S&P 500 Insights by Sector

Group members: Moreno, Y.; Pronozin, L.; Merli, G.; El Nakadi, Y., and Lemli, N.

Department of Science and Technology, Ie University

PDMA: Programming for Data Management and Analysis

Dr. Robert Polding

December 5, 2024

Table of Contents

Abstract

We are creating a program that analyses the S&P 500, a stock market index that tracks the performance of the largest US companies listed on the stock exchange. Our objective is to create a user-friendly program with a graphical user interface that would visualize trends and relationships between variables selected by the user. We expect the visualizations to provide users with insights into how companies in different sectors perform compared to each other. However, please note that our data is outdated since stock information is very volatile and changes on a daily basis. So, please use our data to study the intricacies of the S&P 500, but keep note that we are educating our customers, not giving out financial advice.

Keywords

1. Sector
2. Relationships
3. Performance
4. Visualization
5. Interactive

Introduction

The S&P 500 is a market capitalization-weighted index of the 500 biggest publicly traded companies in the United States. It is considered one of the world´s most stable and best-performing equities, which has been listed on stock exchanges since the 1950s. The index has gained a lot of popularity as a result of its attractive and stable growth over decades, resulting in recognition and approval from investors all over the world. Nowadays, the S&P 500 has a market capitalization of over 50 trillion, with millions of people globally holding and investing in the index. It's an index that has eleven different sectors each adding their contribution to its overall performance. The performance in each sector is generally marked by macroeconomics factors, environmental regulations, and individual companies. For instance, the information Technology sector has constantly represented growth due to the emergence of digital transformation , while energy sectors performance changes along with the prices of oil and the need for energy on the earth. Therefore understanding the details and complications of these sectors offers a great perspective on the index’s overall trajectory.

For that reason, the project is eager to analyze and visualize the performance across these sectors using the power of data science and programming. Being students specializing in business and data analytics, we realize how important it is to take advantage of computational tools in finding out patterns and insights that might remain obscure in big and complex data. The program we have designed lets users interact with data from the S&P 500 through an intuitive graphical entrance that is easy to use. Users can select variables, configure visualization, and revise insights into trends that might have passed right by them.

But before starting we first decided to do a deep research on the composition and structure of the S&P 500. We researched the historical background and the current structure of the index from reliable sources such as Yahoo finance and Investopedia. Then we thought of what a user of our product would be interested in learning, which led us to the following research potential questions:

- *Which sector in the S&P 500 performed the best?*
- *Which is the most volatile sector?*
- *How much does each sector contribute to the market capitalization of the Index?*

Therefore, to carry out this analysis, we found a data set that included the companies in the Index as rows and variables indicating financial performance as columns. Then, we thought about the different types of visual analysis that could be carried out and brainstormed the different variables that would be interesting for our users to look at and get insights from. We concluded that we should try to include as many analysis options as possible but also have a balance between complexity, personalization possibilities, and user-friendliness. We tried implementing a real time dataset that was updated automatically by connecting the Yahoo finance webpage to our program. However when doing so the program didn't work as well and many visualizations weren't insightful, which is why we decided to keep with our original dataset. Based on our group discussions, we decided to base our analysis based on variables related to financial analysis and most capable of yielding actionable insights:

Variables: company, sector, market capitalisation, income (ttm), revenue (ttm), book value per share (mrq), dividend yield (annual), full-time employees, analysts mean recommendation (1=buy, 5=sell), current ratio (mrq), total debt to equity (mrq), long_term debt to equity (mrq), annual EPS growth past 5 years, annual sales growth past 5 years, return on asset (ttm), return on equity (ttm), return on investment (ttm), operating margin (ttm), shares outstanding, volume, performance (year), beta, average true range (14), volatility (month), current stock price.
(In the program, before starting the analysis the user will be asked whether he knows what all variables mean, if the user does not know, the definition of the unknown variable will appear.)

Graphs: Line plot, Scatter plot, Bar chart, Error plot, Histogram, Pie chart, Violin plot, Box plot

The analysis of these across the sectors would give a trend, and by that, performance could be compared and turned into decisions made with knowledge. One of the main challenges in the design of this program was how to balance such complexity with ease of use. There is so much information from the dataset that it requires careful consideration on just how to present it in the best way possible to be accessible and informative for the user. We accommodate different types of visualization as listed before depending on the analytical need. It also includes the feature of personalization of visualizations: variable selection, axis scales, and graph titles are some of the functions users can work with. This project scope is not limited to some functional program. But it is about contributing to a bigger understanding of financial data in an easier way and what it implies for the world. In turn we would like users to be empowered to dynamically delve into the

intricacies of the S&P 500 and create their own view. This goes to the vision of using data science as a tool for education and decision making.

Objectives

Requirements:

- Our program must have an introductory page including a brief description of what the program does.
- Our program should be able to clean the dataset getting rid of missing values
- The visualizations displayed by our program must be grouped by sectors.
- Users have to choose how many variables they want to analyze apart from the eleven sectors, and select the variables preferred.
- The program must be displayed in a GUI using the Python package graphics.py.
- Users must be able to customize different components of their final graph, for example, title and axis.
- Users must be allowed to quit the program at any time.

Constraints:

- The program requires the file to be named "snp500.csv" in order for it to function.
- Users must have prior knowledge of what the S&P 500 index is and its components, in order for this program to be useful to them.
- Users are limited to choosing up to two extra variables for analysis
- Our program doesn't fully function on Mac devices and is recommended to be used on window devices

Methodology

The methodology of the project lies in designing and complementing a Python program to meet the aims outlined of our project. The first step was making the pseudocode for the program so that it outlines the logical flow leading to the solution. It was necessary because it helps to have an efficient design prior to the coding part. The main function of this program is to walk the user through a series of different options: variable selection, selecting the type of visualization, and finally customizing the graph. Major sub functions include cleaning the data, selecting variables, and actually creating the graphs. For example, the function variable_selector will allow the user to specify which variables they want to analyze, while the function graph_creator will actually create the visualizations based on the user's input.

The design of the program inducted error handling and input validation. We made sure that our program was done to be as robust as possible. Therefore we have features that make it run seamlessly and avoid crashes because of invalid inputs. So, what we did was we added buttons only to the options that work, so every button you press works properly.

The graphical user interface was designed using the graphics.py library, which includes buttons, text boxes, and dropdown menus. This design makes it very usable, in that users can easily work

their way through the program. During the planning phase, mockups of the interface were created to visually see what layout and functionality of the GUI would look like.

In addition to that, to maximize the efficiency of our program, we made sure not to use functions within functions so that we don't use recursion that could lead us to potential stack overflows. This way we can ensure an efficient and well working program where we create our function separately and finally have a main function where the final user interface goes there.

Pseudocode:

1. Create a function called “main”
    a. Read csv file and store it in variable named “df”
    b. Call clean() function to clean “df”
    c. Show introduction message about uses of our program
    d. If user presses “next” button:
        i. Call num_of_var() function
        ii. If num_of_var() = = 0
            1. Default has been chosen (sector is used as variable)
            2. Call graph_selector(num_of_var(),sector)
            3. Call graph_creator(graph_selector())
            4. Call graph_editor(graph_creator())
        iii. If num_of_var() = = 1
            1. Call var_selector()
            2. Call graph_selector(num_of_var())
            3. Call graph_creator(graph_selector(),var_selector())
            4. Call graph_editor(graph_creator())
        iv. If num_of_var() = = 2
            1. Call var_selector()
            2. Call graph_selector(num_of_var())
            3. Call graph_creator(graph_selector(),var_selector())
            4. Call graph_editor(graph_creator())
    e. If user presses the quit button, close app

2. Create clean() function
    a. Take “df” as argument
    b. Find all NA’s and replace with respective mean

3. Create a function called num_of_var():
    a. Displays three choices to either select 0, 1 or 2 variables apart from the default variable (“sectors”)
    b. The number chosen is stored in the variable: “extra_variable_number”
    c. Finally, “extra_variable_number” is returned

4. Create a function called var_selector(extra_variable_number):
    a. “chosen_var” = [“Sector”]
    b. If the “extra_variable_number” = = 0:
        i. “chosen_var” will be returned
    c. If the “extra_variable_number” = = 1:
        i. The user is only going to be able to select one variable
        ii. The variable chosen will be appended in the list named “chosen_var”
        iii. The “chosen_var” list will be returned

    d. Else:
        i. The user is only going to be able to select two variables
        ii. The chosen variables will be appended in the variable “chosen_var” as a list
        iii. The list “chosen_var” will be returned

5. Create a function called graph_selector(extra_variable_number):
    a. If the “extra_variable_number” = = 0:
        i. The following options of graphs will be shown:
            1. Bar Chart
            2. Pie Chart

        ii. The type of graph chosen will be stored in the variable named “chosen_graph”

    b. Elif the “extra_variable_number” = = 1:
        i. The following options of graphs to plot will be shown:
            1. Bar Chart
            2. Line plot
            3. Boxplot
            4. Error plot
            5. Histogram
            6. Scatterplot
        ii. The type of graph chosen will be stored in the variable named “chosen_graph”

    c. Else: *(2 variables were chosen)*
            1. Scatterplot
            2. Line plot

        ii. The type of graph chosen will be stored in the variable named “chosen_graph”

d. The “chosen_graph” variable will be returned

6. Create a function called graph_creator(“chosen_graph”, “chosen_var”):

    a. Calls the appropriate function based on the “chosen_graph” return with variables as arguments of that function
    Eg.
    if “chosen_graph” = = “Scatter plot”:
        Call the scatterplot function with the chosen_var as a parameter
    if “chosen_graph” = = “Boxplot”:
        Call the boxplot function with the chosen_var as a parameter

7. Create a function called graph_editor(graph_creator)
    a. Asks user through interface to add title
    b. Asks user through interface to add x-label
    c. Asks user through interface to add y-label
    d. If “chosen graph” == Line plot
        i. Asks user through interface to add legend
        ii. Asks user through interface to add markers
    e. Asks user through interface to add colour
    f. If “chosen_graph” == bar chart
        i. Asks user through interface to add width
    g. If “chosen_graph” == pie chart
        i. Asks user through interface to add explode

8. Create a function graph_export(final_graph, export_format):
    a. Export the graph

9. Create a function scatter_plot(chosen_var, title, yaxis, xaxis,colour)
    a. If len(chosen_var) = = 2
        i. We create a scatter plot with two variables.
        The sector variable will be represented by the colour of the points
    b. Else: *(variables chosen = = 3)*
        i. We create a scatter plot with three variables.
        The sector variable will be represented by the colour of the points
    c. Export the graph

10. Create function line_plot(chosen_var, title, yaxis, xaxis, colour)
    a. If len(chosen_var) = = 2
        i. We create a line plot with two variables.

