**Name: Joshua Solomons**

**Candidate Number : 2705**

**Centre Name : Highgate School**

**Centre Number : 12610**

**Project Title : Modelling Financial Data**

**Qualification Code : H446**

**Date : 2021/22**

**Table of Contents**

Contents

[Analysis 3.1 3](#_Toc88555100)

[Problem Identification 3.1.1 3](#_Toc88555101)

[Proposed Solution 3.1.1a) 3](#_Toc88555102)

[Computational Approach 3.1.1b) 3](#_Toc88555103)

[Stakeholders 3.1.2 3](#_Toc88555104)

[Research 3.1.3 3](#_Toc88555105)

[Specify the solution 3.1.4 7](#_Toc88555106)

[Requirements 3.1.4 a) 7](#_Toc88555107)

[Success Criteria 3.1.4 b) 7](#_Toc88555108)

[Design 3.2 9](#_Toc88555109)

[Decomposition of Problem 3.2.1 9](#_Toc88555110)

[Describe the solution 3.2.2 9](#_Toc88555111)

[Explain and Justify the structure of the solution 3.22 a) 9](#_Toc88555112)

[Algorithms 3.2.2 b) 10](#_Toc88555113)

[Class Tables 3.2.2 bi) 10](#_Toc88555114)

# Analysis 3.1

## Problem Identification 3.1.1

### Proposed Solution 3.1.1a)

* I Propose to make a model that allows users to create a stock trading dashboard by graphing the stock selected and creating certain indicators that can improve the user’s experience

### Computational Approach 3.1.1b)

* This project is amenable to a computational approach because:
  + My Project will use a Database which I have learnt about in the Theory side of the course
    - I will also use Object Orientated Methods which is a Programming Paradigm to decompose my project into more manageable smaller code
      * The decomposition will include
  + This project lends itself to abstraction

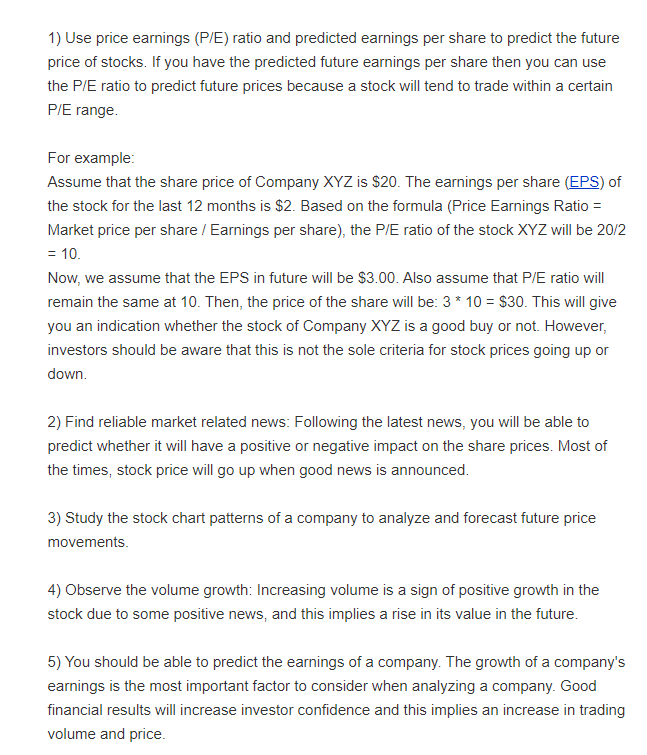
## Stakeholders 3.1.2

* My Dad is currently the Chief Customer Officer at a London stock company, which means he guides Customers through the stock market
  + Creating a program that predicts the future of certain stocks will be hugely beneficial for him and allow him to navigate clients through the stock market with much more certainty
* I am also going to use some of my classmates to give me real-time advice on how to make the GUI considerably easier to use for people with limited knowledge of the stock market

|  |  |  |
| --- | --- | --- |
| Stakeholder | Role | Interaction |
| Dad | * Chief Customer Officer at RJO Brian | * Will give me ideas on what to include on my Stock Dashboard |
| Matthew | * Classmate, limited Compsci knowledge | * Will give me feedback on how to make my dashboard most User Friendly |

## Research 3.1.3

* Future Prices of Stocks can be found using the Price Earnings ratio. However, this takes a lot of effort and skill and can be quite difficult



* There are three conventional approaches for stock price prediction
  + Technical analysis
  + Traditional time series
  + Machine learning
* However, Stocks can be extremely volatile which means it’s very difficult to accurately predict, however although it won’t be easy, I plan to make my stock predictor the one that will not fail

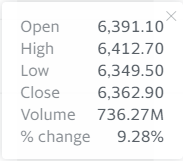
I also plan to conduct a survey of my fellow pupils to see what their most important feature of a stock market predictor would be. However, this is a niche topic which means not a huge amount of people would care about a stock market predictor, which means I may have to limit the survey to my fellow pupils in my Computer Science class.



