Mini project report

Stock Analyser

Zhu Huijin, Jia Xiaorui, An Yueran

G2004249F G2000863J G2004211C

Catalogue

[1. Introduction 1](#_Toc83195494)

[2. Design and implementation 1](#_Toc83195495)

[1) Dependency, Structure and Flowchart 1](#_Toc83195496)

[2) Description of important user defined functions 3](#_Toc83195497)

[3) Program testing and test cases 4](#_Toc83195498)

[3. Challenges 7](#_Toc83195499)

[4. Limitations and improvement suggestions 7](#_Toc83195500)

# Introduction

In our previous practical experience, we found that when conducting industry analysis or comparative analysis of the same industry, we need to check financial data on the official website of the corresponding company. This process is repetitive and tedious. To solve this problem, we want to use python to capture financial data, so as to achieve the purpose of simplifying the process and improving work efficiency.

Our project is used for:

1. Crawl the financial data of A-share listed companies from Snowball.

2. Calculate financial ratios such as operating profit margin, net profit margin and gross profit margin of related companies

3. Draw a line chart of the company's income, profit and other financial data in recent years.

4. Conduct a horizontal comparison and analysis of the financial development of many companies in the same industry.

Our symbol.txt contains all the stock codes for which we can find relevant financial information, and there are about 3,000 listed companies. Users can compare the financial data of multiple companies at the same time, and can also view the financial data of a company in recent years. We use the gui module to add a user interface to the entire project, so as to enhance the user experience. First, users can grab the required financial data of listed companies through Add and Select File on the page. Second, Crawl Progress can display the progress of the financial data that has been captured. Finally, the 2013-2017 revenue, Profit and Shareholder Profit data of related listed companies are obtained and a line chart is formed, so as to clearly and intuitively convey the financial development trend of related companies.

# Design and implementation

## Dependency, Structure and Flowchart

**Dependency**

* matplotlib==3.4.1
* pandas==1.2.4
* requests==2.25.1
* PyQt5==5.15.4
* threading

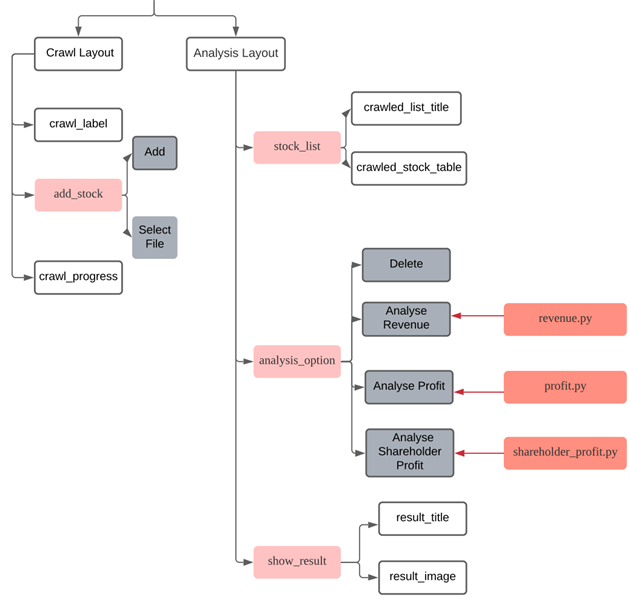
We choose PyQt5 GUI Framework to handle our Widget management, which makes it easy to generate an elegant layout.

We use requests and threading library for our crawl progress, which provide the high concurrency for data downloading.

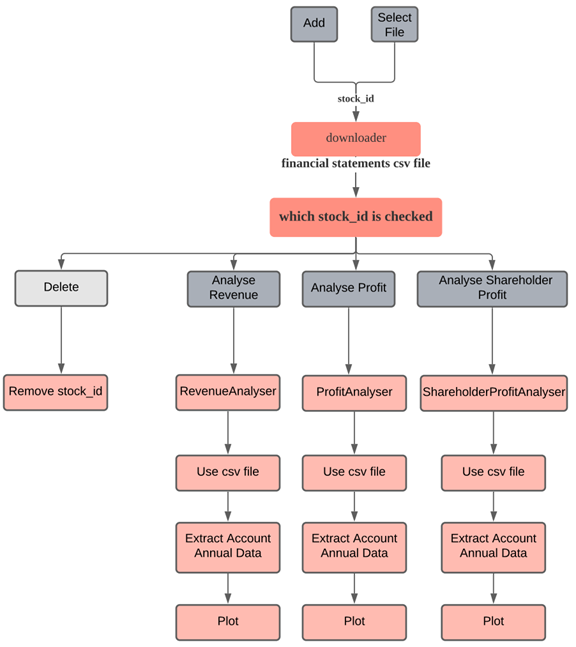
When it comes to data analysis, we use pandas library to help with the data processing of downloaded data.

