# Runtime Difference in VBA Code

## Overview:

The purpose of this analysis is to compare the difference in start to finish time between the two sets of VBA codes. The first set of VBA code is provided in the lesson on this module (the original cod), and the second set of VBA code is a modified, or refactored, version of the lesson’s VBA code.

### Background:

#### VBA Code Provided in the Lesson

The original code contains a set of nested “For” loops, which has 2 “For” loops, one loop (loop “i”) to sort through the 12 different ticker symbols and another loop (loop “j”) to go through all of the rows that contain ticker symbol data (The 2 “For” loops can be seen (here) [LINK]). The 12 different ticker symbols are stored in an output array called “tickers”, seen (here) [LINK].

The VBA code runs through each iteration of the first “For” loop “i”, which is set from 0 to 11; this matches the number of ticker symbols. Excel is told to look for the specific ticker symbol in order, as defined by the output array “tickers”.

The second ‘For” loop “j” will then go through all of the rows in the ticker symbol column and check the previous, current and next cell against the current ticker symbol set by “i” to determine the first and last time a ticker symbol is present. The first time the current ticker symbol is shown, the price of that appearance will be stored as the starting price for that particular ticker; while the last time the current ticker symbol is shown, its price will be stored as the ending price for that particular ticker. While “j” runs through all the rows, the volume of the matching ticker symbol will be stored and added to the total daily volume of that particular ticker.

Once the “j” loop has completed the analysis for the current ticker symbol, the next iteration of the “i” loop will execute to set a new current ticker symbol and the “j” loop will begin again find all the data that matches the new ticker symbol. This process will execute until the last iteration of the “i” loop is completed.

#### Refactored VBA Code

The refactored VBA code is modified to using only 1 “For” loop (“i”) instead of 2. There is now only 1 “For” loop created to go through all of the rows in the ticker symbol column. A new variable, “tickerIndex”, is created and initialized to have the value of 0. “tickerIndex” is then used as an index value for each output array (ticker volume, starting and ending prices) when the “For” loop runs through all of the rows in the ticker symbol column. The 1 “For” loop can be seen (here) [LINK].

The difference in the refactored code is that since there is one “For” loop, the loop will go through the ticker symbol column looking for the current ticker symbol as defined by “tickerIndex”. The volume data in that row will be added to the current ticker, while also checking the previous and next rows whether this is the first / last time the current ticker is present to store values for staring / ending prices. When this process is done for the current row, the “For” loop will go to the next index position in “tickerIndex” and set a new current ticker symbol and the same analysis is performed again for the next iteration of loop “i” until it reaches the end of the ticker symbol column.

##Results

### Results for year 2017

The original code completed the analysis in 0.71875 seconds ( (results)[LINK] ), while the refactored code finished in 0.09375 seconds ( (results)[LINK] ).

### Results for year 2018

The original code completed the analysis in 0.7304688 seconds ( (results)[LINK] ), while the refactored code finished in 0.1054688 seconds ( (results)[LINK] ).

### Verdict

Comparing the results of the original code versus the refactored code, it can be spotted the refactored code is about 7 times faster than the original code. The original code runs slower in comparison because the second “For” loop runs through all the rows 12 times (once for each iteration of the “i” loop), whereas the single “For” loop in the refactored code only runs through all the rows 1 time. There are 3012 rows in the data, so the original code processes 36144 rows (3012 rows \* 12 ticker symbols) to finish the analysis, while the refactored code only has to go through 3012 rows and tallies up each row to complete the analysis.

## Summary

### Advantages and Disadvantages of Refactoring Code

#### Advantages

One advantage of refactoring code is to develop new code that can be used for new analyses. The new code can be modified to various degrees depending on how different the analysis needs to be. Minor differences can be made to the original code to obtain new information quickly, while keeping the majority of the code the same to save a great deal of time.

Another advantage of refactoring code is to come up with newer versions of the code to accomplish the same purpose, as exhibited by this challenge. By applying changes to the original code, developers can test the efficiency of the new code and decide whether the original code needs to be updated based on the results.

#### Disadvantages

A disadvantage of refactoring code is the time commitment of writing new code. It should be thought out before refactoring the original code whether the new code will have significant improvement on efficiency vs. the time spent on writing it. In the real world, spending money to have developers to try to improve code that already took significant resources to accomplish should have obvious improvements. In other words, the new outcome should be economically feasible to justify the costs of accomplishing it.

### Advantages and Disadvantages of Each Version of the VBA Code

The clear advantage of the original code is its efficiency over the refactored code. In this project, the runtime differences might only be tenths of a second, but if the datasets were 10 or 100 times bigger, there will be a proportionally greater difference in time savings by using the original code.

The downside of the original code is that it uses a nested “For” loop. While it proves that it is more efficient than the single “For” loop in the refactored code, nested “For” loops are convoluted and more difficult to follow along. If a different project has nested “For” loops that contains many more “For” loops than just the two in the original code, it is easy to lose visibility of the code and more difficult to debug.

On the other hand, the obvious disadvantage of the refactored code is that it completes the analysis slower but it is more concise and easier to follow. When it comes to debugging the code, the single “For” loop is easier to understand and less likely for errors to occur.