This shows the graph of Barclays with numerous stats above it

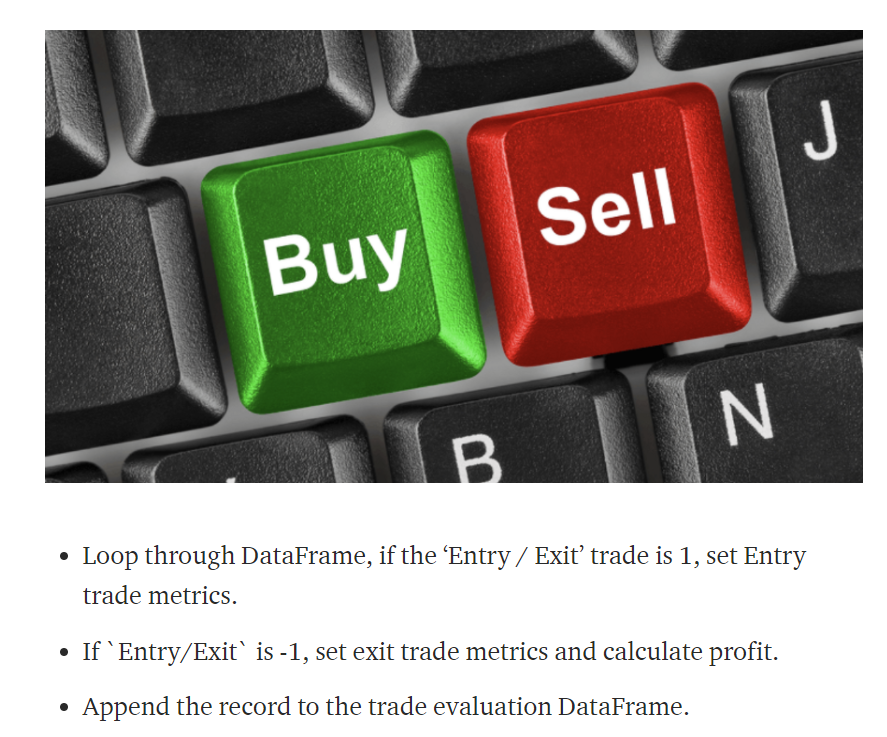
* During my research, I have come across a few interfaces.
  + This one has given me the idea to have a table of statistics either above or below the graph when it comes the GUI
  + This graph also allows the user to change the time series with just a click of the mouse, which has given me the idea to give the user the ability to change the time series
  + The table of contents is also independent on the change of time series
    - But I also want me software to give the max, min, and other key statistics of the selected time frame



* This is the yahoo finance API and has some features that extend from the Bloomberg screen
  + For example, this API allows the user to compare any graph they want to the FTSE 100 – the one on screen right now
  + The GUI also gives the user the ability to change way the graph is presented
    - From a line to an Area graph to a candle/ hollow candle and a bar/ coloured bar
  + The toolbar also gives the user the ability to change the colour of graph and background
    - It also gives the user to implement certain indicators onto the graph such as moving averages
    - The cursor – when hovering over the graph – shows key statistics of the day hovered over.
      * + It would be nice to be able to implement this into a GUI, so I can improve user experience



* Early on into my research I found a trading dashboard in Python already created. However, as seen above there are many ways of doing trading dashboards and not all are the same: for example, this one doesn’t have soe things that I would like to do such as:



* For example, this dashboard features an Entry/ Exit feature which gives the user an idea of when the best time to buy and sell was previously

## Specify the solution 3.1.4

### Requirements 3.1.4 a)

My Project will not use any external hardware and will run on a standard computer with PyCharm as the IDE and tkInter to do the GUI

### Success Criteria 3.1.4 b)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | | User Requirement | What does this do? | Link to SC |
| 1 | 1.1 | * Allows the user to choose a stock * yFinance has a large range of available stocks, almost all stocks in use | * Important for this stock dashboard to give the user free reign to choose what stock they want |  |
| 1.2 | * Allows the user to choose a time frame * yFinance has about 10 different time frames, ranging from * When it comes to graphing two stocks, the time frame will be limited to 3mo or max for both stocks | * Allows the user to change the graph time frame to their choice |  |
| 1.3 | * Give the user the ability to change the colour of the graph for both graphing one stock and two stocks | * Not too hard to code and gives the user more freedom on the dashboard |  |
| 2 | 2.1 | * Allows the user to choose different stocks to compare, allow to them to compare two different stocks | * Once again hugely important for a trading dashboard * Most of the dashboard that I researched gave the user the ability to compare two different graphs |  |
| 3 | 3.1 | * Give simple information to the user such as min, max and mean in the selected time frame | * Statistics are important and give the user more information on the dashboard |  |
| 3.2 | * Give more complex information to the user such as price change in their selected time frame * For example, if a stock started in a 3mo time frame at £10 and then at the present day was £15, I want to be able to print a statistic that says it grew 50% |  |  |
| 3.3 | * Finally, I would like to be able to give the user indictors such as Div Yield and P/E ratio as well as its market cap |  |  |

|  |  |
| --- | --- |
| **SC Number** | **Success Criteria** |
| 1 | The System should not take too long to load up the data; around 10 seconds maybe less |
| 2 | The System should have an appreciable looking GUI that is easy to navigate around |
| 3 | The System should have the ability for the user to successfully change the time that the graph shows, and the graphs should be easy to understand |
| 4 | Under the graphs should be an information tab that gives the user simple information such as Max in the Period and Min in that Period |
| 5 | The information tabs should also show much more detailed statistics |
| 6 | Possibly allow the user to change the colour of the graph |
| 7 | Allow the user to compare two different stocks |

