A View of the Stock Market the Past Year

Overview of Project

Purpose

Grateful, Steve, that you have appreciated our work so far and would like us to expand the dataset to include the entire stock market over the last few years. We will check to see if the information is the same as previously delivered then we will evaluate if the VBA script runs in a timely fashion.

Background

To recap, this project is a result of your parent’s desire to invest in alternative energy sources which lead to their investment in DAQO New Energy Corp. You promised them an assessment of the company’s performance as well as you have concerns about diversity. Your request of a new program will help provide you with data to assist you in possible choices so you can diversify.

Results

Summary: In a summary statement, address the following questions.

* 1. What are the advantages or disadvantages of refactoring code?
  2. How do these pros and cons apply to refactoring the original VBA script?

**Analysis and Challenges**

Here's a quick look at the Kickstarting Analysis and Challenges of this Project, including the following tasks:

* Prepare our dataser VBA\_Challenge.vbs file for the project.
* Create our resources folder in **GitHub** to hold the run-time pop-up messages that we’ll screenshot after running refactored analyses for 2017 and 2018.
* Create and convert our XLSM file from \*.vbs dataset that you used in this module as VBA\_Challenge.xlsm.
* Add the VBA\_Challenge.vbs script to the Microsoft Visual Basic editor.
* Use the steps **Refactor VBA code and measure performance** to add code where indicated by the numbered comments in the starter code file.

Use your knowledge of VBA and the starter code provided in this Project to refactor the VBA Script dataset so we loop through the data one time and collect all of the information.

**Our Challenge Data Background**

Steve loves the workbook you prepared for him. At the click of a button, he can analyze an entire dataset. Now, to do a little more research for his parents, he wants to expand the dataset to include the entire stock market over the last few years. Although your code works well for a dozen stocks, it might not work as well for thousands of stocks. And if it does, it may take a long time to execute.

In this challenge, you’ll edit, or refactor, the Module 2 solution code to loop through all the data one time in order to collect the same information that you did in this module. Then, you’ll determine whether refactoring your code successfully made the VBA script run faster. Finally, you’ll present a written analysis that explains your findings.

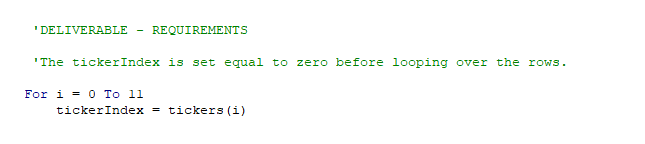
Refactoring is a key part of the coding process. When refactoring code, you aren’t adding new functionality; you just want to make the code more efficient—by taking fewer steps, using less memory, or improving the logic of the code to make it easier for future users to read. Refactoring is common on the job because first attempts at code won’t always be the best way to accomplish a task. Sometimes, refactoring someone else’s code will be your entry point to working with the existing code at a job.

**RESULTS: Refactor VBA Code and Measure Performance**

**Deliverable Requirements, Code Examples, Compare Stock Performance and Timestamp procedure below:**

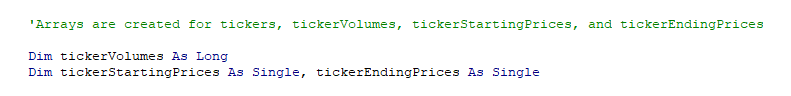
**1. The tickerIndex is set equal to zero before looping over the rows.**

Created a tickerIndex variable and set it equal to zero before iterating over all the rows. Will use this tickerIndex to access the correct index across the four different arrays on VBA Code: the tickers array and the three output arrays created on next requierement.

[](https://github.com/emmanuelmartinezs/stock-analysis/blob/master/data_files/resources/The%20tickerIndex.PNG?raw=true)

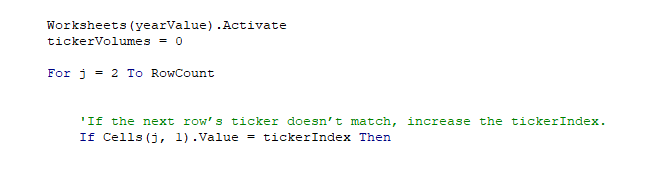
**2. Arrays are created for tickers, tickerVolumes, tickerStartingPrices, and tickerEndingPrices.**

Created three output arrays: tickerVolumes, tickerStartingPrices, and tickerEndingPrices. In our VBA code, the tickerVolumes array should be a **Long** data type. But in our VBA code the tickerStartingPrices and tickerEndingPrices arrays should be a **Single** data type.

