $30 and 10 shares of Apple at $67.
   An error message is printed on screen if we try to sell shares that we do not own (i.e., of a company not managed or shares of a managed company not in the portfolio).
4. buy_multiple(company_list, number_list, sell_price_list) : buy shares of different companies in one go (parameters are set similarly to sell_multiple())
   An error message is printed on screen if we try to buy shares of a company that the StockManager does not manage.
5. All “sell” methods should return the profit/loss made by the investor in the current transaction(s)

6. The software should keep track of the cumulative profit (or loss) made by the investor since the initialisation of the “StockManager”. This information can be queried at any time calling the method get_profit()

EXAMPLE

```
SM = StockManager()                          # Initialises new Stock Manager
SM.buy_shares(‘Google’, 10, 50)
SM.buy_shares(‘Apple’, 20, 30)
SM.buy_shares(‘Google’, 20, 40)
SM.buy_shares(‘Apple’, 20, 35)
print(SM.sell_shares(‘Apple’, 5, 30))        # Output: -25
SM.get_profit()                              # Output: Loss of $25 from stock
manager initialisation
print(SM.sell_shares(‘Google’, 25, 60)) # Output: 450        (= 20*20 + 5*10)
SM.get_profit()                              # Output: Profit of $452 from stock
manager initialisation
```

Instructions:

- You have to conduct this project **in groups of 2. Please notify your group to the TAs**
- You have to **submit the code** that you develop **on blackboard** by the **deadline of Thursday October 21 at 10pm**.
  Please zip the folder “assignment01” on your computer and upload the zip file. The submission area will be available in due time. The folder **assignment01 must contain all the files that are needed to run stock_manager.py** (that is, stock_manager.py must not import anything that is outside the folder “assignment01”)
- Your submission should also include a text file containing **the link to a video demo uploaded on some public Web site (**e.g., Youtube). In the video, you should demonstrate the fulfilment of all the requirements (points will be deducted for each unfulfilled requirement). Keep this video within 3 minutes.

- You have to **give a demo** of your code to the professor/TAs. The exact date of the demo will be communicated later on.

  <u>You are not allowed to change your code between the submission deadline and the demo</u> (we will check!).
  “Giving a demo” means to show the functionality of your software. So, you have to develop appropriate code in the main() of your application to showcase the implemented functionality. Failing to demonstrates the implementation of (some of the) functionality will lead to point deductions. **Both members of the group must attend the demo**.