# Design 3.2

## Decomposition of Problem 3.2.1

Before beginning my project, it was important to create a Decomposition list to use when creating the project

* GUI start page
  + Three buttons
    - One for graphing a singular stock
    - One for graphing two stocks
    - One to close the window
* One stock page
  + Three buttons once again
    - One to direct to a help page
    - One to direct to the two-stock page
    - One to direct to the homepage
  + Main part of the window will be taken up with the graph of the stock
  + On the side to the graph will be a list of key indictors in the selected time frame
  + The title at the top will say ‘One Stock’
* Two stock pages
  + Three buttons once again
    - One to direct to the same help page as the one stock button
    - One to direct to the one stock page
    - One to the direct to the homepage
  + Largely similar design to the one stock
  + 2 columns in which the stocks indicators will be compared
  + Hopefully will have a third column that will indicate which one is better off in that certain indicator

## Describe the solution 3.2.2

### Explain and Justify the structure of the solution 3.22 a)

* My solution will heavily rely on two key modules
  + yfinance – the successor to the popular yahoo finance module, this module holds all stock data from inception of that certain stock
  + Pandas – this module provides high performance data structures such as data frames and high-performance analysis tools
    - The data frame structure will be crucial to hold my stock data which will be used to calculate all my indictors
* The solution will also use a GUI to display the data in a user-friendly way
  + The GUI will contain my stock graph and a range of indictors from max to market cap
  + It will also allow the user to choose the stock, time series and the colour of the graph line

### Algorithms 3.2.2 b)

#### Class Tables 3.2.2 bi)

Firstly, I want to create a control line interface so the user can create the graph using inputs from the control interface. As this is before I have begun to code, it is likely these method and attributes names will change. However, to save time I will only implement the ability to graph a singular stock in the interface

|  |  |  |
| --- | --- | --- |
| Class: oSInterace | | |
| **Methods ()** | **Attributes** | **Explanations** |
| def \_\_init\_\_ () | Self.time\_series | Will hold the list of time series |
| Self.time | Set the variable time to be changed later by the user |
| def setOutStock() | Self. stocks | Will set out the list of stocks suggested to be chosen by the user, but they would be free to choose whatever they would like |
| def chooseStock() | Self.stockChoice | Gives an input for the user to choose the desired stock to graph |
| def chooseTimeSeries() | Self.timeChoice | Gives an input for the user to choose the desired time series |
| def styleGraph() | Self.colour | Sets out a list of colours available to style the graph |
| Self.colourChoice | Gives the user an input to choose the colour they would like to make the graph |
| def graph() | Self.stockticker | Changes the inputted stock into a ticker that is recognised by the yfinance module |
| Self.stockticker\_history | Downloads the history of the stock in the timeframe chosen in self.time |
| Self.dframe | Creates a data frame with columns date and open stock price |
| def simpleStats() | Self.column | Pulls the open column out of the data frame so it can be used to calculate the indicators and rounds it to 2dp |
| Max\_value | Calculates the maximum value of the stock in the time frame |
| Min\_value | Calculates the minimum value of the stock in the time frame and rounds it to 2dp |
| Mean\_value | Calculates the average value of the stock in the time frame and rounds it to 2dp |
| def calcPercentage() | Self.First | Stores the first data point in the open column of the data frame |
| Self.Last | Stores the last point in the open column of the data frame |
| Self.Change | Calculates the percentage difference between self.first and self.last |
| def indictorsPlus() | Self.yahooFinance | Changes the stock input into an input that can be read by the module yahoofinancials |
| Self.quoteTable | A variable that gets the price earnings ratio of the stock chosen |
| Self.marketCap | Variable that retrieves the market cap of the stock chosen |

|  |  |  |
| --- | --- | --- |
| Class: startPageGUI | | |
| **Methods()** | **Attributes()** | **Explanation** |
| def \_\_init\_\_() | N/A | N/A |
| def titlepage() | Self.title | Creates the title of the GUI front page ‘Stock Grapher’ |
| Self.oneStock | Creates a button to allow the user to go to the oneStock graph page |
| Self.twoStocks | Creates a button to allow the user to go to the twoStocks graph page |
| def grid() | Self.oneStock.grid | Grids the buttons and labels created in the title page method |
| Self.title.grid |
| Self.twoStocks.grid |
| def changeFrame() | container | Creates a container of the Frame |
| Self.frames | Creates an empty array for the frames |
| Pages | Creates an array of the frames |

During this process, I believed the CI interface would be enough for the GUI to work. However, I soon came to the realistion that I would need to create an inputless range of functions to process the data that is outputted by the user from inputs on the GUI. So, I came up with the test functions file.