The matplotlib is a powerful plot library with various pattern, so we use it to draw our analysis figures.

**Structure**



**Flowchart**



## Description of important user defined functions

There are several user defined functions in this project:

**Download.py**

download\_lrb()/ download\_fzb()/ download\_llb(): Crawl the income/ cashflow statement/ balance sheet statement data from Snowball (http://api.xueqiu.com/stock/f10/) and store the data in a file locally for

futher analysis.

downloadStock(): As the inefficiency when having a set of data files need to be downloaded, we introduce a thread pool to crawl the data needed concurrently, which improves the download speed 10x.

**shareholder\_profit.py**

get\_all\_data(): select the information of all account items in selected stock's income statement (in chronological order, Python datetime as index).

get\_data\_month(): #extract all account info whose date ends at the "specified month number", set month=12 then we can get Annual report number in each year.

get\_data\_ratio: get Shareholder Profit Account number "归属于母公司的净利润" , split this account name into three keywords: "归属", "母公司" and "净利润" and check them together, so this program works correctly even though account name might differ a little.

data\_plot(): find 5 most recent dates time, ilocate all income statement information at that date; plot Shareholder Profit Account number with respect to which year of these 5 dates. The plot will be saved loally and also shown in the result layout.

**revenue.py & profit.py**

Most functions and structures are the same as that of shareholder\_profit.py but select different data.

data\_ratio(): get Profit Account number/ get Revenue Account number

data\_plot(data\_ratio): plot Profit Account number/ Revenue Account number with respect to time

**gui+error.py**

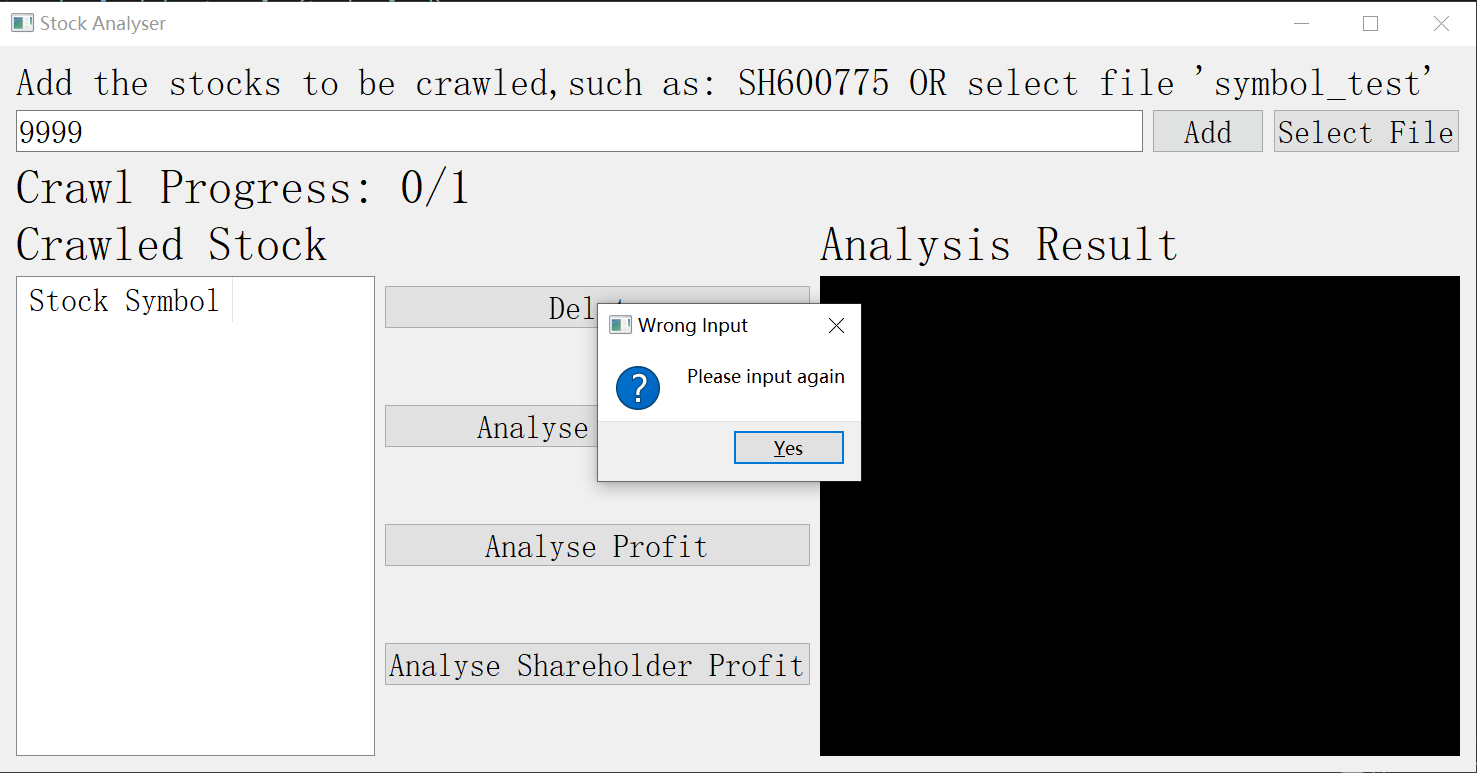
addStock(): If the stock number that user inputs is correct, which means it is within the all stock file, we will crawl the related data using our concurrent downloader, after which the stock number will be added into the Stock List layout successfully. But if the stock number is invalid, the user will be warned “Please input again”. Nothing will be downloaded and the stock number will not be added.

