

---

# Contents

---

## **1. Python Project Structure**

- 1.1 Introduction
- 1.2 The Methodology
- 1.3 Versioning
  - I Version 0
  - II Version 1
  - III Version 2
- 1.4 Conclusion

## **2. Changelog**

- 2.1 Version 0 to Version 1
- 2.2 Version 1 to Version 2

# SECTION 1

---

## Project Structure

---

*Stock Portfolio – Write a program to hold a stock portfolio. User will be able to define her portfolio and follow share prices. If you are very engaged in it, you can even suggest what to sell or buy.*

### 1.1 Introduction

Winterfell is a stock portfolio tracking and management platform. The user shall be able to construct his/her portfolio of equity shares listed on the New York Stock Exchange, subsequently investing and/or divesting from their portfolio, all using real time market prices retrieved from the NYSE.

In addition, the user shall have comprehensive summary of the performance of their portfolio, including metrics like returns, current market value, etc. The user shall be able to retrieve fundamental market information for each stock like PE Ratio, Earnings per share, etc.

The user shall be able to aid his buy or sell decisions by analyzing the past performance of his stocks over different time horizons (e.g. – 6months, 1 year, 5-year stock prices graphs).

### 1.2 The Methodology

Winterfell is fundamentally dependent on three libraries: Yahoo Finance's Application Programming Interface (API) to pull stock data, Tkinter to generate an extensive graphical user interface, and matplotlib to generate graphs for the user to be able to analyze the performance of stocks over periods of time.

## 1.3 Versioning

This section details the build-up of the Winterfell platform including the different stages of development categorized into three versions – Version 0 (V0), Version 1 (V1), and Version 2 (V2). We illustrate the outcomes of the python code and libraries in each version below.

### I. Version 0 – (V0)

The maiden version of Winterfell, Version 0 (V0), provides a platform for users to input their desired NYSE stock ticker and in turn, displays information on the inputted stock in terms of the day's price and movement. To do this, we run the “main” class which will call the “stocks” class, thus displays the interface as seen in Figure 1.1 below. Moreover, V0 provides market data on the inputted stock ticker, that include but are not limited to Earnings Per Share, Average Traded Volume, Market Capitalization, Price-to-Earnings Ratio, Previous Close, Open Price, Day's Range, to mention a few. The Tkinter library is responsible for the graphic user interface (GUI) that enables the user to interact with the platform.

Generally, there is a functionality that runs the application, sets the main title, logo, and keeps the window open. Precisely, there are functionalities for stock selection and market data (stock fundamentals).

- a. *Winterfell (App Functionality)*: To start with, we define a class – Winterfell – in the “main” file. We then further define the attributes and properties that enable the functionality of the class. Firstly, we create frames using Tkinter to house and arrange navigation buttons in the Winterfell V0 interface. Furthermore, for subsequent versions, we create dictionaries that will hold all pages (or classes) when we have more windows to be displayed.

Secondly, we define another class – Stocks – in the “stock” file which is called by the “main” file. In this class, we also define attributes and properties that validate the functionality of the Stocks class. Frames were set up to house the strings displayed, navigation buttons such as “Confirm”, and an entry widget as in Figure 1.1.

- b. *Stock Selection*: Since we can define different instances or attributes or objects in a class. Thus, we also define the instance of displaying the details of the desired stock in the Stocks class. In this case, we define frames to house stock information. Also, we connect with Yahoo Finance's API to display stock details including prices changes and movements.

Welcome to Winterfell!

Enter the NYSE stock ticker to see current price  
and day's movement

**Figure 1.1 – Winterfell**

First, we define a function to refresh the interface upon stock selection and create frames to house the stock ticker symbol and its associated information i.e., current price and day's movement in price (in percentage and absolute terms). Then, we pull real-time stock information from Yahoo Finance by calling its API and storing the requested information in the defined frames. Consequently, we get Figure 1.2 below.

MSFT	324.9 USD
(0.32%)	1.06

MSFT
<input type="button" value="Confirm"/>

**Figure 1.2 – Stock Selection**

- c. *Market Data (Stock Fundamentals)*: We define another function that displays additional market information pulled from Yahoo Finance. Just as we did for previous attributes, we define a frame to house additional market data on selected stock – “*stock information*” – pulled from Yahoo Finance. Upon clicking this

navigation button, a new window is opened that displays market data on selected stock as seen in Figure 1.3.