The sector variable will be represented by the each line

b. Else: *(variables chosen = = 3)*
   i. We create a line plot with three variables.
      The sector variable will be represented by each line
c. Export the graph

11. Create a function bar_chart(chosen_var, title, yaxis, xaxis, colour)
    a. If len(chosen_var) = = 1
       i. We create a bar chart with 1 variable.
          The sector variable will be represented by each bin and the y-axis would be a count.

    b. Else: *(variables chosen = = 2)*

       i. We create a bar chart with 2 variables.
          The sector variable will be represented by each bin and the y-axis would be the other variable.

    c. Export the graph

12. Create a function boxplot(chosen_var, title, yaxis, xaxis, colour)
    a. Create a boxplot with chosen_var as y-axis and sector as x-axis
    b. Export the graph

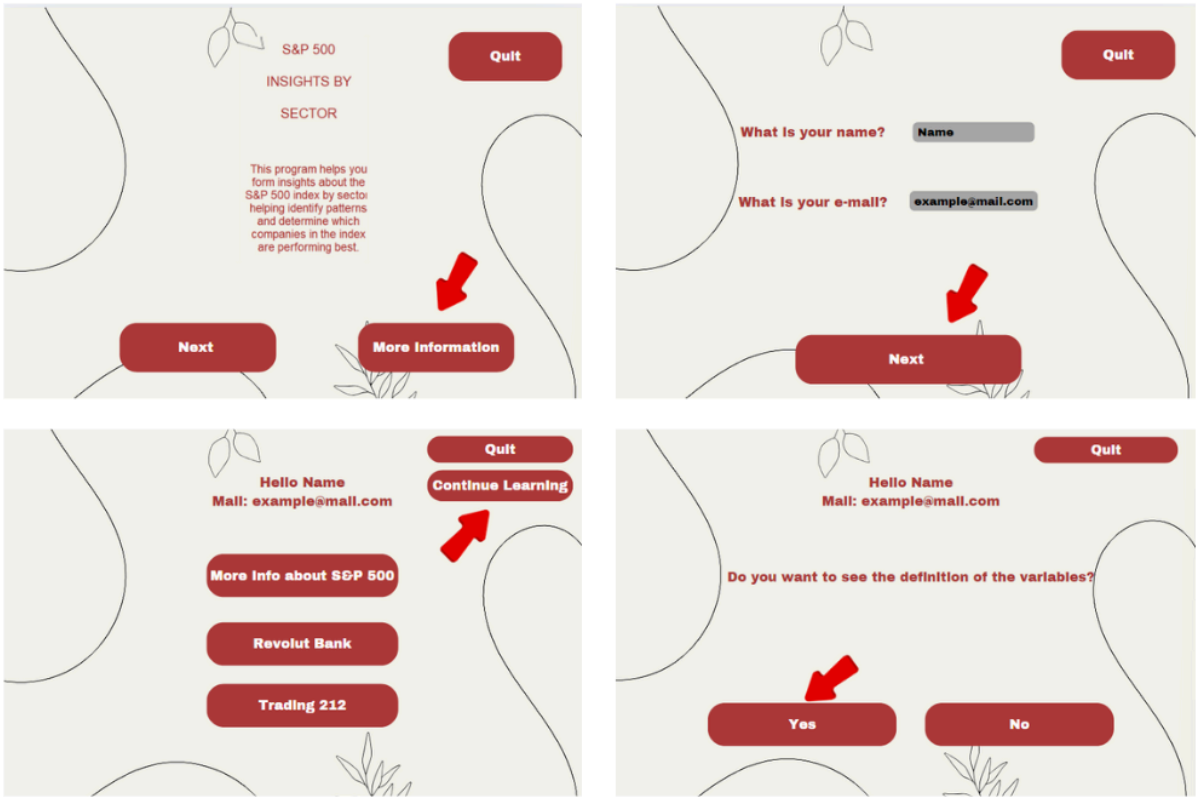
13. Create a function pie_chart(chosen_var, title, yaxis, xaxis, colour).
    a. Create a pie chart using as values the count of each sector and labelled appropriately
    b. Export the graph

14. Create a function histogram(chosen_var, title, yaxis, xaxis, colour)
    a. Draw histogram with sector as x-axis and chosen_var as y-axis
    b. Export the graph

15. Create a function violin_plot(chosen_var, title, yaxis, colour, colour)
    a. Create a violin plot with chosen_var as y-axis and sector as x-axis
    b. Export the graph

16. Create a function error_bar(chosen_var, title, yaxis, xaxis, colour)
    a. Create an error bar with chosen_var as x-axis and/or y-axis and sector as legend
    b. Export the graph

Mockups and Design



Above is the first thing the user sees, a welcome page, where the user can select next. This sends the user to the sign up page where they can enter their name and email, which is then used to greet them, creating a personalized experience for each user.

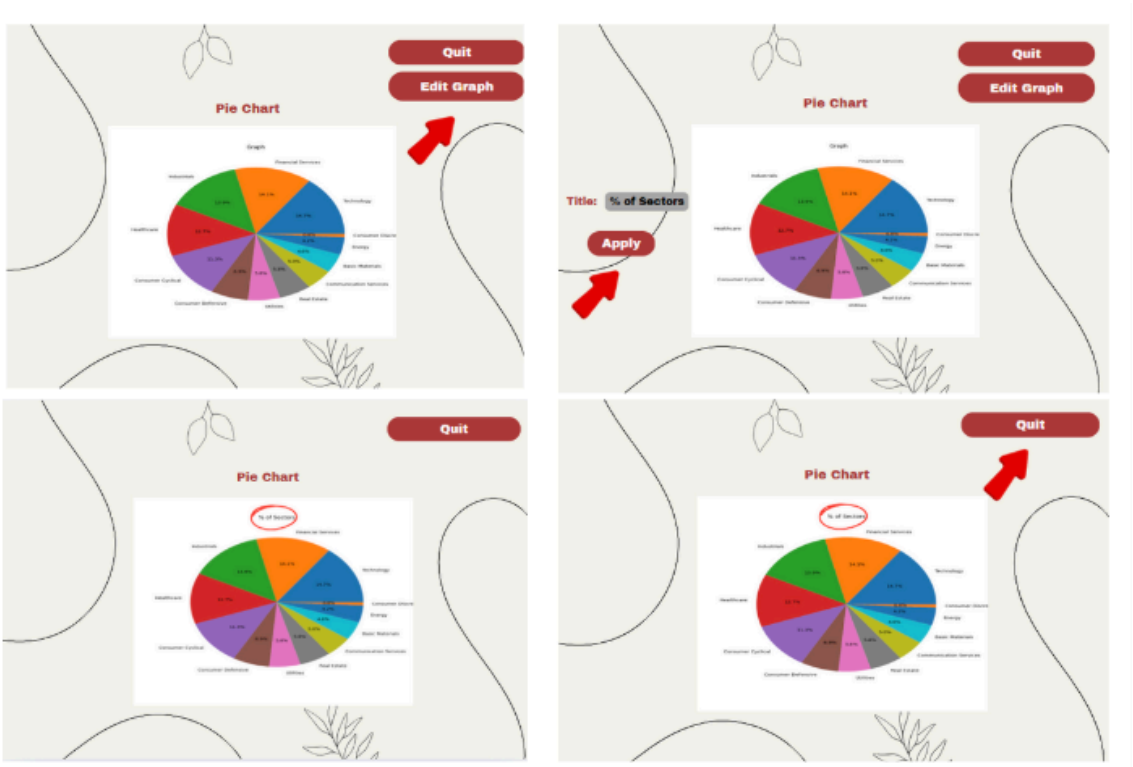
In order to make our program more user-friendly and in case users have no previous knowledge of the S&P 500 we decided to incorporate the 'More Information' button. This button will take the user to another page where they can select either "More Info about S&P 500", "Revolut Bank", or "Trading 212", which when clicked they redirect the user to three different websites depending on the one chosen, where they can investigate about finance and investing to their liking and at their own convenience. After this they can click "Continue Learning", and are asked if they want to see the definitions of the variables.

If they decide to see the definitions of the variables, there is an arrow included allowing the user to scroll up and down. This was done to improve not only the visual presentation but also to improve readability of the definitions.

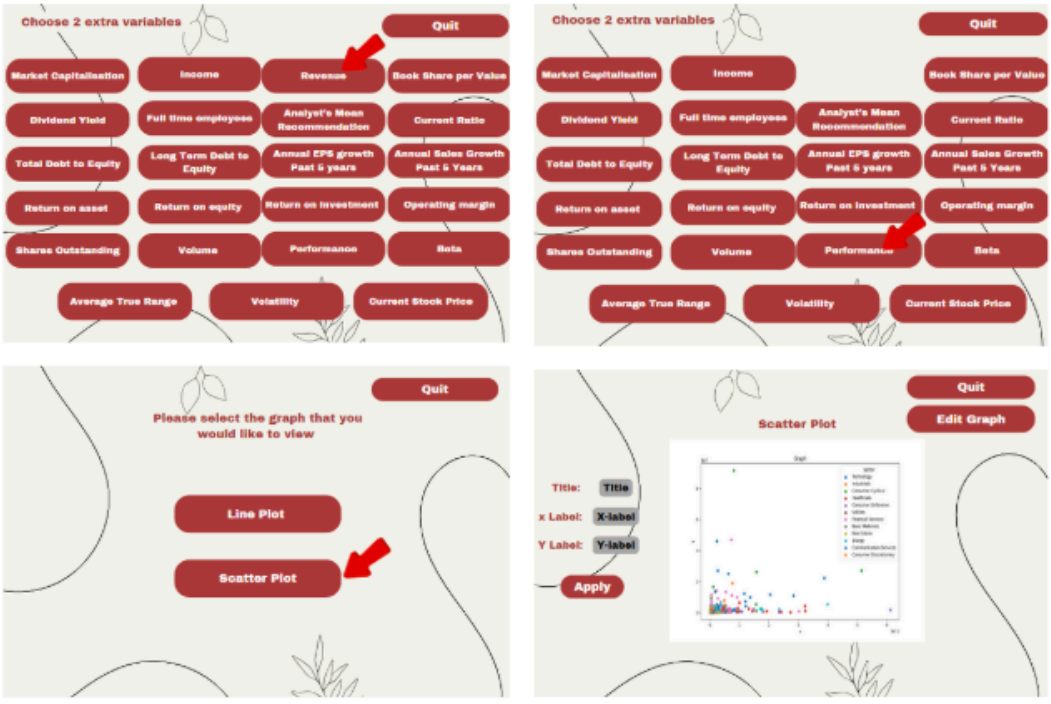
On the other hand, if the user chose "No" to view the definitions or if they click "Next" after viewing them, they are prompted to choose how many variables they want in addition to the predetermined one "Sectors". Depending on the amount of variables selected different options are given:

- If 0 is selected: then the user won't be able to select other variables and instead just choose the type of graph (as shown above).
- If the user chooses 1 or 2: they will be able to select the variables wanted and the type of graph to their preference.

In addition, to make it more interactive we added a feature where users are able to edit the graphs produced, while seeing the graph, by changing the title, x-axis, and y-axis (as shown below). Also in addition to everything, the users can quit the program at any point, without having to close the window or crashing the program.



Finally, to include error handling when the user chooses two additional variables, the first variable disappears after selection, ensuring that the same variable cannot be chosen twice. This prevents redundancy and ensures accurate and meaningful visualizations. This can be seen in the picture below.



Conclusion and Recommendations

To conclude everything, our program allows users to explore the complexity of the S&P 500, offering them a friendly interface that gives them accessibility to it. By making them visualize and simplifying the analysis, it will allow them to identify trends and gain valuable insight to drive them to make smart investment decisions.

Users can explore the data, at their own pace, by selecting variables and a visualization that best represents sectoral contributions, performance trends, and relationships among key indicators. We encourage people to analyze and interpret trends in finance themselves to trigger informed decisions and curiosity about the inner mechanisms driving the stock market. This program is something more than an analytical tool but a learning and exploration platform, opening ways to more intelligent investment strategies and deep insight into financial markets.

In the future, new versions of this tool could be developed to include real-time data feeds for the S&P 500. Other features that would add value to this tool are predictive analytics and sector-specific recommendations. The program could also include advanced customization options, integrating support for more datasets, such as global market indices, to increase appeal to a wide range of users. The continuous collection of feedback from the users will go a long way in refining and expanding the functionality of this tool, therefore helping the tool remain very useful to both the amateur and professional alike in the financial landscape.