addStockFile(): First, check if the file added is valid or not. The file will not be added if it is blank. Then, remove extra blank space from every line read, and then the crawl progress will be displayed. And stocks that have been downloaded before will not be downloaded again.

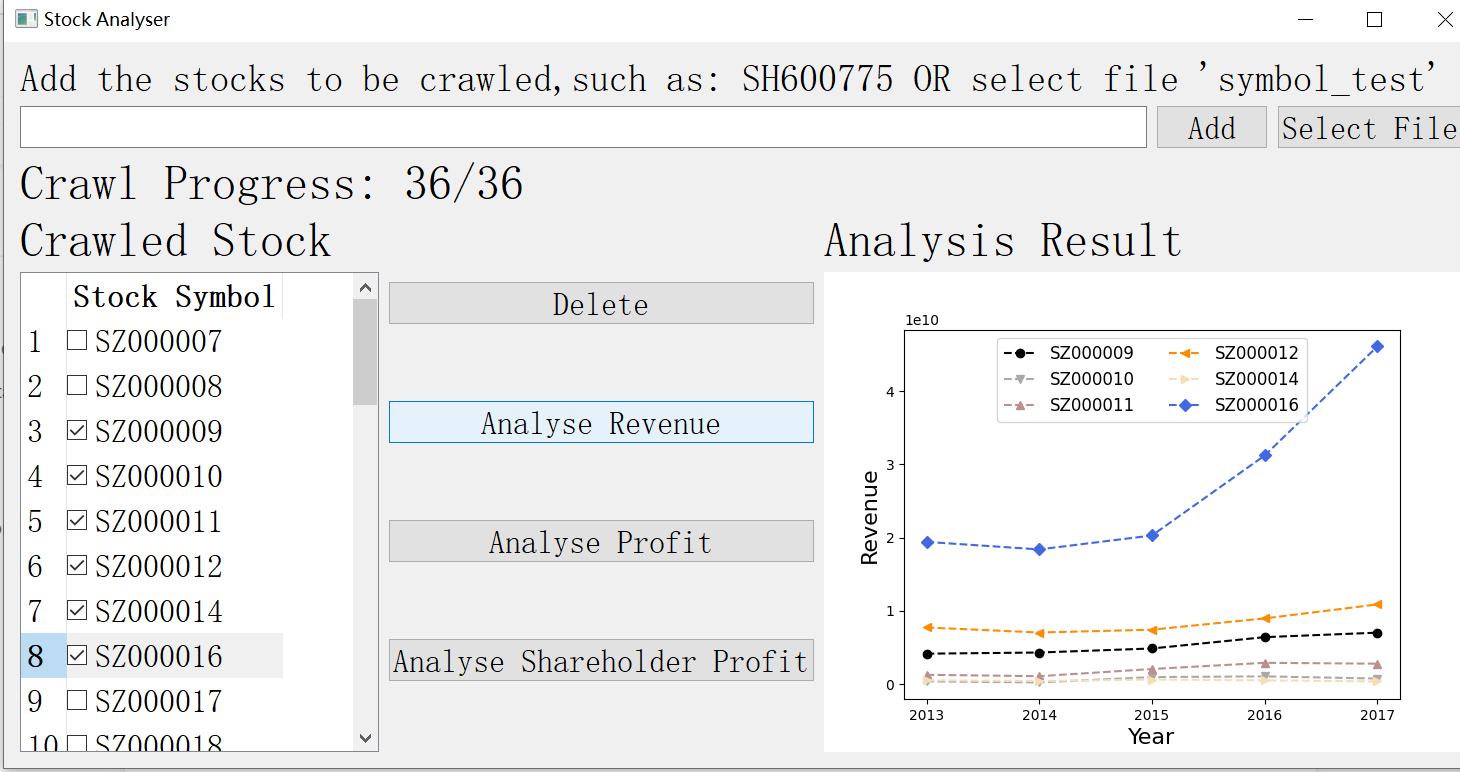
analyseRevenue()/ analyseProfit()/ analyseShareholderProfit(): Find out which stocks have been chosen and analyse the first 10 stocks. Then show the graph.