MSFT
1y Target Est : 364.9
52 Week Range : 211.94 - 349.67
Ask : 325.19 x 1300
Avg. Volume : 26581785.0
Beta (5Y Monthly) : 0.87
Bid : 325.01 x 900
Day's Range : 323.02 - 336.76
EPS (TTM) : 8.94
Earnings Date : Jan 24, 2022 - Jan 28, 2022
Ex-Dividend Date : Feb 16, 2022
Forward Dividend & Yield : 2.48 (0.74%)
Market Cap : 2.439T
Open : 335.71
PE Ratio (TTM) : 36.35
Previous Close : 334.65
Quote Price : 324.8999938964844
Volume : 34567321.0

**Figure 1.3 – Stock Details**

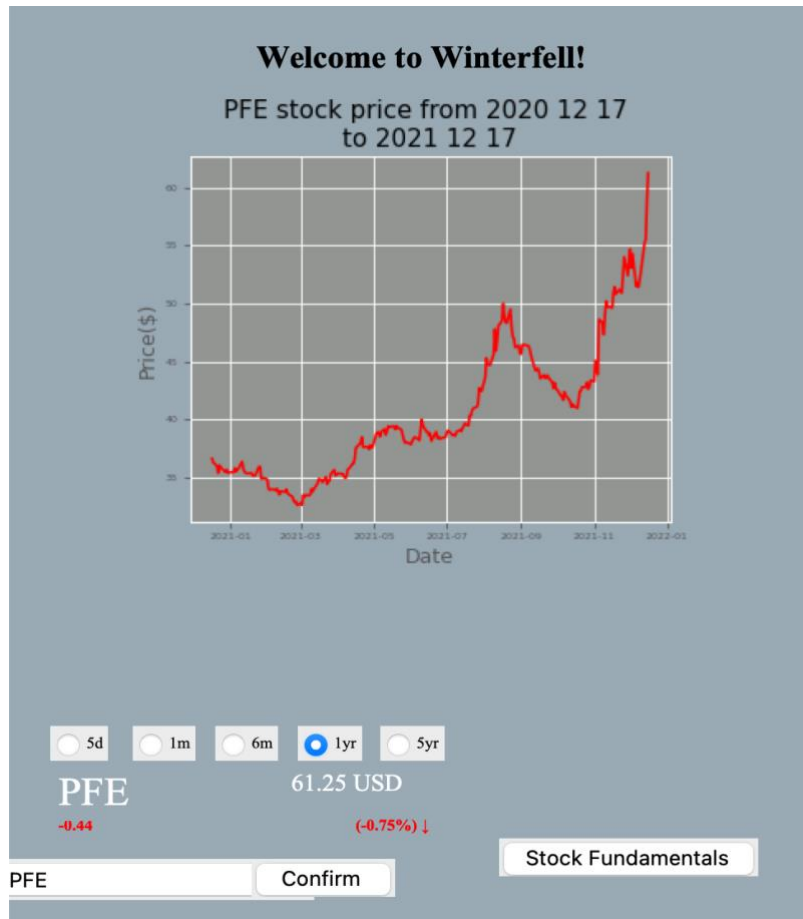
Interestingly, after all major instances have been defined, we also define error instructions for invalid ticker symbols entered since this is the only case we get inputs from users in this version. See example in Figure 1.4.

**Invalid ticker entered - must be NYSE stock code.**

**Figure 1.4 – Error Handling**

## II. Version 1 – (V1)

Version 1 (V1) is a build-up on V0 with more functionalities using the python libraries earlier mentioned. In a holistic view, our code returns Figure 2.1 below. Here, we introduced graphs and plots for a certain timeline using matplotlib.