Bibliography

Bloomberg. (n.d.). *SPX:IND overview*. Retrieved June 17, 2024, from https://www.bloomberg.com/quote/SPX:IND

Business Insider. (n.d.). *What is the S&P 500?*. Retrieved June 17, 2024, from https://www.businessinsider.com/personal-finance/investing/what-is-the-sp-500

Deloitte. (n.d.). *Insights*. Retrieved June 17, 2024, from https://www2.deloitte.com/insights

Finance Yahoo. (n.d.). *S&P 500 (^GSPC) stock price*. Retrieved June 17, 2024, from https://finance.yahoo.com/quote/%5EGSPC/

Forbes. (n.d.). *Finance*. Retrieved June 17, 2024, from https://www.forbes.com/finance

Harvard Business Review. (n.d.). *HBR*. Retrieved June 17, 2024, from https://hbr.org

IBM. (n.d.). *Predictive analytics*. Retrieved June 17, 2024, from https://www.ibm.com/topics/predictive-analytics

Investopedia. (n.d.-a). *S&P 500 definition*. Retrieved June 17, 2024, from https://www.investopedia.com/terms/s/sp500.asp

Investopedia. (n.d.-b). *History of the S&P 500*. Retrieved June 17, 2024, from https://www.investopedia.com/ask/answers/041015/what-history-sp-500.asp

Khan Academy. (n.d.). *Khan Academy*. Retrieved June 17, 2024, from https://www.khanacademy.org

NASDAQ. (n.d.). *NASDAQ*. Retrieved June 17, 2024, from https://www.nasdaq.com

NerdWallet. (n.d.). *Best investing tools*. Retrieved June 17, 2024, from https://www.nerdwallet.com/best-investing-tool

Revolut. (n.d.). *Revolut*. Retrieved June 17, 2024, from https://www.revolut.com/

Reuters. (n.d.). *Reuters*. Retrieved June 17, 2024, from https://www.reuters.com

Tableau. (n.d.). *Data visualization*. Retrieved June 17, 2024, from https://www.tableau.com/learn/articles/data-visualization

TechCrunch. (n.d.). *TechCrunch*. Retrieved June 17, 2024, from https://techcrunch.com

Trading212. (n.d.). *Trading 212*. Retrieved June 17, 2024, from https://www.trading212.com/es

World Economic Forum. (n.d.). *World Economic Forum*. Retrieved June 17, 2024, from https://www.weforum.org

World P/E Ratio. (n.d.). *S&P 500 sectors*. Retrieved June 17, 2024, from https://worldperatio.com/sp-500-sectors/