[](https://github.com/emmanuelmartinezs/stock-analysis/blob/master/data_files/resources/Arrays%20are%20created.PNG?raw=true)

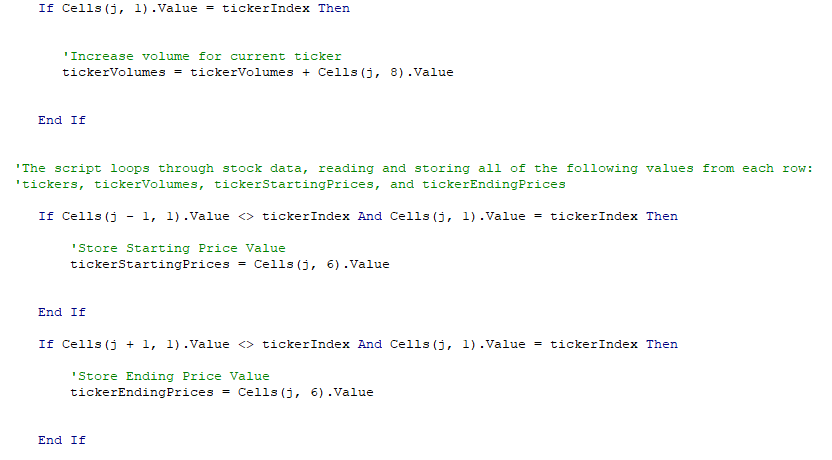
**3. The tickerIndex is used to access the stock ticker index for the tickers, tickerVolumes, tickerStartingPrices, and tickerEndingPrices arrays.**

Created a for loop to initialize the tickerVolumes to **zero**. And if the next row’s ticker doesn’t match, increase the tickerIndex.

[](https://github.com/emmanuelmartinezs/stock-analysis/blob/master/data_files/resources/The%20tickerIndex%20is%20used%20to%20access%20the%20stock%20ticker.PNG?raw=true)

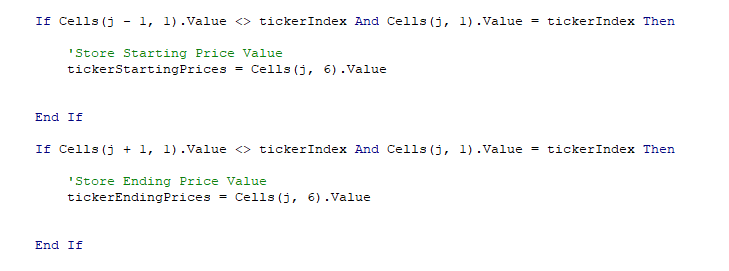
**4. The script loops through stock data, reading and storing all of the following values from each row: tickers, tickerVolumes, tickerStartingPrices, and tickerEndingPrices.**

Created a **loop** that will loop over all the rows in the spreadsheet. Inside the **loop**, we created a script that increases the current tickerVolumes **(stock ticker volume)** variable and adds the ticker volume for the current stock ticker.

[](https://github.com/emmanuelmartinezs/stock-analysis/blob/master/data_files/resources/The%20script%20loops%20through%20stock%20data,%20reading%20and%20storing.PNG?raw=true)

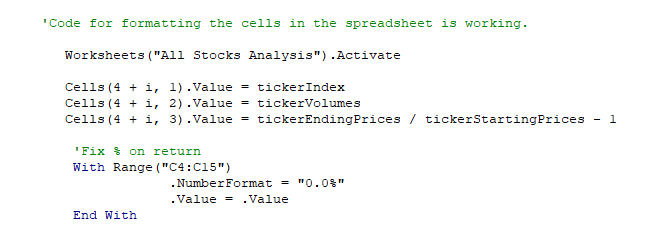
**Stored values from** tickerStartingPrices **and** tickerEndingPrices

Created an **if-then** statement to check if the current row is the first row with the selected tickerIndex. If it is, then assign the current closing price to the tickerStartingPrices and tickerEndingPrices variable.

[](https://github.com/emmanuelmartinezs/stock-analysis/blob/master/data_files/resources/Start%20and%20EndPrices%20Code.PNG?raw=true)

**5. Code for formatting the cells in the spreadsheet is working.**

We make positive returns green and negative returns red, to be a lot easier to determine which stocks did well and which ones didn't. Added some formatting based on the values of the returns.

[](https://github.com/emmanuelmartinezs/stock-analysis/blob/master/data_files/resources/Code%20for%20formatting%20the%20cells%20in%20the%20spreadsheet%20is%20working.PNG?raw=true)

**6. There are comments to explain the purpose of the code.**

Adding **Comments** is requiered, as a **Best Practices for Writing Super Readable Code** such,

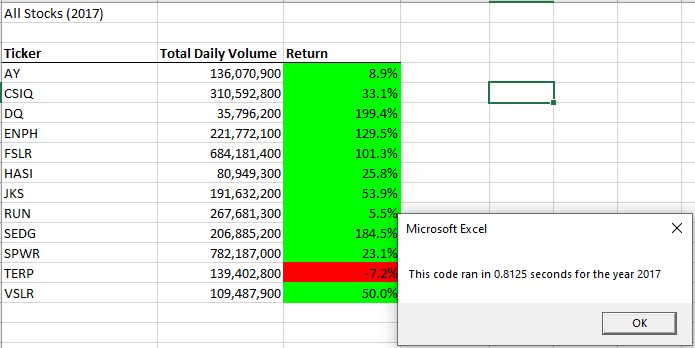
* Commenting & Documentation,
* Consistent Indentation,
* Avoid Obvious Comments.
* Code Grouping,
* Consistent Naming Scheme,
* DRY (Don't Repeat Yourself) Principle,
* Avoid Deep Nesting,
* Limit Line Length, etc...