During this, I also hit another roadblock in which while loading up my GUI once I had done my test function, it would take the GUI around a minute to load up. I came to the conclusion that the reason this was happening was that each time any of those functions were called it would create a whole new data frame, which takes up a lot of memory and overall makes it slower.

So, to fix this, I created a function with the sole purpose of creating a data frame and returning the necessary column to calculate my indictors. After this I found that it was much faster overall

This file didn’t lend itself to an Object Orientated Approach so instead I used a procedural programming approach with functions and returns

|  |  |  |  |
| --- | --- | --- | --- |
| File: TestFunctions | | | |
| **Procedure** | **Variables** | **Explanation** | **Return** |
| def getDataFrame | Ticker | Changes the stock input by the user into a readable input for the module yfinance | Returns the column of Dframe called Open  return df[‘Open’] |
| tickerHistory | Fetches the history of the stock in the time selected by the user |
| Dframe | Creates a dataframe using pandas with the column headings of date and open |
| def getMean | Column | Assigns the output of the getDataFrame function to a variable column | Returns the mean rounded to 2dp of the stock in the selected time  return meanRou |
|  | meanUnr | Gets the average of the stock in the time frame |
|  | meanRou | Rounds the mean to 2dp |

**Algorithms for code**

* Note while building these algorithms, I assumed all the required modules were implemented

function getDataFrame(stock):

ticker = yf.Ticker(stock)

tickerHistory = ticker.history(period='3mo')

sf = tickerHistory['Open']

df = pd.DataFrame({'Date': sf.index, 'Open' sf.values})

return df['Open']

function getMean(stock):

column = getDataFrame(stock)

meanUnr = column.mean()

meanRou = round(meanUnr, 2)

return meanRou

function getMin(stock):

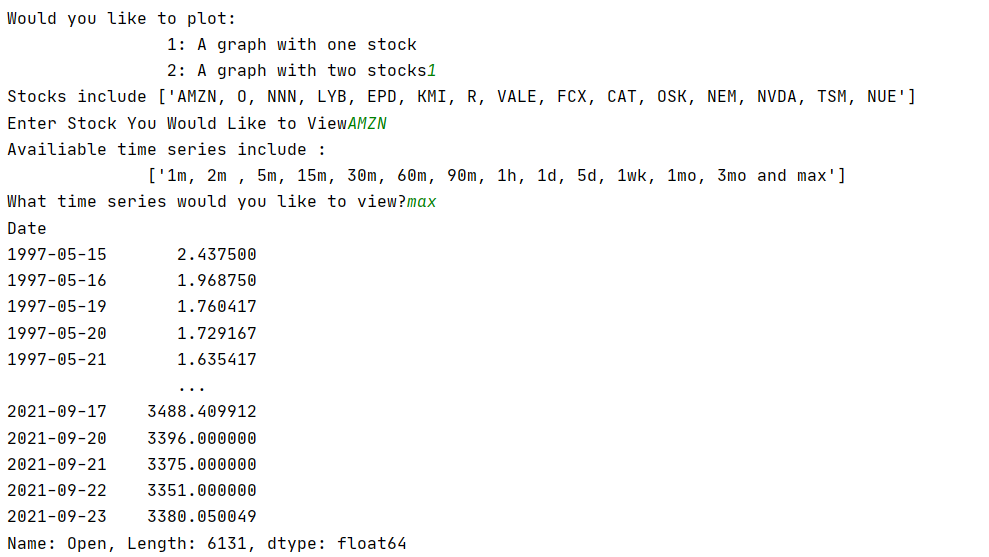
column = getDataFrame(stock)

minUnr = column.min()

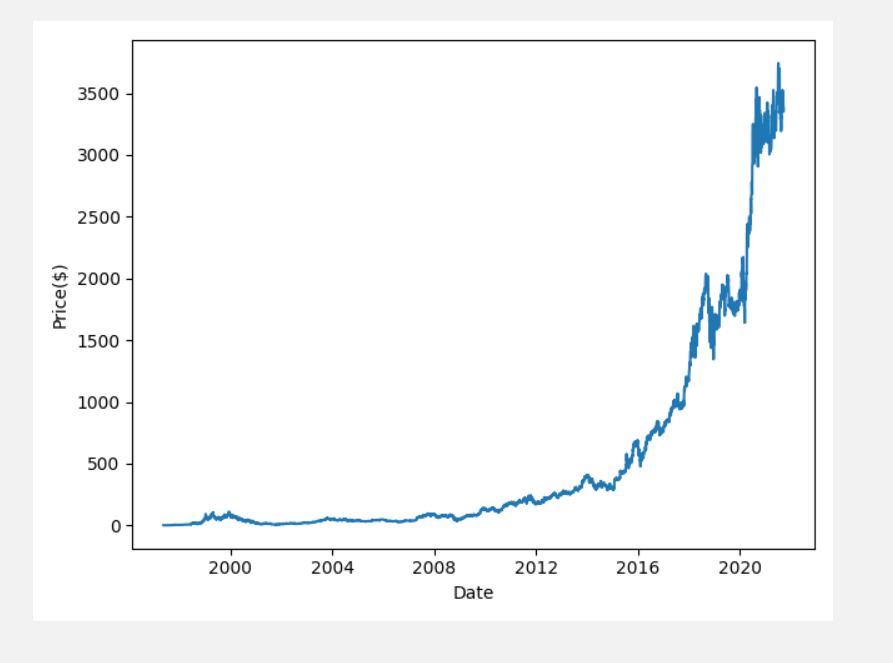
minRou = round(minUnr, 2)