Appendix

For better visibility of the: Mockups

Code:

```
#S&P 500 project
#Yago, Gio, Lev, Nicolás and Yara

import matplotlib.pyplot as plt
import pandas as pd
from graphics import *
import numpy as np
import webbrowser as webs
from wand.image import Image as im  #For this to work it is
necessary to install the Python Package "Wand"
```

```
                                    #Also it is necessary to
install the ImageMagick.exe included in the code directory

def main():
   #Importing the data set
   df = pd.read_csv("snp500.csv")

   #Calling the clean function handle missing data
   clean(df)

   #Opening the window for GUI and setting the coordinates
   win = GraphWin("S&P500", 1500, 750)
   win.setCoords(0.0,0.0,50.00,50.00)

   #Changing the background of the window using an image that is
in the code directory
   background = Image(Point(25, 25), "background_image.png")
   background.draw(win)

   #Calling the introduction function
   quitx1, quitx2, quity1, quity2, invest = introduction(win)

   #Moving on to the login page
   name, mail = login_page(quitx1, quitx2, quity1, quity2, win)

   #If the user has decided to see web pages to invest, we show
it in the GUI
   if invest == "yes":
       invest_function(quitx1, quitx2, quity1, quity2, win,
name, mail)

   #Here the user is going to decide if he wants to see the
definitions of the variables
   definitions = show_def_choice(win, quitx1, quitx2, quity1,
quity2, name, mail)

   #If the user decides to see the definitions of the variables,
we call that function
   if definitions == "Yes":
       var_definition(win, quitx1, quitx2, quity1, quity2)
```

```
   #We call the function so that the user can choose the number
of extra variables he wants to choose
   extra_variable_number = num_of_var(win, quitx1, quitx2,
quity1, quity2)

   #Based on the output of the before function, the user chooses
the variables
   chosen_var = var_selector(win, quitx1, quitx2, quity1,
quity2, extra_variable_number, df)

   #Based on the number of variables, the user chooses the graph
for the analysis
   chosen_graph = graph_selector(win, quitx1, quitx2, quity1,
quity2, extra_variable_number)

   #Here establish a predefined title, x-label and y-label for
the graph
   title = "Graph"
   x_label = "x"
   y_label = "y"

   #This count will act as flag to know in which part of the
code we have to enter afterwards
   count = 0

   #We enter this part of the code if the chosen graph is a Line
Plot
   if chosen_graph == "Line Plot":
       #Here we are setting some coordinates and drawing some
buttons for the GUI
       editx1 = 40
       edity1 = 43
       editx2 = 49
       edity2 = 39

       edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
       edit.setFill(color_rgb(139,69, 19))
       edit.draw(win)

       edittext = Text(Point(44.5, 41), "Edit graph")
       edittext.setFace("arial")
```

```
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #Here we are calling the function to make a Line Plot
        count = lineplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

        while True:
            #Here we are getting a click
            click = win.getMouse()

            #Based on the coordinates of the click, we are making
different actions
            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
                edittext.undraw()

                #Here we call the function to edit the graph
                title, x_label, y_label = graph_editor(win,
chosen_graph, quitx1, quitx2, quity1, quity2)

                #Then we show the updated graph
                lineplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

            #If the clicks are in the quit button, we quit the
program
            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

    #We enter this part of the code if the chosen graph is a Box
Plot
    elif chosen_graph == "Box Plot":
        editx1 = 40
        edity1 = 43
        editx2 = 49
        edity2 = 39
```

```
        edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
        edit.setFill(color_rgb(139,69, 19))
        edit.draw(win)

        edittext = Text(Point(44.5, 41), "Edit graph")
        edittext.setFace("arial")
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #Here we call the function to make the Box Plot
        count = boxplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

        while True:
            click = win.getMouse()

            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
                edittext.undraw()

                #Here we are calling the graph editor function
                title, x_label, y_label = graph_editor(win,
chosen_graph, quitx1, quitx2, quity1, quity2)

                #Here we are applying the changes of the graph
editor function to the graph
                boxplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

    #We enter this part of the code if the chosen graph is a
Histogram
    elif chosen_graph == "Histogram":
```

```
        editx1 = 40
        edity1 = 43
        editx2 = 49
        edity2 = 39

        edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
        edit.setFill(color_rgb(139,69, 19))
        edit.draw(win)

        edittext = Text(Point(44.5, 41), "Edit graph")
        edittext.setFace("arial")
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #We call the function to ake the histogram
        count = histogram(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)

        while True:
            click = win.getMouse()

            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
                edittext.undraw()

                #We call the graph editor function
                x_label, y_label = graph_editor_histogram(win,
chosen_graph, quitx1, quitx2, quity1, quity2)

                #We apply the changes
                histogram(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()
```

```
   #We enter this part of the code if the chosen graph is a Pie
Chart
   elif chosen_graph == "Pie Chart":
       editx1 = 40
       edity1 = 43
       editx2 = 49
       edity2 = 39

       edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
       edit.setFill(color_rgb(139,69, 19))
       edit.draw(win)

       edittext = Text(Point(44.5, 41), "Edit graph")
       edittext.setFace("arial")
       edittext.setStyle("bold")
       edittext.setTextColor("white")
       edittext.setSize(17)
       edittext.draw(win)

       #We make the Pie Chart
       count = piechart(chosen_var, df, title, win, quitx1,
quitx2, quity1, quity2,count)

       while True:
           click = win.getMouse()

           if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
               edit.undraw()
               edittext.undraw()

               #We edit the graph
               title = graph_editor_pie(win, quitx1, quitx2,
quity1, quity2)

               #We apply changes
               piechart(chosen_var, df, title, win, quitx1,
quitx2, quity1, quity2,count)

           elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
```

```
                win.close()
                quit()

    #We enter this part of the code if the chosen graph is a
Violin Plot
    elif chosen_graph == "Violin Plot":
        editx1 = 40
        edity1 = 43
        editx2 = 49
        edity2 = 39

        edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
        edit.setFill(color_rgb(139,69, 19))
        edit.draw(win)

        edittext = Text(Point(44.5, 41), "Edit graph")
        edittext.setFace("arial")
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #We make the Violin Plot
        count = violinplot(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)

        while True:
            click = win.getMouse()

            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
                edittext.undraw()

                #We edit the graph
                title, x_label, y_label = graph_editor(win,
chosen_graph, quitx1, quitx2, quity1, quity2)

                #We update the graph
                violinplot(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)
```

```
            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

    #We enter this part of the code if the chosen graph is an
Error Plot
    elif chosen_graph == "Error Plot":
        editx1 = 40
        edity1 = 43
        editx2 = 49
        edity2 = 39

        edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
        edit.setFill(color_rgb(139,69, 19))
        edit.draw(win)

        edittext = Text(Point(44.5, 41), "Edit graph")
        edittext.setFace("arial")
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #We create the error plot
        count =errorplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

        while True:
            click = win.getMouse()

            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
                edittext.undraw()

                #We edit the graph
                title, x_label, y_label = graph_editor(win,
chosen_graph, quitx1, quitx2, quity1, quity2)
```

```
                #We apply changes
                errorplot(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

    #We enter this part of the code if the chosen graph is a Bar
Plot
    elif chosen_graph == "Bar Plot":
        editx1 = 40
        edity1 = 43
        editx2 = 49
        edity2 = 39

        edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
        edit.setFill(color_rgb(139,69, 19))
        edit.draw(win)

        edittext = Text(Point(44.5, 41), "Edit graph")
        edittext.setFace("arial")
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #We create the bar plot
        count =barplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

        while True:
            click = win.getMouse()

            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
                edittext.undraw()

                #We edit the graph
```

```
                title, x_label, y_label = graph_editor(win,
chosen_graph, quitx1, quitx2, quity1, quity2)

                #We apply changes
                barplot(chosen_var, df, title, x_label, y_label,
win, quitx1, quitx2, quity1, quity2,count)

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

    # We enter this part of the code if the chosen graph is a
Scatter Plot
    else:
        editx1 = 40
        edity1 = 43
        editx2 = 49
        edity2 = 39

        edit = Rectangle(Point(editx1, edity1), Point(editx2,
edity2))
        edit.setFill(color_rgb(139,69, 19))
        edit.draw(win)

        edittext = Text(Point(44.5, 41), "Edit graph")
        edittext.setFace("arial")
        edittext.setStyle("bold")
        edittext.setTextColor("white")
        edittext.setSize(17)
        edittext.draw(win)

        #We make the scatterplot
        count =scatterplot(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)

        while True:
            click = win.getMouse()

            if editx1 <= click.getX() <= editx2 and edity2 <=
click.getY() <= edity1:
                edit.undraw()
```

```
                edittext.undraw()

                #We edit the graph
                title, x_label, y_label = graph_editor(win,
chosen_graph, quitx1, quitx2, quity1, quity2)

                #We update the graph
                scatterplot(chosen_var, df, title, x_label,
y_label, win, quitx1, quitx2, quity1, quity2,count)

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

def clean(df):
    #We select the numeric columns
    numeric_cols = df.select_dtypes(include=['number']).columns

    #We substitute the missing data with the mean
    df[numeric_cols] =
df[numeric_cols].fillna(df[numeric_cols].mean())

    #We return the cleaned data set
    return df

def introduction(win):
    #We design the interface
    t1 = Text(Point(25, 40), "S&P 500\n\nINSIGHTS BY\n\nSECTOR")
    t1.setSize(20)
    t1.setTextColor("brown")
    t1.draw(win)

    t2 = Text(Point(25, 23), "This program helps you\nform
insights about the\n"
                             "S&P 500 index by sector,\nhelping
identify patterns\n"
                             "and determine which\ncompanies in
the index\nare performing best.")
    t2.setSize(17)
    t2.setTextColor("brown")
    t2.draw(win)
```

```
   continuex1 = 5
   continuey1 = 10
   continuex2 = 20
   continuey2 = 5

   r1 = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
   r1.setFill(color_rgb(139,69, 19))
   r1.draw(win)

   nexttext = Text(Point(12.5, 7.5), "Next")
   nexttext.setSize(17)
   nexttext.setFace("arial")
   nexttext.setStyle("bold")
   nexttext.setTextColor("white")
   nexttext.draw(win)

   investx1 = 30
   investy1 = 10
   investx2 = 45
   investy2 = 5

   r2 = Rectangle(Point(investx1, investy1), Point(investx2,
investy2))
   r2.setFill(color_rgb(139,69, 19))
   r2.draw(win)

   investtext = Text(Point(37.5, 7.5), "More information")
   investtext.setSize(17)
   investtext.setFace("arial")
   investtext.setStyle("bold")
   investtext.setTextColor("white")
   investtext.draw(win)

   quitx1 = 40
   quity1 = 49
   quitx2 = 49
   quity2 = 45

   qt = Rectangle(Point(quitx1, quity1), Point(quitx2, quity2))
   qt.setFill(color_rgb(139,69, 19))
```

```
    qt.draw(win)

    quittext = Text(Point(44.5, 47), "Quit")
    quittext.setFace("arial")
    quittext.setStyle("bold")
    quittext.setTextColor("white")
    quittext.setSize(17)
    quittext.draw(win)

    while True:
        click = win.getMouse()

        #Based on the coordinates of the click, the user can
decide if he wants to invest or no
        if continuex1 <= click.getX() <= continuex2:
            if continuey2 <= click.getY() <= continuey1:
                invest = "no"
                break

        #Here the user can close the program
        elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

        elif investx1 <= click.getX() <= investx2 and investy2 <=
click.getY() <= investy1:
            invest = "yes"
            break

    nexttext.undraw()
    t1.undraw()
    t2.undraw()
    r1.undraw()
    r2.undraw()
    investtext.undraw()

    #Here we return some coordinates to reuse them during the
program and also the decision of
    #investing or not
    return quitx1, quitx2, quity1, quity2, invest
```

```
def login_page(quitx1, quitx2, quity1, quity2, win):
   #We design the GUI and we set the texts of the Entries
   g1 = Text(Point(20, 30), 'What is your name?')
