# Stock Analysis Using VBA

## Overview of Project

Steve asked me for assistance in developing and modifying an Excel spreadsheet that could analyze an entire dataset of stock information.

### Purpose

Steve requested that I refactor, i.e. edit, the VBA code from the Excel spreadsheet that I developed earlier for him. I then changed the code to loop through all the data one time in order to collect the same information that was gathered from the earlier design but used multiple loops. By doing this, I sought to determine whether or not the refactor made the VBA script run faster.

## Analysis

To get started I reviewed the original green\_stocks.xlsm file and the refactored code provided by Steve in the challenge\_starter\_code.vbs file. In particular, I compared the VBA code from each of the files. The refactored code reduced the number of “for” loops in the code. This was intended to reduce the compiling time. To correct for the reduced number of “for” loops, I added a tickerIndex to access the correct index across the four different arrays I used. These arrays were the tickers array, and the three output arrays, tickerVolumes, tickerStartingPrices and the tickerEndingPrices. However, a series of steps were missing in the code that were needed in order to run the stock analysis. That is, pull from the stock data set a specific subset of stocks and place that data on a separate worksheet, “All Stock Analysis.” This report provides the ticker, total daily volume, and annual percentage return of the subset of stocks. See below.

![Stock\_report.png](hyperlink)

## Advantages and Disadvantages of Refactoring Code

The advantages of refactoring are (1) improving software design, (2) helps make software easier to understand, (3) helps find bugs, and (4) helps to program faster. The disadvantages of refactoring are that it takes time and money. This was referenced from a StackOverflow answer. (Masud Shrabon, May 18, 2017, https://stackoverflow.com/questions/43983284/what-are-the-advantages-and-disadvantages-of-refactoring-code-smell-in-software).

## Challenges

I had significant challenges getting the refactored code to compile and do the required data pull. In fact, I spent over three hours modifying the refactored code to get it to compile. During this I went through three different ways to diagnose code issues: using breakpoints, step-by-step execution using the F8 key and the print and immediate window and watch. These helped but not enough to get the code to compile. After reviewing code, I found on GitHub, I realized I had made minor errors in the code, specifically on creating the three output arrays, initializing the output arrays, and looping through the arrays to provide the three outputs. See corrected code and the referenced code below. The corrections changed the function.

## Results

I was able to achieve the objective of refactoring the code to shorten the compiling time while still producing the required report. See below the screenshots of the compiling time of the original green\_stocks.xlsm file for years 2017 and 2018, which were ~ 0.4 seconds each.

![green\_stocks\_2017.png](hyperlink)

![green\_stocks\_2018](hyperlink)

The refactored code compiled a little quicker at ~0.1 seconds each. See below the screenshots of the VBA\_challenge.xlsm file for year 2017 and 2018.

![green\_stocks\_2017.png](hyperlink)

![green\_stocks\_2018](hyperlink)

Given the time involved in getting the refactored code to compile, I did not find it was worth my time of more than three hours to refactor the code as compared to using my original design since it only improved compiling time by ~ 0.3 seconds. I did, however, learn tools to help in diagnosing and fixing code errors.