**Figure 2.1 – Winterfell (V1)**

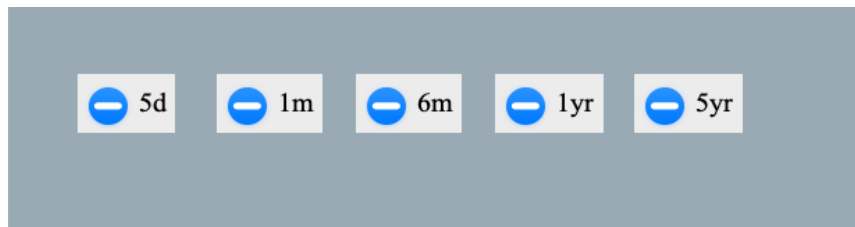
Figure 2.1 is still a partial deployment of the Winterfell platform which depicts the different functionalities that are built into this version. In this version, Winterfell allows the user to view the details of selected stocks ranging from trend to the company's market data. Like the V0, we run the "main" class in V1, which will call the "stocks" class, thus displays the interface.

Specifically, there are functionalities for stock selection, graph of selected stock, timeline options for the graph of selected stock, and detailed market data of the selected stock. Stock selection and market data have been discussed in V0. The app functionality remains the same but with additional features which are described below.

- a. *Graph of Selected Stock:* We import the Matplotlib used to plot, manipulate, and style plots of the selected stock. We also set defaults for variables used throughout

the program such as fonts, navigation colors, stock graph colors, to mention a few. Using the matplotlib, we plot the graph of the selected stock with the current live data obtained from calling Yahoo Finance API. We set the colors of the graph as green or red using conditional logic - if statements. Tkinter was employed to design the frame that holds the graphed stock data and matplotlib to make the graph visible and fit into the frame. The title for the stock graph is displayed. Also, sliders are for adjusting the position of the graphs are made available.

- b. *Timeline Options:* Using Tkinter, we design radio buttons for users to choose the desired timeline for the stock trend to be displayed. This is done for the 5-day, 1 month, 6-month, 1-year, and 5-year timelines as displayed in Figure 2.2 below. In the event there is no timeline selected, just the stock details are displayed as in V0.



**Figure 2.2 – Timeline Options**

### **III. Version 2 – (V2)**

V2 is an improved version of V1, which fulfils the main objective of Winterfell – the user shall be able to construct his/her portfolio of equity shares listed on the New York Stock Exchange, subsequently investing and/or divesting from their portfolio, all using real time market prices retrieved from the NYSE.

All functionalities of V0 and V1 are still present but improved upon in V2. In V0, we were able to achieve stock selection and retrieve market data and information. In V1, we were able to introduce a graphical representation of the selected stock which facilitates trend analysis for users before buying or shorting a stock in the portfolio. V2 completes the project by introducing a login page for different users, introducing a window for investment or divestment, and afterwards, a dashboard where users can view their entire portfolio balance and summary. As discussed earlier, a dictionary was created to house all pages which are being looped over for display given a certain condition is true.

- a. *Signup & Login Page:* We import the OS module in python so we can define functions for creating a files to store user information. We create frames for the homepage to house the signup navigation button and create containers for signup credentials (see Figure 3.1). We create a temporary file where these credentials

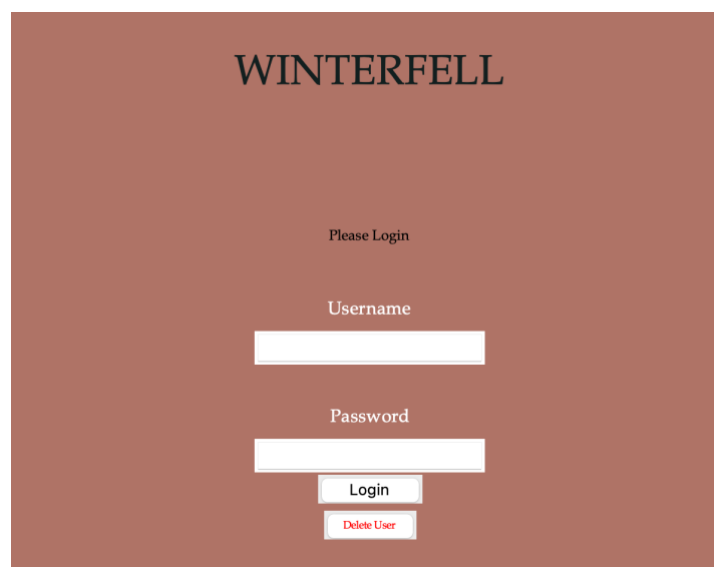
are stored upon signing up which later help us to create actual files. We create a login page. Similarly, we create frames to house navigation buttons and containers for user credentials on the login page (see Figure 3.2).