   g1.setSize(17)
   g1.setTextColor("brown")
   g1.draw(win)
   e1 = Entry(Point(30, 30), 30)
   e1.draw(win)
   e1.setText("Name")

   g2 = Text(Point(20, 20), 'What is your e-mail?')
   g2.setSize(17)
   g2.setTextColor("brown")
   g2.draw(win)
   e2 = Entry(Point(30, 20), 30)
   e2.draw(win)
   e2.setText("example@mail.com")

   continuex1 = 15
   continuey1 = 10
   continuex2 = 35
   continuey2 = 5

   r1 = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
   r1.setFill(color_rgb(139,69, 19))
   r1.draw(win)

   nexttext = Text(Point(25, 7.5), "Next")
   nexttext.setSize(17)
   nexttext.setFace("arial")
   nexttext.setStyle("bold")
   nexttext.setTextColor("white")
   nexttext.draw(win)

   while True:
       click = win.getMouse()

       if continuex1 <= click.getX() <= continuex2:
           if continuey2 <= click.getY() <= continuey1:
               break
```

```
       elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

   #If the user decides not to quit, we get the text of the
entries, either the predefined one or the one that
   #the user gives
   name = e1.getText()
   mail = e2.getText()

   g1.undraw()
   g2.undraw()
   e1.undraw()
   e2.undraw()
   nexttext.undraw()
   r1.undraw()

   return name, mail

def invest_function(quitx1, quitx2, quity1, quity2, win, name,
mail):
   #Adjusting the GUI
   string = "Hello " + name + "!"
   string2 = "Mail: " + mail

   n = Text(Point(25, 45), string)
   n.setSize(17)
   n.setTextColor("brown")
   n.draw(win)

   e = Text(Point(25, 40), string2)
   e.setSize(17)
   e.setTextColor("brown")
   e.draw(win)

   rectx1, rectx2 = 20,30
   snpy1,snpy2 = 35,30
   revoluty1, revoluty2 = 25, 20
   tradingy1, tradingy2 = 15,10

   r1 = Rectangle(Point(rectx1, snpy1), Point(rectx2, snpy2))
```

```
   r1.setFill(color_rgb(139,69, 19))
   r1.draw(win)
   r2 = Rectangle(Point(rectx1, revoluty1), Point(rectx2,
revoluty2))
   r2.setFill(color_rgb(139,69, 19))
   r2.draw(win)
   r3 = Rectangle(Point(rectx1, tradingy1), Point(rectx2,
tradingy2))
   r3.setFill(color_rgb(139,69, 19))
   r3.draw(win)

   snpinfo = Text(Point((rectx1+rectx2)/2, (snpy1+snpy2)/2),
"More info about S&P 500")
   snpinfo.setFace("arial")
   snpinfo.setStyle("bold")
   snpinfo.setTextColor("white")
   snpinfo.setSize(17)
   snpinfo.draw(win)

   revolut = Text(Point((rectx1+rectx2)/2,
(revoluty1+revoluty2)/2), "Revolut bank")
   revolut.setFace("arial")
   revolut.setStyle("bold")
   revolut.setTextColor("white")
   revolut.setSize(17)
   revolut.draw(win)

   trading = Text(Point((rectx1+rectx2)/2,
(tradingy1+tradingy2)/2), "Trading 212")
   trading.setFace("arial")
   trading.setStyle("bold")
   trading.setTextColor("white")
   trading.setSize(17)
   trading.draw(win)

   continuex1 = 40
   continuey1 = 43
   continuex2 = 49
   continuey2 = 39

   continuerectangle = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
```

```
    continuerectangle.setFill(color_rgb(139,69, 19))
    continuerectangle.draw(win)

    continuetext = Text(Point(44.5, 41), "Continue learning")
    continuetext.setFace("arial")
    continuetext.setStyle("bold")
    continuetext.setTextColor("white")
    continuetext.setSize(17)
    continuetext.draw(win)

    while True:
        click = win.getMouse()

        #Depending on the coordinate os the click, we open a
webpage or another
        if rectx1 <= click.getX() <= rectx2 and snpy2 <=
click.getY() <= snpy1:

webs.open("https://www.businessinsider.com/personal-finance/inve
sting/what-is-the-sp-500#:~:text=The%20S%26P%20500%20is%20a,gaug
e%20performance%20of%20other%20assets.")

        elif rectx1 <= click.getX() <= rectx2 and revoluty2 <=
click.getY() <= revoluty1:
            webs.open("https://www.revolut.com/")

        elif rectx1 <= click.getX() <= rectx2 and tradingy2 <=
click.getY() <= tradingy1:
            webs.open("https://www.trading212.com/es")

        #Here we continue with the program
        elif continuex1 <= click.getX() <= continuex2 and
continuey2 <= click.getY() <= continuey1:
            break

        #Here we quit the program
        elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    r1.undraw()
```

```
    r2.undraw()
    r3.undraw()
    snpinfo.undraw()
    revolut.undraw()
    trading.undraw()
    continuerectangle.undraw()
    continuetext.undraw()
    n.undraw()
    e.undraw()

def show_def_choice(win, quitx1, quitx2, quity1, quity2, name,
mail):
    #Here we set the GUI
    string = "Hello " + name + "!"
    string2 = "Mail: " + mail

    n = Text(Point(25, 45), string)
    n.setSize(17)
    n.setTextColor("brown")
    n.draw(win)

    e = Text(Point(25, 40), string2)
    e.setSize(17)
    e.setTextColor("brown")
    e.draw(win)

    t1 = Text(Point(25, 30), "Do you want to see the definitions
of the variables?")
    t1.setSize(17)
    t1.setTextColor("brown")
    t1.draw(win)

    yesx1, yesx2 = 13, 24
    nox1, nox2 = 26, 37
    y1, y2 = 15, 10

    ryes = Rectangle(Point(yesx1, y1), Point(yesx2, y2))
    ryes.setFill(color_rgb(139,69, 19))
    ryes.draw(win)

    yestext = Text(Point(((yesx1+yesx2)/2),((y1+y2)/2)), "Yes")
    yestext.setFace("arial")
```

```
    yestext.setStyle("bold")
    yestext.setTextColor("white")
    yestext.setSize(17)
    yestext.draw(win)

    rno = Rectangle(Point(nox1, y1), Point(nox2, y2))
    rno.setFill(color_rgb(139,69, 19))
    rno.draw(win)

    notext = Text(Point(((nox1+nox2)/2), ((y1+y2)/2)), "No")
    notext.setFace("arial")
    notext.setStyle("bold")
    notext.setTextColor("white")
    notext.setSize(17)
    notext.draw(win)

    while True:
        click = win.getMouse()

        #Depending on the coordinates of the click, the user can
choose to see the definition of the variables
        if yesx1 <= click.getX() <= yesx2:
            if y2 <= click.getY() <= y1:
                definitions = "Yes"

                break

        elif nox1 <= click.getX() <= nox2:
            if y2 <= click.getY() <= y1:
                definitions = "No"

                break

        elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    t1.undraw()
    yestext.undraw()
    ryes.undraw()
    rno.undraw()
```

```
    notext.undraw()
    n.undraw()
    e.undraw()

    return definitions

def var_definition(win, quitx1, quitx2, quity1, quity2):
    #We design the GUI
    title = Text(Point(25, 48), "Variables Definition")
    title.setSize(20)
    title.setTextColor("brown")
    title.draw(win)

    t1 = Text(Point(25, 45), "Company: The name of the business
entity being analyzed.")
    t2 = Text(Point(25, 42),
              "Sector: The segment of the economy in which the
company operates,\nsuch as technology, healthcare, or finance.")
    t3 = Text(Point(25, 38.5),
              "Market Capitalisation: The total market value of a
company's outstanding shares,\ncalculated as share price
multiplied by shares outstanding.")
    t4 = Text(Point(25, 35.5),
              "Income (TTM): Total net income generated by the
company over the trailing twelve months.")
    t5 = Text(Point(25, 33),
              "Revenue (TTM): Total income generated from
business operations over the trailing twelve months.")
    t6 = Text(Point(25, 30),
              "Book Value Per Share (MRQ): A company’s equity
available to common shareholders divided by\nthe number of
outstanding shares, based on the most recent quarter.")
    t7 = Text(Point(25, 27),
              "Dividend Yield (Annual): A company's annual
dividend payments expressed as a percentage of its stock
price.")
    t8 = Text(Point(25, 24),
              "Full-Time Employees: The total number of employees
working full-time for the company.")
    t9 = Text(Point(25, 21),
```

```
            "Analysts Mean Recommendation (1=Buy, 5=Sell): The
average recommendation from financial analysts,\nranging from 1
(strong buy) to 5 (strong sell).")
   t10 = Text(Point(25, 17),
            "Current Ratio (MRQ): A liquidity ratio that
measures a company's ability to pay short-term
obligations,\ncalculated as current assets divided by current
liabilities.")
   t11 = Text(Point(25, 13),
            "Total Debt to Equity (MRQ): A measure of
financial leverage,\ncalculated as total liabilities divided by
shareholders' equity.")
   t12 = Text(Point(25, 9),
            "Long-Term Debt to Equity (MRQ): A measure of
long-term financial leverage,\ncalculated as long-term
liabilities divided by shareholders' equity.")
   t13 = Text(Point(25, 6),
            "Annual EPS Growth Past 5 Years: The compound
annual growth rate of earnings per share over the past five
years.")
   t14 = Text(Point(25, 44),
            "Annual Sales Growth Past 5 Years: The compound
annual growth rate of\na company's revenue over the past five
years.")
   t15 = Text(Point(25, 40),
            "Return on Asset (TTM): A profitability metric
showing the percentage\nof profit a company earns relative to
its total assets.")
   t16 = Text(Point(25, 36),
            "Return on Equity (TTM): A measure of financial
performance,\ncalculated as net income divided by shareholders'
equity.")
   t17 = Text(Point(25, 32.5),
            "Return on Investment (TTM): A measure of the
profitability of an investment relative to its cost.")
   t18 = Text(Point(25, 28.5),
            "Operating Margin (TTM): A profitability ratio
showing what percentage of\nrevenue remains after covering
operating expenses.")
   t19 = Text(Point(25, 25),
            "Shares Outstanding: The total number of shares of
a company's stock currently held by all shareholders.")
```

```
    t20 = Text(Point(25, 22), "Volume: The number of shares
traded in a given time period.")
    t21 = Text(Point(25, 19), "Performance (Year): The percentage
change in a stock's price over the last year.")
    t22 = Text(Point(25, 16),
               "Beta: A measure of a stock's volatility relative
to the market. A beta of 1 indicates the stock moves with the
market.")
    t23 = Text(Point(25, 12),
               "Average True Range (14): A measure of market
volatility, calculated as the\naverage range between high and
low prices over 14 days.")
    t24 = Text(Point(25, 8.5),
               "Volatility (Month): A measure of the variation in
a stock's price over the past month.")
    t25 = Text(Point(25, 6),
               "Current Stock Price: The latest market price at
which a stock trades.")

    list = [t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11,
t12,t13]
    list2 = [t14, t15, t16, t17, t18, t19, t20, t21, t22, t23,
t24, t25]

    arrow_down = Image(Point(45, 25), "arrow_down.png")
    arrow_down.draw(win)

    for i in list:
        i.setSize(15)
        i.setTextColor("brown")
        i.draw(win)

    arrowx1 = 43.5
    arrowx2 = 46.5
    arrowy1 = 30
    arrowy2 = 20

    count = 1

    continuex1 = 15
    continuey1 = 4
    continuex2 = 35
```

```
   continuey2 = 1

   r1 = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
   r1.setFill(color_rgb(139,69, 19))
   r1.draw(win)

   nexttext = Text(Point(25, 5/2), "Next")
   nexttext.setSize(17)
   nexttext.setFace("arial")
   nexttext.setStyle("bold")
   nexttext.setTextColor("white")
   nexttext.draw(win)

   while True:
       click = win.getMouse()

       #If the user presses in these coordinates, he will quit
the program
       if quitx1 <= click.getX() <= quitx2:
           if quity2 <= click.getY() <= quity1:
               win.close()
               quit()

       #If the user press in the arrow, we will undraw the
variables that are shown and draw the new ones.
       if arrowx1 <= click.getX() <= arrowx2:
           if arrowy2 <= click.getY() <= arrowy1:
               if count == 1:
                   count = 2

                   for i in list:
                       i.undraw()

                   for i in list2:
                       i.setSize(15)
                       i.setTextColor("brown")
                       i.draw(win)

                   arrow_down.undraw()
                   arrow_up = Image(Point(45, 25),
"arrow_up.png")
```

```
                    arrow_up.draw(win)

                elif count == 2:
                    count = 1

                    for i in list2:
                        i.undraw()

                    for i in list:
                        i.setSize(15)
                        i.setTextColor("brown")
                        i.draw(win)

                    arrow_up.undraw()
                    arrow_down = Image(Point(45, 25),
"arrow_down.png")
                    arrow_down.draw(win)

        elif continuex1 <= click.getX() <= continuex2:
            if continuey2 <= click.getY() <= continuey1:
                break

    if count == 1:
        for i in list:
            i.undraw()

        arrow_down.undraw()

    elif count == 2:
        for i in list2:
            i.undraw()

        arrow_up.undraw()

    r1.undraw()
    nexttext.undraw()
    title.undraw()

def num_of_var(win, quitx1, quitx2, quity1, quity2):
    #We establish the GUI interface
    t1 = Text(Point(25,36.5), 'How many extra variables\ndo you
want to use?\n\n'
```

```
                          'Note that the variable\n"Sector"\nis
a predefined one.')

   t1.setSize(17)
   t1.setTextColor("brown")
   t1.draw(win)

   x1= 21.5
   x2 = 28.5

   var0y1 = 25
   var0y2 = 20

   var1y1 = 17
   var1y2 = 12

   var2y1 = 9
   var2y2 = 4

   extra0 = Text(Point(25,22.5), "0")
   extra1 = Text(Point(25,14.5), "1")
   extra2 = Text(Point(25,6.5), "2")

   var0 = Rectangle(Point(x1, var0y1), Point(x2, var0y2))
   var1 = Rectangle(Point(x1,var1y1), Point(x2,var1y2))
   var2 = Rectangle(Point(x1,var2y1), Point(x2,var2y2))

   var = [var0,var1,var2]

   for i in var:
       i.setFill(color_rgb(139,69, 19))
       i.draw(win)

   t = [extra0,extra1,extra2]

   for i in t:
       i.setFace("arial")
       i.setStyle("bold")
       i.setSize(17)
       i.setTextColor("white")
       i.draw(win)
```

```
    while True:
        click = win.getMouse()

        #Depending on the coordinates of the click a different
number of extra variables will be chosen.
        if x1 <= click.getX() <= x2:
            if var0y2 <= click.getY() <= var0y1:
                extra_variable_number = 0
                break

            elif var1y2 <= click.getY() <= var1y1:
                extra_variable_number = 1
                break

            elif var2y2 <= click.getY() <= var2y1:
                extra_variable_number = 2
                break

        elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    t1.undraw()

    for i in t:
        i.undraw()

    for i in var:
        i.undraw()

    return extra_variable_number

def var_selector(win, quitx1, quitx2, quity1, quity2,
extra_variable_number, df):
    #We create an empty list with the chosen variables
    chosen_var = []
    #We put into a list the columns of the data set
    columns = df.columns.tolist()
    #We establish sector as a predefined variable
    chosen_var.append(columns[1])
```

```
    if extra_variable_number == 0:
        return chosen_var

    #We establish some coordinates for the GUI and we do the GUI
adjustments
    x1, x2, x3, x4, x5, x6, x7, x8 = 1, 12.25, 13.25, 24.5, 25.5,
36.75, 37.75, 49
    x3_1, x3_2, x3_3, x3_4, x3_5, x3_6 = 1, 16.33, 17.33, 32.66,
33.66, 49
    y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12 = 43,
36.83, 35.83, 29.66, 28.66, 22.49, 21.49, 15.32, 14.32, 8.15,
7.15, 1

    var1 = Rectangle(Point(x1,y1), Point(x2,y2))
    var2 = Rectangle(Point(x3,y1), Point(x4,y2))
    var3 = Rectangle(Point(x5,y1), Point(x6,y2))
    var4 = Rectangle(Point(x7,y1), Point(x8,y2))
    var5 = Rectangle(Point(x1,y3), Point(x2,y4))
    var6 = Rectangle(Point(x3,y3), Point(x4,y4))
    var7 = Rectangle(Point(x5,y3), Point(x6,y4))
    var8 = Rectangle(Point(x7,y3), Point(x8,y4))
    var9 = Rectangle(Point(x1,y5), Point(x2,y6))
    var10 = Rectangle(Point(x3,y5), Point(x4,y6))
    var11 = Rectangle(Point(x5,y5), Point(x6,y6))
    var12 = Rectangle(Point(x7,y5), Point(x8,y6))
    var13 = Rectangle(Point(x1,y7), Point(x2,y8))
    var14 = Rectangle(Point(x3,y7), Point(x4,y8))
    var15 = Rectangle(Point(x5,y7), Point(x6,y8))
    var16 = Rectangle(Point(x7,y7), Point(x8,y8))
    var17 = Rectangle(Point(x1,y9), Point(x2,y10))
    var18 = Rectangle(Point(x3,y9), Point(x4,y10))
    var19 = Rectangle(Point(x5,y9), Point(x6,y10))
    var20 = Rectangle(Point(x7,y9), Point(x8,y10))
    var21 = Rectangle(Point(x3_1,y11), Point(x3_2,y12))
    var22 = Rectangle(Point(x3_3,y11), Point(x3_4,y12))
    var23 = Rectangle(Point(x3_5,y11), Point(x3_6,y12))

    var = [var1, var2, var3, var4, var5, var6, var7, var8, var9,
var10, var11,var12,
           var13, var14, var15, var16, var17, var18, var19,
var20, var21, var22, var23]
```

```
    for i in var:
        i.setFill(color_rgb(139, 69, 19))
        i.draw(win)

    t1 = Text(Point((x1+x2)/2, (y1+y2)/2),
"Market\ncapitalisation")
    t2 = Text(Point((x3+x4)/2, (y1+y2)/2), "Income")
    t3 = Text(Point((x5+x6)/2, (y1+y2)/2), "Revenue")
    t4 = Text(Point((x7+x8)/2, (y1+y2)/2), "Book value\nper
share")
    t5 = Text(Point((x1+x2)/2, (y3+y4)/2), "Dividend\nyield")
    t6 = Text(Point((x3+x4)/2, (y3+y4)/2), "Full
time\nemployees")
    t7 = Text(Point((x5+x6)/2, (y3+y4)/2),
"Analysts'\nmean\nrecomendation")
    t8 = Text(Point((x7+x8)/2, (y3+y4)/2), "Current\nratio")
    t9 = Text(Point((x1+x2)/2, (y5+y6)/2), "Total debt\nto
equity")
    t10 = Text(Point((x3+x4)/2, (y5+y6)/2), "Long term\ndebt
to\nequity")
    t11 = Text(Point((x5+x6)/2, (y5+y6)/2), "Annual EPS\ngrowth
past\n5 years")
    t12 = Text(Point((x7+x8)/2, (y5+y6)/2), "Annual sales\ngrowth
past\n5 years")
    t13 = Text(Point((x1+x2)/2, (y7+y8)/2), "Return on\nasset")
    t14 = Text(Point((x3+x4)/2, (y7+y8)/2), "Return on\nequity")
    t15 = Text(Point((x5+x6)/2, (y7+y8)/2), "Return
on\ninvestment")
    t16 = Text(Point((x7+x8)/2, (y7+y8)/2), "Operating\nmargin")
    t17 = Text(Point((x1+x2)/2, (y9+y10)/2),
"Shares\noutstanding")
    t18 = Text(Point((x3+x4)/2, (y9+y10)/2), "Volume")
    t19 = Text(Point((x5+x6)/2, (y9+y10)/2), "Performance")
    t20 = Text(Point((x7+x8)/2, (y9+y10)/2), "Beta")
    t21 = Text(Point((x3_1+x3_2)/2, (y11+y12)/2), "Average\nTrue
Range")
    t22 = Text(Point((x3_3+x3_4)/2, (y11+y12)/2), "Volatility")
    t23 = Text(Point((x3_5+x3_6)/2, (y11+y12)/2), "Current\nStock
Price")
```

```
   t =
[t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t16,t17,t18,
t19,t20,t21,t22,t23]

   for i in t:
       i.setFace("arial")
       i.setStyle("bold")
       i.setSize(11)
       i.setTextColor("white")
       i.draw(win)

   if extra_variable_number == 1:
       #If the user wants only one extra variable, we enter the
following code
       text = Text(Point(25,46), 'Choose 1 extra variable')
       text.setSize(17)
       text.setTextColor("brown")
       text.draw(win)

       while True:
           click = win.getMouse()

           #Depending on the position of the click we will
choose a variable or another
           if y2 <= click.getY() <= y1:
               if x1 <= click.getX() <= x2:
                   chosen_var.append(columns[2])
                   break

               elif x3 <= click.getX() <= x4:
                   chosen_var.append(columns[3])
                   break

               elif x5 <= click.getX() <= x6:
                   chosen_var.append(columns[4])
                   break

               elif x7 <= click.getX() <= x8:
                   chosen_var.append(columns[5])
                   break

           elif y4 <= click.getY() <= y3:
```

```
            if x1 <= click.getX() <= x2:
                chosen_var.append(columns[6])
                break

            elif x3 <= click.getX() <= x4:
                chosen_var.append(columns[7])
                break

            elif x5 <= click.getX() <= x6:
                chosen_var.append(columns[8])
                break

            elif x7 <= click.getX() <= x8:
                chosen_var.append(columns[9])
                break

        elif y6 <= click.getY() <= y5:
            if x1 <= click.getX() <= x2:
                chosen_var.append(columns[10])
                break

            elif x3 <= click.getX() <= x4:
                chosen_var.append(columns[11])
                break

            elif x5 <= click.getX() <= x6:
                chosen_var.append(columns[12])
                break

            elif x7 <= click.getX() <= x8:
                chosen_var.append(columns[13])
                break

        elif y8 <= click.getY() <= y7:
            if x1 <= click.getX() <= x2:
                chosen_var.append(columns[14])
                break

            elif x3 <= click.getX() <= x4:
                chosen_var.append(columns[15])
                break
```

```
                elif x5 <= click.getX() <= x6:
                    chosen_var.append(columns[16])
                    break

                elif x7 <= click.getX() <= x8:
                    chosen_var.append(columns[17])
                    break

            elif y10 <= click.getY() <= y9:
                if x1 <= click.getX() <= x2:
                    chosen_var.append(columns[18])
                    break

                elif x3 <= click.getX() <= x4:
                    chosen_var.append(columns[19])
                    break

                elif x5 <= click.getX() <= x6:
                    chosen_var.append(columns[20])
                    break

                elif x7 <= click.getX() <= x8:
                    chosen_var.append(columns[21])
                    break

            elif y12 <= click.getY() <= y11:
                if x3_1 <= click.getX() <= x3_2:
                    chosen_var.append(columns[22])
                    break

                elif x3_3 <= click.getX() <= x3_4:
                    chosen_var.append(columns[23])
                    break

                elif x3_5 <= click.getX() <= x3_6:
                    chosen_var.append(columns[24])
                    break


            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
```

```
                quit()

        return chosen_var

    elif extra_variable_number == 2:
        #Here we will do exactly the same as with one variable
but undrawing the variable button when it is
        #clicked and including an error handling method so that
the user cannot select twice the same variable.
        text = Text(Point(25,46), 'Choose 2 extra variables')
        text.setSize(17)
        text.setTextColor("brown")
        text.draw(win)

        while len(chosen_var) < 3:
            click = win.getMouse()

            if y2 <= click.getY() <= y1:
                if x1 <= click.getX() <= x2 and columns[2] not in
chosen_var:
                    chosen_var.append(columns[2])
                    var1.undraw()
                    t1.undraw()
                elif x3 <= click.getX() <= x4 and columns[3] not
in chosen_var:
                    chosen_var.append(columns[3])
                    var2.undraw()
                    t2.undraw()
                elif x5 <= click.getX() <= x6 and columns[4] not
in chosen_var:
                    chosen_var.append(columns[4])
                    var3.undraw()
                    t3.undraw()
                elif x7 <= click.getX() <= x8 and columns[5] not
in chosen_var:
                    chosen_var.append(columns[5])
                    var4.undraw()
                    t4.undraw()

            elif y4 <= click.getY() <= y3:
                if x1 <= click.getX() <= x2 and columns[6] not in
chosen_var:
```

```
                    chosen_var.append(columns[6])
                    var5.undraw()
                    t5.undraw()
                elif x3 <= click.getX() <= x4 and columns[7] not
in chosen_var:
                    chosen_var.append(columns[7])
                    var6.undraw()
                    t6.undraw()
                elif x5 <= click.getX() <= x6 and columns[8] not
in chosen_var:
                    chosen_var.append(columns[8])
                    var7.undraw()
                    t7.undraw()
                elif x7 <= click.getX() <= x8 and columns[9] not
in chosen_var:
                    chosen_var.append(columns[9])
                    var8.undraw()
                    t8.undraw()

            elif y6 <= click.getY() <= y5:
                if x1 <= click.getX() <= x2 and columns[10] not
in chosen_var:
                    chosen_var.append(columns[10])
                    var9.undraw()
                    t9.undraw()
                elif x3 <= click.getX() <= x4 and columns[11] not
in chosen_var:
                    chosen_var.append(columns[11])
                    var10.undraw()
                    t10.undraw()
                elif x5 <= click.getX() <= x6 and columns[12] not
in chosen_var:
                    chosen_var.append(columns[12])
                    var11.undraw()
                    t11.undraw()
                elif x7 <= click.getX() <= x8 and columns[13] not
in chosen_var:
                    chosen_var.append(columns[13])
                    var12.undraw()
                    t12.undraw()

            elif y8 <= click.getY() <= y7:
```

```
                if x1 <= click.getX() <= x2 and columns[14] not
in chosen_var:
                    chosen_var.append(columns[14])
                    var13.undraw()
                    t13.undraw()
                elif x3 <= click.getX() <= x4 and columns[15] not
in chosen_var:
                    chosen_var.append(columns[15])
                    var14.undraw()
                    t14.undraw()
                elif x5 <= click.getX() <= x6 and columns[16] not
in chosen_var:
                    chosen_var.append(columns[16])
                    var15.undraw()
                    t15.undraw()
                elif x7 <= click.getX() <= x8 and columns[17] not
in chosen_var:
                    chosen_var.append(columns[17])
                    var16.undraw()
                    t16.undraw()

            elif y10 <= click.getY() <= y9:
                if x1 <= click.getX() <= x2 and columns[18] not
in chosen_var:
                    chosen_var.append(columns[18])
                    var17.undraw()
                    t17.undraw()
                elif x3 <= click.getX() <= x4 and columns[19] not
in chosen_var:
                    chosen_var.append(columns[19])
                    var18.undraw()
                    t18.undraw()
                elif x5 <= click.getX() <= x6 and columns[20] not
in chosen_var:
                    chosen_var.append(columns[20])
                    var19.undraw()
                    t19.undraw()
                elif x7 <= click.getX() <= x8 and columns[21] not
in chosen_var:
                    chosen_var.append(columns[21])
                    var20.undraw()
                    t20.undraw()
```

```
            elif y12 <= click.getY() <= y11:
                if x3_1 <= click.getX() <= x3_2 and columns[22]
not in chosen_var:
                    chosen_var.append(columns[22])
                    var21.undraw()
                    t21.undraw()
                elif x3_3 <= click.getX() <= x3_4 and columns[23]
not in chosen_var:
                    chosen_var.append(columns[23])
                    var22.undraw()
                    t22.undraw()
                elif x3_5 <= click.getX() <= x3_6 and columns[24]
not in chosen_var:
                    chosen_var.append(columns[24])
                    var23.undraw()
                    t23.undraw()

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()


        return chosen_var

def graph_selector(win, quitx1, quitx2, quity1, quity2,
extra_variable_number):
    #Because of technical factors, we have decided to put the
background image
    #to simulate a new window without having to close it and
opening a new one.

    background = Image(Point(25, 25), "background_image.png")
    background.draw(win)

    qt = Rectangle(Point(quitx1, quity1), Point(quitx2, quity2))
    qt.setFill(color_rgb(139, 69, 19))
    qt.draw(win)

    quittext = Text(Point(44.5, 47), "Quit")
    quittext.setFace("arial")
```

```
   quittext.setStyle("bold")
   quittext.setTextColor("white")
   quittext.setSize(17)
   quittext.draw(win)

   #Here, depending on the number of extra variables, we can
choose some graphs or others
   if extra_variable_number == 0:
       text = Text(Point(25, 35), 'Please select the graph\nthat
you would like to view')
       text.setTextColor("brown")
       text.setSize(17)
       text.draw(win)

       x1 = 15
       x2 = 35
       r1y1 = 25
       r1y2 = 20

       r1 = Rectangle(Point(x1,r1y1), Point(x2,r1y2))

       r2y1 = 15
       r2y2 = 10

       r2 = Rectangle(Point(x1,r2y1), Point(x2,r2y2))

       t1 = Text(Point((x1+x2)/2, (r1y1+r1y2)/2), "Pie Chart")
       t2 = Text(Point((x1+x2)/2, (r2y1+r2y2)/2), "Bar Chart")