return minRou

Initially began designing a CLinteface, which is basic but can graph a singular stock using a user preference of time series.



Interface with ability to choose stock and time series to show



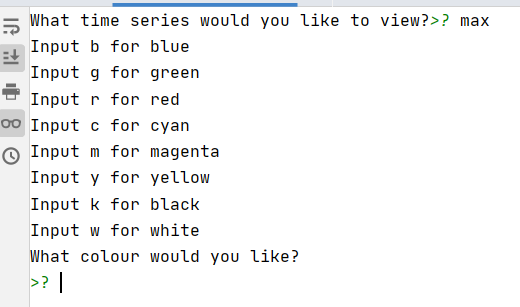
Output of the selected input of Amazon Stock and maximum time series

Link to Success Criteria by:

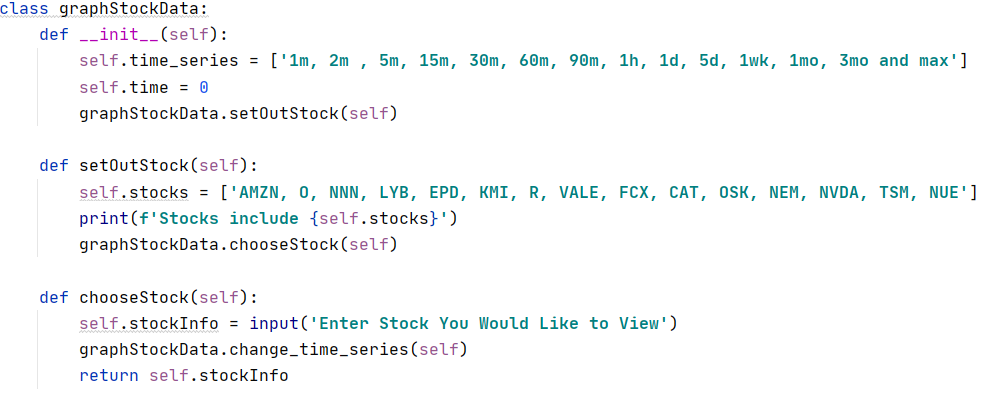
* The system does not take too long to up the data; almost instantaneously right now
* The system allows the user to change the time series
* However, a GUI is currently not present; but will in time
* Currently the data is presented within the console such as close on a particular date; however, I want to be able to show the min and max of the selected time series, which I will work on
* It also does not currently have an ability for the user to change the colour of the graph or compare two different stocks

***Initial Code***

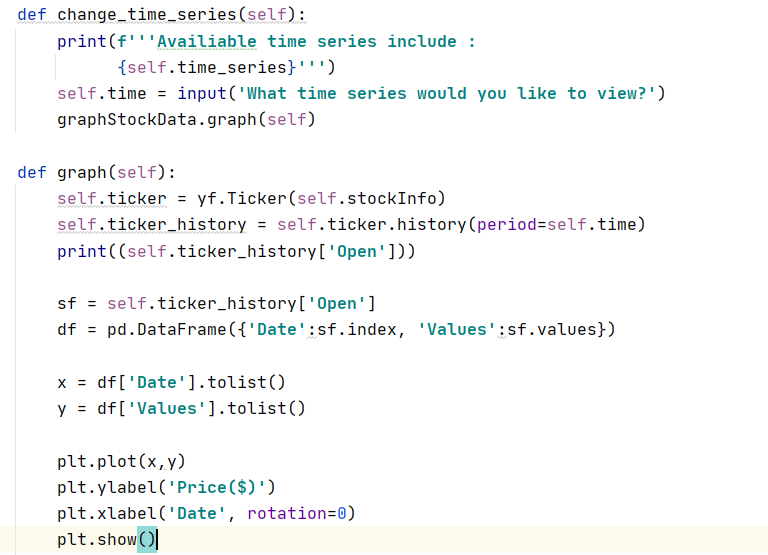
* Currently the code is help in a singular OOP class, with several methods and attributes
* The methods include:
  + def \_\_init\_\_() – this sets out the time variable and the available time series’
    - Attributes here include:
      * self.time\_series - creates an array with the time series available with module yfinance
      * self.time – creates a variable and sets it as zero
  + def setOutStock() – Prints out the stocks available
    - self.stocks – Creates an array with some stock abbreviations
  + def chooseStock() - Allows the user to input their stock they want to graph
    - self.stockInfo – an input to allow the user to input their desired stock
  + def changeTimeSeries() – Prints out the available time series’ and gives the user the option to input their desired time series
    - self.time – the variable set up earlier is now changed to the time series chosen by the user based on the ones in self.time\_series
  + def graphDesign() – Prints out the available colours for the graph and allows the user to choose the colour they would like
    - self.colourInitial – creates an array that has the initals of the colours that can be used according to Matplotlib
    - self.colour – creates an array with the colour names that can be used
      * I then used a for loop to print out the colours alongside their initials, as show below:



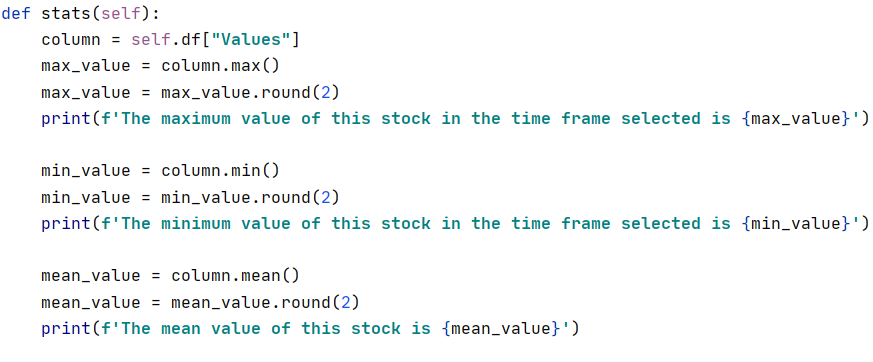
* + - self.colourChoice – Allows the user the choose the colour wanted, however the initial must be used rather than the full name of the colour
      * For example – ‘b’ or ‘k’
  + def graph() – Creates a Pandas DataFrame of the stock chosen and plots a graph with the user chosen stock, time series and colour choice
    - self.ticker – creates a ticker with the stock chosen by the user
    - self.ticker\_history – collects the history of the ticker during the time series chosen earlier
    - sf – collects the ticker history when it opens
    - self.df – creates a Pandas DataFrame with the columns of the date and the value when it opens on that certain day
    - x – puts the DataFrame column values of the Date on the X - Axis
    - y – puts the DataFrame column values of the value when the stock opens on the y - axis
  + def stats() – Outputs the max, min and mean stock price of the time series selected
    - column – selects the column with the values of when the stock opens on a particular day
    - max\_value – initially it is set to the biggest value in the column ‘open’ and after it rounds that value to two decimal places
    - min\_value – initially it is set to the smallest value in the column ‘open’ and after it rounds to two decimal points



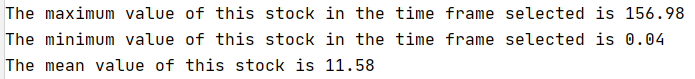
Code to allow the user to select the stock and the time series



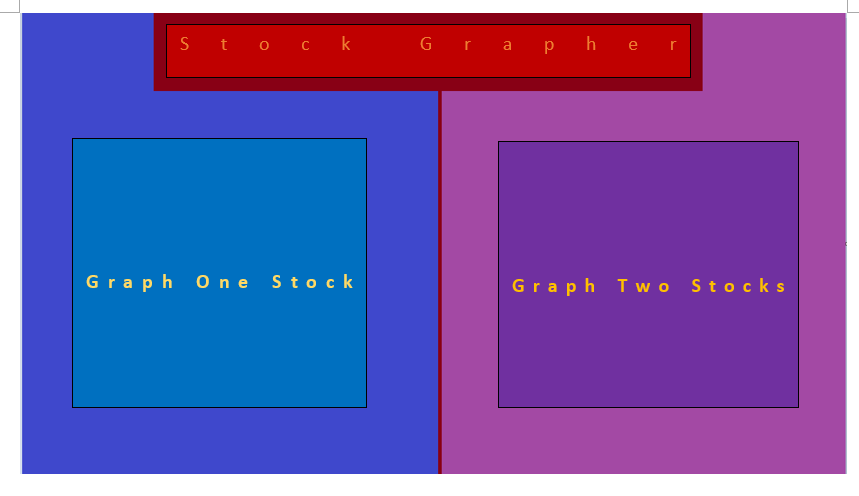
Shows the code used to output the graph with the stock and time series selected