We then specify a conditional statement to check whether login information matches the signup data. The temp file is referenced to match with the credentials provided so users can have access to the platform. It is important to note that if wrong credentials are provided, the program prints (on a new window) “invalid login” to the user (see Figure 3.3).



The image shows a web page titled "WINTERFELL" with a dark red background. Below the title, the text "Please Enter new Credentials" is centered. There are two input fields: "New Username" and "New Password". Below the "New Password" field is a "Signup" button.

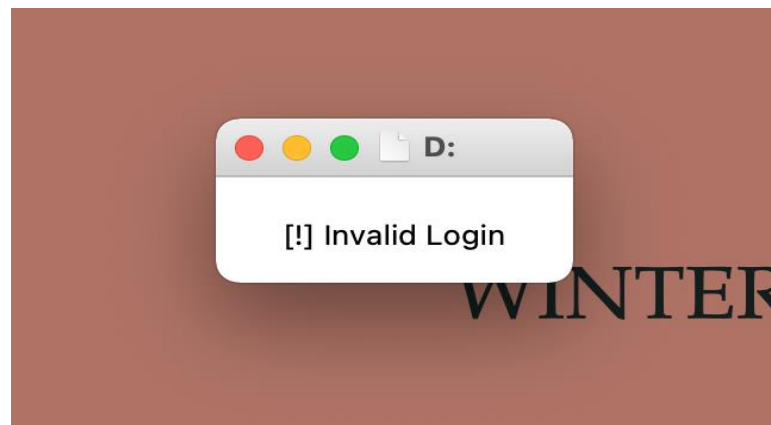
**Figure 3.1 – Signup Page**



The image shows a web page titled "WINTERFELL" with a dark red background. Below the title, the text "Please Login" is centered. There are two input fields: "Username" and "Password". Below the "Password" field are two buttons: "Login" and "Delete User".

**Figure 3.2 – Login Page**

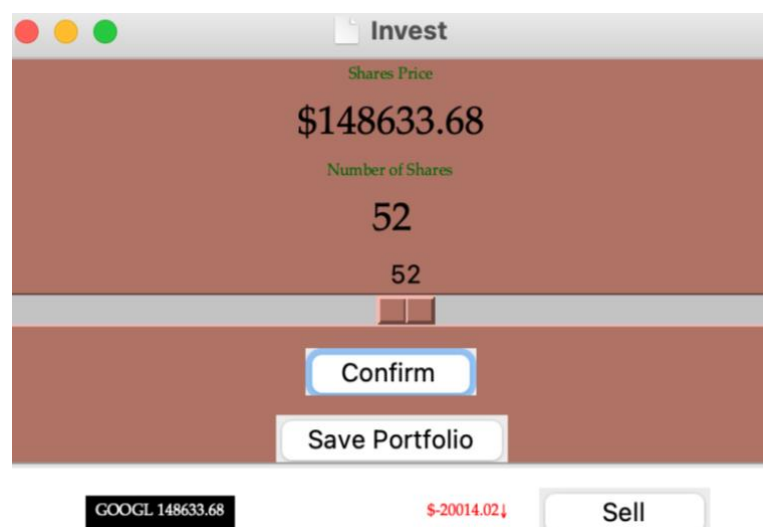




**Figure 3.3 – Error Handling (Wrong Credentials)**

If need be, the delete user button in Figure 3.2 allows users to go back to the signup page and restart. This is useful when users are not sure of the credentials provided for signing up.

- b. Investment/Divestment:* We develop the investment page by designing frames to house navigation buttons, labels, and slide bars in the new investment window. We define an “invest” function that allows users to view share price and number of shares bought. As the slide bar is moved to the desired position, the user presses the confirm button and this saves the number of shares bought or sold (if stock is already in portfolio) and entire investment amount (see Figure 3.4). For consistency, we define functions and conditional statements to display stock figures in green or red to indicate rising and falling prices, respectively.