       r = [r1,r2]

       for i in r:
           i.setFill(color_rgb(139,69, 19))
           i.draw(win)

       t = [t1, t2]

       for i in t:
           i.setFace("arial")
           i.setStyle("bold")
           i.setSize(17)
           i.setTextColor("white")
```

```
            i.draw(win)

        while True:
            click = win.getMouse()

            if x1 <= click.getX() <= x2:
                if r1y2 <= click.getY() <= r1y1:
                    chosen_graph = "Pie Chart"
                    break

                elif r2y2 <= click.getY() <= r2y1:
                    chosen_graph = "Bar Plot"
                    break

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        text.undraw()
        r1.undraw()
        r2.undraw()
        t1.undraw()
        t2.undraw()

        return chosen_graph

    elif extra_variable_number == 1:
        text = Text(Point(25, 45), 'Please select the graph\nthat
you would like to view')
        text.setSize(17)
        text.setTextColor("brown")
        text.draw(win)

        l1x1 = 1
        l1x2 = 24.5
        l2x1 = 25.5
        l2x2 = 49

        y1 = 40
        y2 = 31
        y3 = 30
```

```
    y4 = 21
    y5 = 20
    y6 = 11
    y7 = 10
    y8 = 1

    r1 = Rectangle(Point(l1x1, y1), Point(l1x2, y2))
    t1 = Text(Point((l1x1+l1x2)/2,(y1+y2)/2), "Line Plot")
    r2 = Rectangle(Point(l2x1, y1), Point(l2x2, y2))
    t2 = Text(Point((l2x1+l2x2)/2,(y1+y2)/2), "Box Plot")
    r3 = Rectangle(Point(l1x1, y3), Point(l1x2, y4))
    t3 = Text(Point((l1x1+l1x2)/2,(y3+y4)/2), "Histogram")
    r4 = Rectangle(Point(l2x1, y3), Point(l2x2, y4))
    t4 = Text(Point((l2x1+l2x2)/2,(y3+y4)/2), "Violin Plot")
    r5 = Rectangle(Point(l1x1, y5), Point(l1x2, y6))
    t5 = Text(Point((l1x1+l1x2)/2,(y5+y6)/2), "Error Plot")
    r6 = Rectangle(Point(l2x1, y5), Point(l2x2, y6))
    t6 = Text(Point((l2x1+l2x2)/2,(y5+y6)/2), "Bar Plot")
    r7 = Rectangle(Point(l1x1, y7), Point(l2x2, y8))
    t7 = Text(Point((l1x1+l2x2)/2,(y7+y8)/2), "Scatterplot")

    r = [r1, r2, r3, r4, r5, r6, r7]

    for i in r:
        i.setFill(color_rgb(139,69, 19))
        i.draw(win)

    t = [t1, t2, t3, t4, t5, t6, t7]

    for i in t:
        i.setFace("arial")
        i.setStyle("bold")
        i.setSize(17)
        i.setTextColor("white")
        i.draw(win)

    while True:
        click = win.getMouse()
        x, y = click.getX(), click.getY()

        if l1x1 <= x <= l1x2 and y2 <= y <= y1:
            chosen_graph = "Line Plot"
```

```
            break

        elif l2x1 <= x <= l2x2 and y2 <= y <= y1:
            chosen_graph = "Box Plot"
            break

        elif l1x1 <= x <= l1x2 and y4 <= y <= y3:
            chosen_graph = "Histogram"
            break

        elif l2x1 <= x <= l2x2 and y4 <= y <= y3:
            chosen_graph = "Violin Plot"
            break

        elif l1x1 <= x <= l1x2 and y6 <= y <= y5:
            chosen_graph = "Error Plot"
            break

        elif l2x1 <= x <= l2x2 and y6 <= y <= y5:
            chosen_graph = "Bar Plot"
            break

        elif l1x1 <= x <= l2x2 and y8 <= y <= y7:
            chosen_graph = "Scatterplot"
            break

        elif quitx1 <= x <= quitx2 and quity2 <= y <= quity1:
            win.close()
            quit()

    text.undraw()

    for i in r:
        i.undraw()

    for i in t:
        i.undraw()

    return chosen_graph

else:
```

```
        text = Text(Point(25, 35), 'Please select the graph\nthat
you would like to view')
        text.setSize(17)
        text.setTextColor("brown")
        text.draw(win)

        x1 = 15
        x2 = 35
        r1y1 = 25
        r1y2 = 20

        r1 = Rectangle(Point(x1, r1y1), Point(x2, r1y2))

        r2y1 = 15
        r2y2 = 10

        r2 = Rectangle(Point(x1, r2y1), Point(x2, r2y2))

        t1 = Text(Point((x1 + x2) / 2, (r1y1 + r1y2) / 2), "Line
Plot")
        t2 = Text(Point((x1 + x2) / 2, (r2y1 + r2y2) / 2),
"Scatterplot")

        r = [r1, r2]

        for i in r:
            i.setFill(color_rgb(139,69, 19))
            i.draw(win)

        t = [t1, t2]

        for i in t:
            i.setFace("arial")
            i.setStyle("bold")
            i.setSize(17)
            i.setTextColor("white")
            i.draw(win)

        while True:
            click = win.getMouse()

            if x1 <= click.getX() <= x2:
```

```
                if r1y2 <= click.getY() <= r1y1:
                    chosen_graph = "Line Plot"
                    break

                elif r2y2 <= click.getY() <= r2y1:
                    chosen_graph = "Scatterplot"
                    break

            elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        text.undraw()
        r1.undraw()
        r2.undraw()
        t1.undraw()
        t2.undraw()

        return chosen_graph

def graph_editor(win, chosen_graph, quitx1, quitx2, quity1,
quity2):
    #This function allows to edit the graph at the same time that
we are visualising it
    g1 = Text(Point(3, 35), 'Title')
    g1.setSize(17)
    g1.setTextColor("brown")
    g1.draw(win)
    e1 = Entry(Point(7, 35), 15)
    e1.draw(win)
    e1.setText(chosen_graph)

    g2 = Text(Point(3, 30), 'X label')
    g2.setSize(17)
    g2.setTextColor("brown")
    g2.draw(win)
    e2 = Entry(Point(7, 30), 15)
    e2.draw(win)
    e2.setText("x-label")

    g3 = Text(Point(3, 25), 'Y label')
```

```
   g3.setSize(17)
   g3.setTextColor("brown")
   g3.draw(win)

   e3 = Entry(Point(7, 25), 15)
   e3.draw(win)
   e3.setText("y-label")

   continuex1 = 2
   continuey1 = 20
   continuex2 = 10
   continuey2 = 17

   r1 = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
   r1.setFill(color_rgb(139,69, 19))
   r1.draw(win)

   nexttext = Text(Point(6, 18.5), "Apply")
   nexttext.setSize(17)
   nexttext.setFace("arial")
   nexttext.setStyle("bold")
   nexttext.setTextColor("white")
   nexttext.draw(win)

   while True:
       click = win.getMouse()

       if continuex1 <= click.getX() <= continuex2:
           if continuey2 <= click.getY() <= continuey1:
               break

       elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
           win.close()
           quit()

   title = e1.getText()
   x_label = e2.getText()
   y_label = e3.getText()

   g1.undraw()
```

```
    g2.undraw()
    g3.undraw()
    e1.undraw()
    e2.undraw()
    e3.undraw()
    nexttext.undraw()
    r1.undraw()

    return title, x_label, y_label

def graph_editor_pie(win, quitx1, quitx2, quity1, quity2):
    #For the pie chart we have a specific function to edit it as
we are visualising it
    #because it does not have x-label and y-label, only title.
    g1 = Text(Point(3, 25), 'Title')
    g1.setSize(17)
    g1.setTextColor("brown")
    g1.draw(win)
    e1 = Entry(Point(7, 25), 15)
    e1.draw(win)
    e1.setText("Pie Chart")

    continuex1 = 2
    continuey1 = 20
    continuex2 = 10
    continuey2 = 17

    r1 = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
    r1.setFill(color_rgb(139,69, 19))
    r1.draw(win)

    nexttext = Text(Point(6, 18.5), "Apply")
    nexttext.setSize(17)
    nexttext.setFace("arial")
    nexttext.setStyle("bold")
    nexttext.setTextColor("white")
    nexttext.draw(win)

    while True:
        click = win.getMouse()
```

```
        if continuex1 <= click.getX() <= continuex2:
            if continuey2 <= click.getY() <= continuey1:
                break

        elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    title = e1.getText()

    g1.undraw()
    e1.undraw()
    nexttext.undraw()
    r1.undraw()

    return title

def graph_editor_histogram(win, chosen_graph, quitx1, quitx2,
quity1, quity2):
    # For the histogram we have a specific function to edit it as
we are visualising it
    # because it does not have a title, only x-label and y-label.
    g2 = Text(Point(3, 30), 'X label')
    g2.setSize(17)
    g2.setTextColor("brown")
    g2.draw(win)

    e2 = Entry(Point(7, 30), 15)
    e2.draw(win)
    e2.setText("x-label")

    g3 = Text(Point(3, 25), 'Y label')
    g3.setSize(17)
    g3.setTextColor("brown")
    g3.draw(win)

    e3 = Entry(Point(7, 25), 15)
    e3.draw(win)
    e3.setText("y-label")

    continuex1 = 2
```

```
    continuey1 = 20
    continuex2 = 10
    continuey2 = 17

    r1 = Rectangle(Point(continuex1, continuey1),
Point(continuex2, continuey2))
    r1.setFill(color_rgb(139,69, 19))
    r1.draw(win)

    nexttext = Text(Point(6, 18.5), "Apply")
    nexttext.setSize(17)
    nexttext.setFace("arial")
    nexttext.setStyle("bold")
    nexttext.setTextColor("white")
    nexttext.draw(win)

    while True:
        click = win.getMouse()

        if continuex1 <= click.getX() <= continuex2:
            if continuey2 <= click.getY() <= continuey1:
                break

        elif quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    x_label = e2.getText()
    y_label = e3.getText()

    g2.undraw()
    g3.undraw()
    e2.undraw()
    e3.undraw()
    nexttext.undraw()
    r1.undraw()

    return x_label, y_label

def lineplot(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
```

```
   #Here we are creating the Line Plot
   text = Text(Point(25, 46), 'Line Plot')
   text.setSize(17)
   text.setTextColor("brown")
   text.draw(win)

   sect = Rectangle(Point(37, 35), Point(47, 9))
   sect.setFill(color_rgb(139,69, 19))
   sect.draw(win)

   #Given that the x-axis is labeled with numbers, here we will
write in the window the relationships
   #between the numbers and the sectors.
   mat = Text(Point(42, 33), "1-Basic Materials")
   mat.setSize(17)
   mat.setFace("arial")
   mat.setTextColor("white")
   mat.draw(win)
   co = Text(Point(42, 31), "2-Communication Services")
   co.setSize(17)
   co.setFace("arial")
   co.setTextColor("white")
   co.draw(win)
   cyc = Text(Point(42, 29), "3-Consumer Cyclical")
   cyc.setSize(17)
   cyc.setFace("arial")
   cyc.setTextColor("white")
   cyc.draw(win)
   cde = Text(Point(42, 27), "4-Consumer Defensive")
   cde.setSize(17)
   cde.setFace("arial")
   cde.setTextColor("white")
   cde.draw(win)
   cdi = Text(Point(42, 25), "5-Consumer Discretionary")
   cdi.setSize(17)
   cdi.setFace("arial")
   cdi.setTextColor("white")
   cdi.draw(win)
   en = Text(Point(42, 23), "6-Energy")
   en.setSize(17)
   en.setFace("arial")
   en.setTextColor("white")
```

```
    en.draw(win)
    fin = Text(Point(42, 21), "7-Financial Services")
    fin.setSize(17)
    fin.setFace("arial")
    fin.setTextColor("white")
    fin.draw(win)
    hel = Text(Point(42, 19), "8-Healthcare")
    hel.setSize(17)
    hel.setFace("arial")
    hel.setTextColor("white")
    hel.draw(win)
    ind = Text(Point(42, 17), "9-Industrial")
    ind.setSize(17)
    ind.setFace("arial")
    ind.setTextColor("white")
    ind.draw(win)
    ret = Text(Point(42, 15), "10-Real Estate")
    ret.setSize(17)
    ret.setFace("arial")
    ret.setTextColor("white")
    ret.draw(win)
    tech = Text(Point(42, 13), "11-Technology")
    tech.setSize(17)
    tech.setFace("arial")
    tech.setTextColor("white")
    tech.draw(win)
    uti = Text(Point(42, 11), "12-Utilities")
    uti.setSize(17)
    uti.setFace("arial")
    uti.setTextColor("white")
    uti.draw(win)

    #If the total number of variables is 2, we will show a line
plot
    if len(chosen_var) == 2:
        tempvar = df.groupby(chosen_var[0])[chosen_var[1]].mean()
        plt.plot(tempvar,"o-r")
        plt.title(title)
        plt.xlabel(x_label)

plt.xticks(ticks=range(len(tempvar)),labels=[1,2,3,4,5,6,7,8,9,1
0,11,12])
```

```
        plt.ylabel(y_label)
        plt.ioff()
        plt.savefig("graph.png", format='png')
        plt.close()

        graph = Image(Point(25, 23), "graph.png")
        graph.draw(win)

        while True and count==1:
            click = win.getMouse()

            if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        count=1

        return count

    #If the number of total variables is 3, we will make a line
plot with 2 lines
    elif len(chosen_var) == 3:
        tempvar1 =
df.groupby(chosen_var[0])[chosen_var[1]].mean()
        tempvar2 =
df.groupby(chosen_var[0])[chosen_var[2]].mean()
        plt.plot(tempvar1,"o-b")
        plt.plot(tempvar2,"o--r")
        plt.title(title)
        plt.xlabel(x_label)
        plt.xticks(ticks=range(len(tempvar1)), labels=[1, 2, 3,
4, 5, 6, 7, 8, 9, 10, 11, 12])
        plt.ylabel(y_label)
        plt.legend([chosen_var[1], chosen_var[2]],loc=0)
        plt.ioff()
        plt.savefig("graph.png", format='png')
        plt.close()

        graph = Image(Point(25, 23), "graph.png")
        graph.draw(win)
```

```
        #Here the count = 1 is a flag. To enter this part of the
code only at a specific time of the program
        while True and count == 1:
            click = win.getMouse()

            if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        count = 1

        return count

def boxplot(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
    #Here we are going to make the boxplot
    text = Text(Point(25, 47), 'Box Plot')
    text.setSize(17)
    text.setTextColor("brown")
    text.draw(win)

    plt.figure(figsize=(12, 6))

    # Group the data by the chosen category (e.g., "Sector") and
retrieve the values for the chosen variable
    data = [df[df[chosen_var[0]] ==
category][chosen_var[1]].dropna() for category in
df[chosen_var[0]].unique()]
    labels = df[chosen_var[0]].unique()

    plt.boxplot(data, labels=labels)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.title(title)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.ioff()
    plt.savefig("graph.png", format='png')
    plt.close()

    #Here we are resizing the graph
```

```
    with im(filename="graph.png") as img:
        img.resize(840, 660)
        img.save(filename="graph.png")

    graph = Image(Point(25, 23), "graph.png")
    graph.draw(win)

    #Here the count = 1 is a flag. To enter this part of the code
only at a specific time of the program
    while True and count == 1:
        click = win.getMouse()

        if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    count = 1

    return count

def histogram(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
    #Here we are creating the histogram
    text = Text(Point(25, 47), 'Histogram')
    text.setSize(17)
    text.setTextColor("brown")
    text.draw(win)

    bins = 10