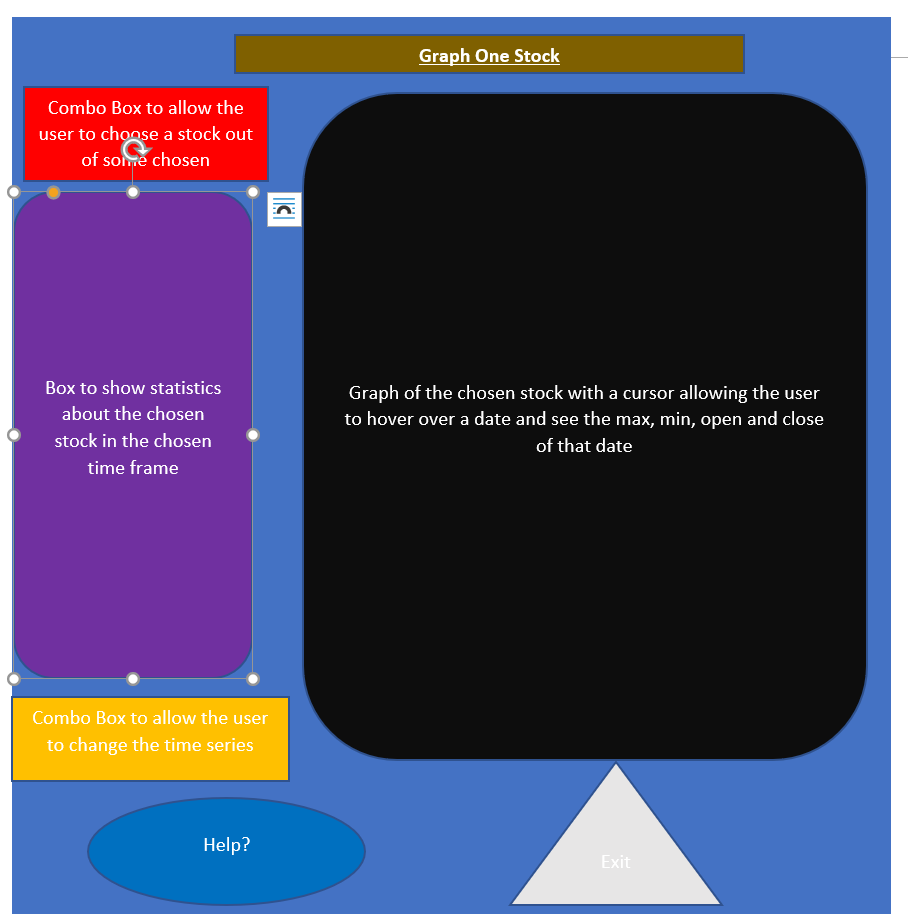
Shows the code to calculate key statistics for the time frame selected by the user



*Initial GUI design*

Shows an initial design for the title/front page of my GUI

* This is a graphical design of my title page of my GUI
  + The Graph One Stock and Graph Two Stocks will be buttons which will take the user to another Frame of the GUI
  + The title will be a label with the title of my program
  + Although not in this, there will be an exit button at the top of my screen which will allow the user to close the Frames

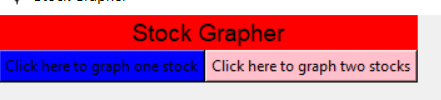


Shows an initial design of my Graph One Stock Frame

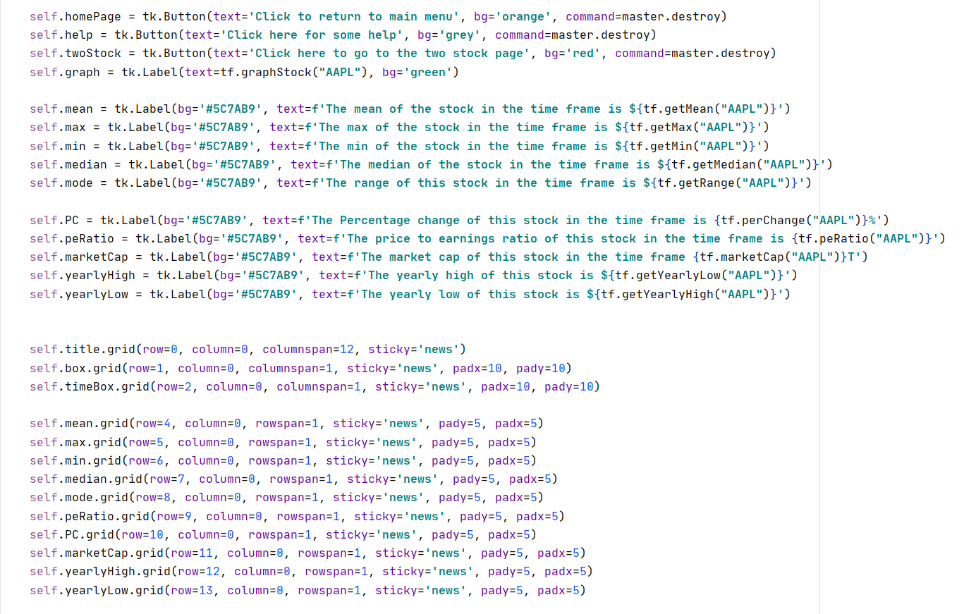
* This frame has a lot more widgets on it than the title frame as expected
  + The Statistics Widget/ Label will show key stats such as:
    - Mean
    - Mode
    - Median
    - Max
    - Min
    - % Change from start to finish
  + All these stats will be in the time series as selected by the user in the time series combobox which are:
    - ‘1m, 2m, 5m, 15m, 30m, 60m, 90m, 1h, 1d, 5d, 1wk, 1mo, 3mo and max’
  + And the stocks which will be available in the Combo Box would be:
    - AMZN – Amazon
    - AAPL – Apple
    - MSFT – Microsoft
    - GOOGL - Google
    - FB – Facebook
    - TSLA – Tesla
    - NVDA – Nvidia
  + The help button would give the user key FAQs for the One Stock Graph frame and would take the user to a whole new frame to give answers the questions
    - This would include:
      * What is this? – A program to allow for the graphing of a singular stock in a selected time frame
      * What do the stats mean? – These show key details for the stock chosen within the selected time frame