**7. The outputs for the 2017 and 2018 stock analyses in the VBA\_Challenge.xlsm workbook match the outputs from the AllStockAnalysis in the module**

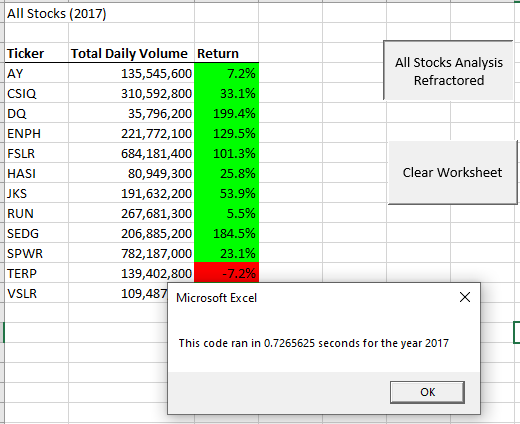
Finally, we run the stock analysis, to confirm that our stock analysis outputs for 2017 and 2018 are the same as dataset example provided (as shown in the images below, named **Dataset Examples Provided**). In adition, in our resources folder and below you can see the final Stock Analysis Results named, **Final VBA Analysis 2017 and 2018** save the pop-up messages showing elapsed run time for the refactored code as VBA\_Challenge\_2017.png and VBA\_Challenge\_2018.png. Then, save the changes to your workbook..

***Dataset Examples Provided***

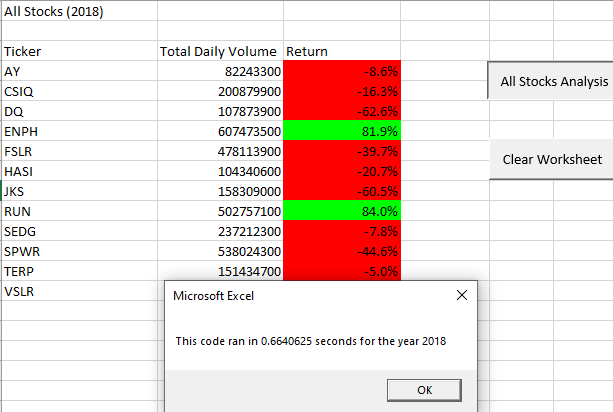
***Before***



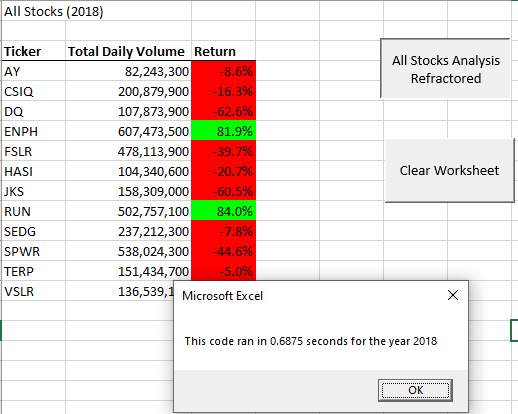
**Refactored**



Before



**Refactored**



Below our Final VBA Analysis PNGs,

***Final VBA Analysis 2017***

***Final VBA Analysis 2018***

**8. The pop-up messages showing the elapsed run time for the script are saved as VBA\_Challenge\_2017.png and VBA\_Challenge\_2018.png**

Running our fully 2017 and 2018 data stock analysis gave us an elapsed run time for each year, below our results.

***Time on VBA\_Challenge\_2017.PNG***

***Time on VBA\_Challenge\_2018.PNG***

**SUMMARY: Our Statement:**

**Deliverable with detail analysis:**

**1. What are the advantages or disadvantages of refactoring code?**

You need to perform code refactoring in small steps. Make tiny changes in your program, each of the small changes makes your code slightly better and leaves the application in a working state.

**Disadvantages:**

* A long procedure may contain the same line of code in several locations, you can change the logic to eliminate the duplicate lines.
* A logical structure may be duplicated in two or more procedures (possibly via copy & paste coding). When detected, this logic is best moved to a new function and called from the other functions.
* A complex unstructured code is usually best to split in several functions.
* Refactoring process can affect the testing outcomes.

**Advantages:**

* Logical errors easily appear in well structure code that contains nested conditionals and loops.
* In our case, using Excel flow displays program logic in a more comprehensible manner, not tied to the order that the underlying code is written.
* VBA interpretation (Excel) of code can reveal patterns that are not easy to see in the source.

**2. How do these pros and cons apply to refactoring the original VBA script?**

Improving or updating the code without changing the software’s functionality or external behavior of the application is known as code refactoring. Now, let's think about something, **What happens after a couple of days or months yo need to troubleshoot your code? Is it complicated? Is it hard to understand?** If yes then definitely you didn’t pay attention to improve your code or to restructure your code.

***We need to consider the code refactoring process as cleaning up the orderly house.*** *Unnecessary clutter in a home can create a chaotic and stressful environment.* - The same goes for written code.

A clean and well-organized code is always easy to change, easy to understand, and easy to maintain. You can avoid facing difficulty later if you pay attention to the code refactoring process earlier.