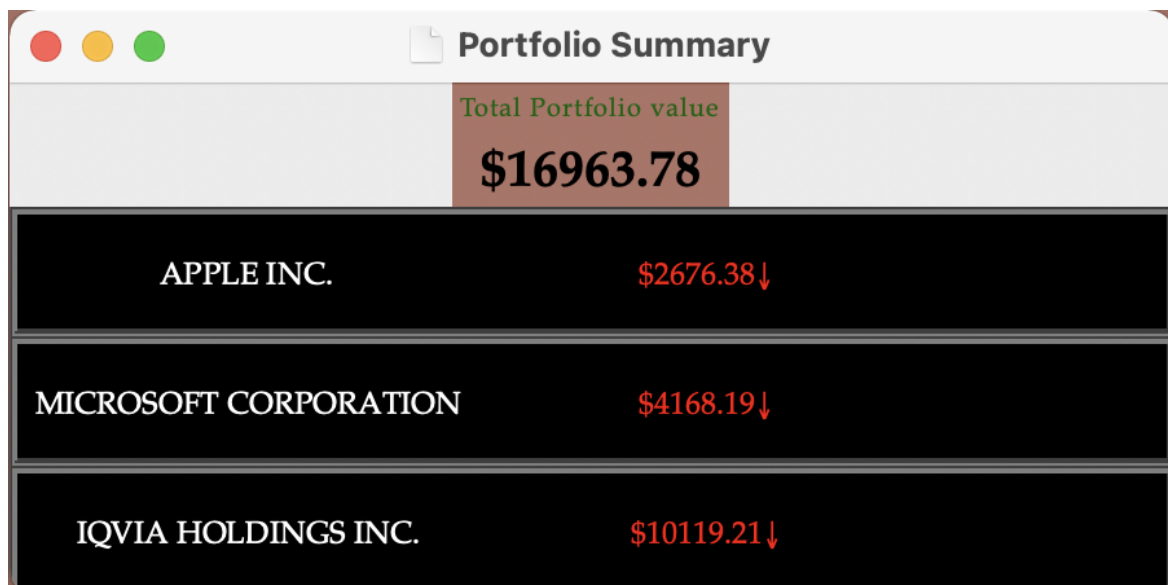


**Figure 3.4 – Investment/Divestment Page**

Upon confirmation of the invested stock, we define an empty list which is appended as more stocks are bought. In the event where stock is already exists in the list, the

additional value is added to the already existing object in the list, which leads to our portfolio summary.

- c. *Portfolio Summary*: We create a list to store all stocks in the user portfolio. We design frames to hold and visualize the total portfolio value, company name, and corresponding stake as seen in Figure 3.5.



**Figure 3.5 – Portfolio Summary**

We define a portfolio summary function in the `Invest_Page` class which reads the holdings of the user, which is then looped over to create a frame for every stock invested by the user and displays the real time stock info.

## 1.4 Conclusion

We sought to provide users of Winterfell with access to manage and track their stock portfolio, which eventually came to fruition in three versions. Winterfell was developed in three versions with each version having its specific improvement and feature(s). Subsequent versions include features of the previous versions but with an add-on for the user. That is, functions and objects defined in the previous versions were also used in later versions but were enhanced. The different versions of Winterfell show the roadmap (further detailed in the changelog) of how this project was done with the most basic form corresponding to V0 and the final corresponding to V2.

# SECTION 2

---

## Changelog

---

### 2.1 Version 0 to Version 1

- V0 was initially built using conditional logic and loops to return market information such as real time stock data and market fundamentals to the user.
- Further developments were introduced via the adoption of the object-oriented functionality of python.
- Subsequently, this led to the creation of classes and objects in our program and led to the final version of V0.
- Improving on what was done, we introduced graphs to bring more functionality to Winterfell, leading to V1.
- V1 allows users to aid investment decisions by giving access to historical trend and market information.

### 2.2 Version 1 to Version 2

- V1 was an improved version of V0 as it provided more functionality to users.
- However, the main objective of the platform was not yet achieved in V1 as users were not able to view the portfolio summary and invest and/or divest in stocks from the NYSE.
- To do this, we enhanced the V1 by creating more pages and functionalities on the Winterfell platform so users can access their accounts and buy/sell their desired stocks, which led to V2.
- V2 checks all the boxes on what we aim to achieve on the platform.