    sectors = df[chosen_var[0]].unique()  # Get unique sectors
    num_sectors = len(sectors)
    rows = (num_sectors // 3) + (num_sectors % 3 > 0)  #
Calculate rows needed for a 3-column grid

    plt.figure(figsize=(15, 5 * rows))

    for i, sector in enumerate(sectors, start=1):
        plt.subplot(rows, 3, i)  # Arrange plots in a grid of 3
columns
        sector_data = df[df[chosen_var[0]] ==
sector][chosen_var[1]].dropna()
```

```
        plt.hist(sector_data, bins=bins, alpha=0.7,
color='skyblue', edgecolor='black')
        plt.title(f'{sector}')
        plt.xlabel(x_label)
        plt.ylabel(y_label)

    plt.tight_layout()
    plt.suptitle(title, y=1.02, fontsize=16)

    plt.ioff()
    plt.savefig("graph.png", format='png')
    plt.close()

    #Here we are resizing the graph
    with im(filename="graph.png") as img:
        img.resize(850,660)
        img.save(filename="graph.png")

    graph = Image(Point(25, 23), "graph.png")
    graph.draw(win)

    while True and count == 1:
        click = win.getMouse()

        #Here the count = 1 is a flag. To enter this part of the
code only at a specific time of the program
        if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    count = 1

    return count

def piechart(chosen_var, df, title, win, quitx1, quitx2, quity1,
quity2,count):
    #Here we are creating the pie chart
    text = Text(Point(25, 46), 'Pie Chart')
    text.setSize(17)
    text.setTextColor("brown")
    text.draw(win)
```

```
    tempvar = pd.value_counts(df[chosen_var[0]])
    labels = tempvar.index
    plt.figure(figsize=(8,8))
    plt.pie(tempvar, autopct='%1.1f%%', labels=labels)
    plt.title(title)

    plt.ioff()
    plt.savefig("graph.png", format='png')
    plt.close()

    #Here we are resizing the graph
    with im(filename="graph.png") as img:
        img.resize(840, 630)
        img.save(filename="graph.png")

    graph = Image(Point(25, 23), "graph.png")
    graph.draw(win)

    #Here the count = 1 is a flag. To enter this part of the code
only at a specific time of the program
    while True and count == 1:
        click = win.getMouse()

        if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    count = 1

    return count

def violinplot(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
    #We create the violin plot
    text = Text(Point(25, 47), 'Violin Plot')
    text.setSize(17)
    text.setTextColor("brown")
    text.draw(win)

    plt.figure(figsize=(12, 6))
```

```
    categories = df[chosen_var[0]].dropna().unique()
    data = [df[df[chosen_var[0]] ==
category][chosen_var[1]].dropna() for category in categories]

    data = [d for d in data if len(d) > 0]
    categories = [cat for cat, d in zip(categories, data) if
len(d) > 0]

    plt.violinplot(data, showmedians=True)
    plt.xticks(ticks=range(1, len(categories) + 1),
labels=categories, rotation=45)
    plt.title(title)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.ioff()
    plt.savefig("graph.png", format='png')
    plt.close()

    #We resize the grpah
    with im(filename="graph.png") as img:
        img.resize(840, 660)
        img.save(filename="graph.png")

    graph = Image(Point(25, 23), "graph.png")
    graph.draw(win)

    #Here the count = 1 is a flag. To enter this part of the code
only at a specific time of the program
    while True and count == 1:
        click = win.getMouse()

        if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    count = 1

    return count

def errorplot(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
```

```
   #We create the error plot
   text = Text(Point(25, 46), 'Error Plot')
   text.setSize(17)
   text.setTextColor("brown")
   text.draw(win)

   sect = Rectangle(Point(37, 35), Point(47, 9))
   sect.setFill(color_rgb(139,69, 19))
   sect.draw(win)

   #We create a legend to establish relationships between the
numbers of the x-label and the sectors.
   mat = Text(Point(42, 33), "1-Basic Materials")
   mat.setSize(17)
   mat.setFace("arial")
   mat.setTextColor("white")
   mat.draw(win)
   co = Text(Point(42, 31), "2-Communication Services")
   co.setSize(17)
   co.setFace("arial")
   co.setTextColor("white")
   co.draw(win)
   cyc = Text(Point(42, 29), "3-Consumer Cyclical")
   cyc.setSize(17)
   cyc.setFace("arial")
   cyc.setTextColor("white")
   cyc.draw(win)
   cde = Text(Point(42, 27), "4-Consumer Defensive")
   cde.setSize(17)
   cde.setFace("arial")
   cde.setTextColor("white")
   cde.draw(win)
   cdi = Text(Point(42, 25), "5-Consumer Discretionary")
   cdi.setSize(17)
   cdi.setFace("arial")
   cdi.setTextColor("white")
   cdi.draw(win)
   en = Text(Point(42, 23), "6-Energy")
   en.setSize(17)
   en.setFace("arial")
   en.setTextColor("white")
   en.draw(win)
```

```
   fin = Text(Point(42, 21), "7-Financial Services")
   fin.setSize(17)
   fin.setFace("arial")
   fin.setTextColor("white")
   fin.draw(win)
   hel = Text(Point(42, 19), "8-Healthcare")
   hel.setSize(17)
   hel.setFace("arial")
   hel.setTextColor("white")
   hel.draw(win)
   ind = Text(Point(42, 17), "9-Industrial")
   ind.setSize(17)
   ind.setFace("arial")
   ind.setTextColor("white")
   ind.draw(win)
   ret = Text(Point(42, 15), "10-Real Estate")
   ret.setSize(17)
   ret.setFace("arial")
   ret.setTextColor("white")
   ret.draw(win)
   tech = Text(Point(42, 13), "11-Technology")
   tech.setSize(17)
   tech.setFace("arial")
   tech.setTextColor("white")
   tech.draw(win)
   uti = Text(Point(42, 11), "12-Utilities")
   uti.setSize(17)
   uti.setFace("arial")
   uti.setTextColor("white")
   uti.draw(win)

   tempvar = df.groupby(chosen_var[0])[chosen_var[1]].sum()
   x = tempvar.index.tolist()
   y = tempvar.values.tolist()
   y_errors = np.std(y)

   plt.errorbar(x,y,yerr=y_errors, fmt='o', capsize = 5,
capthick=1, ecolor='red')
   plt.title(title)
   plt.xlabel(x_label)
   plt.xticks(ticks=range(len(tempvar)), labels=[1, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 12])
```

```
    plt.ylabel(y_label)
    plt.ioff()
    plt.savefig("graph.png", format='png')
    plt.close()

    graph = Image(Point(25, 23), "graph.png")
    graph.draw(win)

    #Here the count = 1 is a flag. To enter this part of the code
only at a specific time of the program
    while True and count == 1:
        click = win.getMouse()

        if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
            win.close()
            quit()

    count = 1

    return count

def barplot(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
    #We create the Bar Plot
    text = Text(Point(25, 46), 'Bar Plot')
    text.setSize(17)
    text.setTextColor("brown")
    text.draw(win)

    sect = Rectangle(Point(37, 35), Point(47, 9))
    sect.setFill(color_rgb(139,69, 19))
    sect.draw(win)

    mat = Text(Point(42, 33), "1-Basic Materials")
    mat.setSize(17)
    mat.setFace("arial")
    mat.setTextColor("white")
    mat.draw(win)
    co = Text(Point(42, 31), "2-Communication Services")
    co.setSize(17)
    co.setFace("arial")
```

```
    co.setTextColor("white")
    co.draw(win)
    cyc = Text(Point(42, 29), "3-Consumer Cyclical")
    cyc.setSize(17)
    cyc.setFace("arial")
    cyc.setTextColor("white")
    cyc.draw(win)
    cde = Text(Point(42, 27), "4-Consumer Defensive")
    cde.setSize(17)
    cde.setFace("arial")
    cde.setTextColor("white")
    cde.draw(win)
    cdi = Text(Point(42, 25), "5-Consumer Discretionary")
    cdi.setSize(17)
    cdi.setFace("arial")
    cdi.setTextColor("white")
    cdi.draw(win)
    en = Text(Point(42, 23), "6-Energy")
    en.setSize(17)
    en.setFace("arial")
    en.setTextColor("white")
    en.draw(win)
    fin = Text(Point(42, 21), "7-Financial Services")
    fin.setSize(17)
    fin.setFace("arial")
    fin.setTextColor("white")
    fin.draw(win)
    hel = Text(Point(42, 19), "8-Healthcare")
    hel.setSize(17)
    hel.setFace("arial")
    hel.setTextColor("white")
    hel.draw(win)
    ind = Text(Point(42, 17), "9-Industrial")
    ind.setSize(17)
    ind.setFace("arial")
    ind.setTextColor("white")
    ind.draw(win)
    ret = Text(Point(42, 15), "10-Real Estate")
    ret.setSize(17)
    ret.setFace("arial")
    ret.setTextColor("white")
    ret.draw(win)
```

```
    tech = Text(Point(42, 13), "11-Technology")
    tech.setSize(17)
    tech.setFace("arial")
    tech.setTextColor("white")
    tech.draw(win)
    uti = Text(Point(42, 11), "12-Utilities")
    uti.setSize(17)
    uti.setFace("arial")
    uti.setTextColor("white")
    uti.draw(win)

    if len(chosen_var) == 1:
        tempvar = pd.value_counts(df[chosen_var[0]])
        plt.bar(tempvar.index,tempvar.values, align='center')
        plt.title(title)
        plt.xlabel(x_label)
        plt.xticks(ticks=range(len(tempvar)), labels=[1, 2, 3, 4,
5, 6, 7, 8, 9, 10, 11, 12])
        plt.ylabel(y_label)
        plt.ioff()
        plt.savefig("graph.png", format='png')
        plt.close()

        graph = Image(Point(25, 23), "graph.png")
        graph.draw(win)

        #Here the count = 1 is a flag. To enter this part of the
code only at a specific time of the program
        while True and count == 1:
            click = win.getMouse()

            if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        count = 1

        return count

    elif len(chosen_var) == 2:
        tempvar = df.groupby(chosen_var[0])[chosen_var[1]].mean()
```

```
        plt.bar(tempvar.index,tempvar.values, align='center')
        plt.title(title)
        plt.xlabel(x_label)
        plt.xticks(ticks=range(len(tempvar)), labels=[1, 2, 3, 4,
5, 6, 7, 8, 9, 10, 11, 12])
        plt.ylabel(y_label)
        plt.ioff()
        plt.savefig("graph.png", format='png')
        plt.close()

        graph = Image(Point(25, 23), "graph.png")
        graph.draw(win)

        #Here the count = 1 is a flag. To enter this part of the
code only at a specific time of the program
        while True and count == 1:
            click = win.getMouse()

            if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        count = 1

        return count

def scatterplot(chosen_var, df, title, x_label, y_label, win,
quitx1, quitx2, quity1, quity2,count):
    #Here we create the scatterplot
    text = Text(Point(25, 47), 'Scatter Plot')
    text.setSize(17)
    text.setTextColor("brown")
    text.draw(win)

    if len(chosen_var) == 2:
        sect = Rectangle(Point(37, 35), Point(47, 9))
        sect.setFill(color_rgb(139,69, 19))
        sect.draw(win)

        mat = Text(Point(42, 33), "1-Basic Materials")
        mat.setSize(17)
```

```
        mat.setFace("arial")
        mat.setTextColor("white")
        mat.draw(win)
        co = Text(Point(42, 31), "2-Communication Services")
        co.setSize(17)
        co.setFace("arial")
        co.setTextColor("white")
        co.draw(win)
        cyc = Text(Point(42, 29), "3-Consumer Cyclical")
        cyc.setSize(17)
        cyc.setFace("arial")
        cyc.setTextColor("white")
        cyc.draw(win)
        cde = Text(Point(42, 27), "4-Consumer Defensive")
        cde.setSize(17)
        cde.setFace("arial")
        cde.setTextColor("white")
        cde.draw(win)
        cdi = Text(Point(42, 25), "5-Consumer Discretionary")
        cdi.setSize(17)
        cdi.setFace("arial")
        cdi.setTextColor("white")
        cdi.draw(win)
        en = Text(Point(42, 23), "6-Energy")
        en.setSize(17)
        en.setFace("arial")
        en.setTextColor("white")
        en.draw(win)
        fin = Text(Point(42, 21), "7-Financial Services")
        fin.setSize(17)
        fin.setFace("arial")
        fin.setTextColor("white")
        fin.draw(win)
        hel = Text(Point(42, 19), "8-Healthcare")
        hel.setSize(17)
        hel.setFace("arial")
        hel.setTextColor("white")
        hel.draw(win)
        ind = Text(Point(42, 17), "9-Industrial")
        ind.setSize(17)
        ind.setFace("arial")
        ind.setTextColor("white")
```

```
        ind.draw(win)
        ret = Text(Point(42, 15), "10-Real Estate")
        ret.setSize(17)
        ret.setFace("arial")
        ret.setTextColor("white")
        ret.draw(win)
        tech = Text(Point(42, 13), "11-Technology")
        tech.setSize(17)
        tech.setFace("arial")
        tech.setTextColor("white")
        tech.draw(win)
        uti = Text(Point(42, 11), "12-Utilities")
        uti.setSize(17)
        uti.setFace("arial")
        uti.setTextColor("white")
        uti.draw(win)

        tempvar = df.groupby(chosen_var[0])[chosen_var[1]].mean()
        plt.scatter(tempvar.index, tempvar.values,marker="*")
        plt.title(title)
        plt.xlabel(x_label)
        plt.xticks(ticks=range(len(tempvar)), labels=[1, 2, 3, 4,
5, 6, 7, 8, 9, 10, 11, 12])
        plt.ylabel(y_label)
        plt.ioff()
        plt.savefig("graph.png", format='png')
        plt.close()

        graph = Image(Point(25, 23), "graph.png")
        graph.draw(win)

        #Here the count = 1 is a flag. To enter this part of the
code only at a specific time of the program
        while True and count == 1:
            click = win.getMouse()

            if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        count = 1
```

```
        return count

    elif len(chosen_var) == 3:
        plt.figure(figsize=(10, 6))
        sectors = df[chosen_var[0]].unique()

        for sector in sectors:
            sector_df = df[df[chosen_var[0]] == sector]
            tempvar =
sector_df.groupby(chosen_var[1])[chosen_var[2]].mean().reset_ind
ex()
            plt.scatter(tempvar[chosen_var[1]],
tempvar[chosen_var[2]], label=sector)

        plt.legend(title=chosen_var[0])
        plt.title(title)
        plt.xlabel(x_label)
        plt.ylabel(y_label)
        plt.ioff()
        plt.savefig("graph.png", format='png')
        plt.close()

        #Here we resize the image
        with im(filename="graph.png") as img:
            img.resize(830, 660)
            img.save(filename="graph.png")

        graph = Image(Point(26, 23), "graph.png")
        graph.draw(win)

        #Here the count = 1 is a flag. To enter this part of the
code only at a specific time of the program
        while True and count == 1:
            click = win.getMouse()

            if quitx1 <= click.getX() <= quitx2 and quity2 <=
click.getY() <= quity1:
                win.close()
                quit()

        count = 1
```

```
        return count

if __name__ == "__main__":
    main()
```