**Python GUI Design**

* First stage was to design the home page for my GUI.
  + It was not a long stage of development as it was a basic page, but it allows to user to direct themselves to either the one stock or two stocks pages
    - However, currently they only destroy the frame and not take the user to another frame because I haven’t implemented that yet



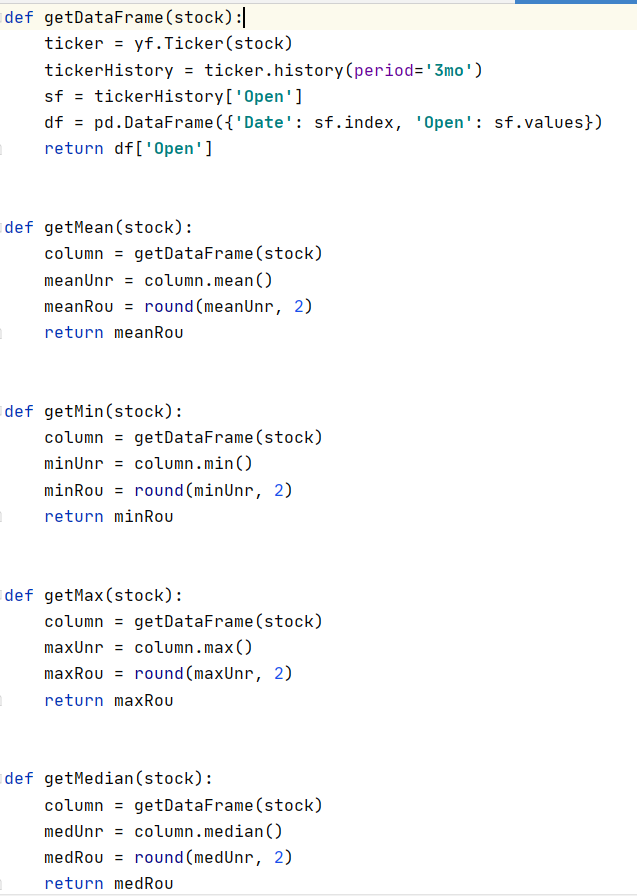
* + This is what my start page GUI currently looks like
* My second stage was going about designing my one stock page
  + Initially I wanted to create it in the same class as my main frame, however I quickly realised that doing that would be inefficient and unsustainable
  + Therefore, I ended up creating the code on another python file, and once it is finished and I feel happy with it – I will copy it over to the page with the start page code



* This is the first part of my oneStock code frame
* This sets out the code for the combobox of both the stock and timeseries box
* However, over time this could change to a tk.Stringvar() if I find out it is easier to use
* Also importing the test function file to allow me to call those functions from the current file
* This is the final snippet of my code which grids the buttons at the bottom of the screen
* This also binds to the box and timebox variable that calls a function once a value in combobox is chosen
* These two functions print both the current choice in the box in the console and a message showing the choice with the string ‘You have chosen {combobox choice}
* This is the second snippet of the code
* The part shows the numerous labels I created to show the indictors and other widgets
* This also shows how I called the functions to output the necessary indicators
* tf.getMean() access the getMean function on the test functions file and called the function getMean with the parament s ‘AAPL



* This is what the current one stock frame looks like – it is a work in progress
  + The two drop-down boxes at the top – which are called combo Boxes in tkinter – are for choosing the stock and time series respectively
    - Currently these boxes don’t have a set value when you run the code
    - When the code is finished it will allow the user to choose the stock they want to view and that will update the indicators with the time series in mind
    - However, currently I have been unsuccessful in getting the data from the box and transferring it into the label values, but I hope to get it sorted soon
    - Along with this, the large green rectangle is a placeholder of where I hope to put the graph within once I figure out how to do it, although my attempts now have been futile
    - Finally, the 3 buttons at the bottom currently destroy the frame but, in the end, they will take the user to another frame that has the text on the button
  + However, when I was making the indicators, I realised I had to write dedicated function that returned the output for the indicators



* This is a snippet of the numerous functions that I wrote to return the certain indicators taking the result from the combo box as a parameter
  + Initially each one of these functions created a data frame for the stock
    - However, realising that this was both inefficient and took up lots of memory
      * I created a single function that created the data frame, and that output is used in all the functions which streamlined my code