**Stock-Analysis**

VBA in Excel Project Challenge: Module 2

**Background**

Purpose Challenge includes 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.

**History of Project**

Steve just graduated with his finance degree. His parents want to invest in DAQO New Energy with Ticker DQ. He wants to analyze a handful of green energy stocks as well as DAQO Stocks with assistance in using VBA to automate tasks. Steve loves workbook and is wanting to expand the dataset to include the entire stock market over the last few years.

**Assignment**

Analysis and Challenges Assignment consists of one technical deliverable and a written report to deliver your results. Submit following in GitHub by creating a repository. Submit the following:

Deliverable 1: Refactor VBA code and measure performance. Assignment to include an updated workbook and a folder with PNGs of the pop-ups with script run time.

Deliverable 2: A written analysis of your results (README.md) Download the challenge\_starter\_code.vbs file and rename it VBA\_Challenge.vbs. Create a folder called “Resources” to hold the run-time pop-up messages that you’ll screenshot after running refactored analyses for 2017 and 2018. Rename the green\_stocks.xlsm file that you used in this module as VBA\_Challenge.xlsm. Add the VBA\_Challenge.vbs script to the Microsoft Visual Basic editor.

Results

1a) Create a ticker Index by setting variable equal to 0

tickerIndex = 0

1b) Create three output arrays: tickerVolumes with long data type, tickerStartingPrices, and tickerEndingPrices with single data type.

Dim tickerVolumes(12) As Long

Dim tickerStartingPrices(12) As Single

Dim tickerEndingPrices(12) As Single

2a) Create a ‘for’ loop to initialize the tickerVolumes to zero. Loop tells computer to repeat lines of code over and over again.

For i = 0 To 11

tickerVolumes(i) = 0

tickerStartingPrices(i) = 0

tickerEndingPrices(i) = 0

Next i

2b) Create a ‘for’ loop that will loop over all the rows in the spreadsheet (count starts with row 2 on (year.Value) worksheet)

For i = 2 To RowCount

3a) Increase volume for current ticker (noted without a conditional ‘If’ because just setting the volume as an increase only). Script written to increase the current tickerVolumes (stock ticker volume) variable and adds the ticker volume for the current stock ticker. ‘tickerIndex’ variable is used as the index.

tickerVolumes(tickerIndex) = tickerVolumes(tickerIndex) + Cells(i, 8).Value

3b) Check if the current row is the first row with the selected tickerIndex. If-Then statement written to check if the current row is the first row with the selected tickerIndex. If it is, then it should assign the current cstarting price to the tickerStartingPrices variable.

'If Then (Creating a pseudocode pattern to break down the algorithm to validate row value.)

Script: If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i -1, 1).Value <> tickers(tickerIndex) Then tickerStartingPrices(tickerIndex) = Cells(i, 6).Value

End If

3c) check if the current row is the last row with the selected tickerIndex. If-Then statement written to check if the current row is the last row with the selected tickerIndex. If it is, then it should assign the current closing price to the tickerEndingPrices variable.

Script: If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i + 1, 1).Value <> tickers(tickerIndex) Then tickerEndingPrices(tickerIndex) = Cells(i, 6).Value

End If

3d) Increase the tickerIndex. Script written that increases the tickerIndex if the next row’s ticker doesn’t match the previous row’s ticker.

Script: If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i + 1, 1).Value <> tickers(tickerIndex) Then tickerIndex = tickerIndex + 1

End If

Next i

Challenges:

Received this when initially ran code…

Graphical user interface, text, application, chat or text message

Description automatically generated

Guidance below retrieved from: <https://stackoverflow.com/questions/22580516/compile-error-end-if-without-block-if>

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

I changed for format in 3b) thru 3d) to reflect

If Cells(i, 1).Value = tickers(tickerIndex) And Cells(i - 1, 1).Value <> tickers(tickerIndex) Then

tickerStartingPrices(tickerIndex) = Cells(i, 6).Value

End If

**Results**

All Stocks (2017) Results

Graphical user interface, application

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

All Stocks (2018) Results:

Graphical user interface, application

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

Analysis

Refactor code ran faster for Stocks data in 2018 (0.13 seconds) versus Stocks data in 2017 (0.14 seconds). These times noted also were faster than the initial code based on refactoring method being a success. Stocks in 2018 had more of a dip in return percentages versus stocks in 2017 (shown by green with positive value and red with negative values). Return of percentages were better in 2017.

**Summary**

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

Advantages to code refactoring includes:

* Maintainability  
  The main goal of code refactoring is to make it easy to enhance and maintain in the future. It should not violate Open Close Principle.
* Removing Bad Smell  
  Bad code smell motivates the refactoring of code. It prevents many future defects. Code Size is reduced. Confused coding is properly restructured.

Information retrieved from: <https://www.c-sharpcorner.com/article/pros-and-cons-of-code-refactoring/>

Disadvantages (dangers) to code refactoring includes:

* It is expensive and risky in the view of management.
* It may introduce bugs.
* Delivery schedule is very tight.
* Management doesn't care about maintainability and extension of code base.

Information retrieved from: <https://www.c-sharpcorner.com/article/pros-and-cons-of-code-refactoring/>

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

Pros apply to original VBA script because it helps in the following areas:

* detecting code smells known as bad patterns like tight coupling, duplicate code, long methods, large classes, etc. are detected in the code;  the code should be refactored in this case.
* And fixes bugs(debugs) codes that are written badly in some cases, and so many bugs are raised. In this case, fixing of bugs take too much effort. So, the root cause of bugs can be code smell. So, before fixing bugs code should be refactored.

Information retrieved from: <https://www.c-sharpcorner.com/article/pros-and-cons-of-code-refactoring/>

Cons apply to original VBA script includes, just to name a few, includes:

* + Being a a huge short-term time sink, because you have to write tests and change existing code. However, long-term, you just gained a lot of potential, because you enabled further refactorings and you cleaned up some code smells.
  + Potentially being unsafe because if you break something, you break it and you might or might not find it with manual tests.

Information retrieved from: <https://www.quora.com/What-are-the-pros-and-cons-of-refactoring>