## Program testing and test cases

In the addStock() function, if users key in invalid inputs that do not match any stock number, the error message will be displayed and user will be warned “Please input again”. And the incorrect stock number will not be added.

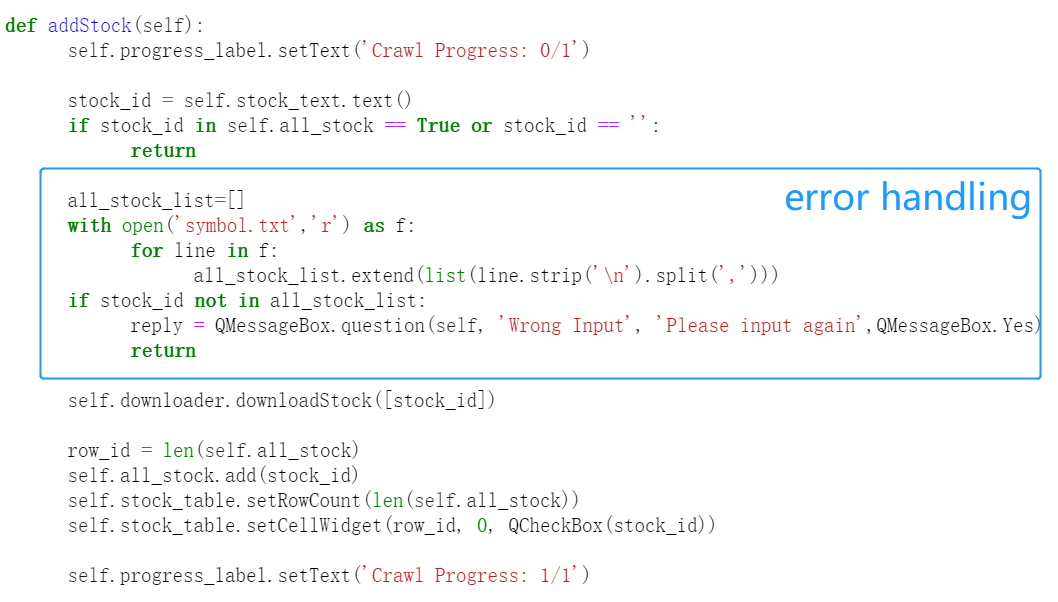


Analyse: e.g. Analyse Revenue



# Challenges

Learning point: error handling in gui part is struggling at first, because this program needs stock code input which consists of English letters and number. So in QLineEdit, QIntValidator which restricts input to integer and QDoubleValidator which restricts input to float could not be used. QRegExpValidator seems practicable, but only guarantees valid input form not the content, the input should be a meaningful stock code. So we decided to constrain the input in our all\_stock\_list which covers almost all stock in Shenzhen and Shanghai Stock Exchange, and designed a pop-up warning window to remind user to input again when input is invalid.



In the process of writing the source code, we encountered the following challenges.

Since the stock codes of related listed companies need to be added separately, when analyzing more companies at the same time, it will be very cumbersome for users to input one by one. Therefore, we use select file to set up a symbol\_test file, which can import the stock codes of multiple companies at one time, so as to obtain financial data more quickly and conveniently.

In addition, when writing the error handling part of the code, because it is not an integer or float checker, and the code is English plus numbers, and the regular expression checker is not perfect. Although users can add numbers in English, the output is possible Is an invalid stock code. So in the end, we limited the ‘input’ to thousands of stock codes (symbol files).

# Limitations and improvement suggestions

Our data is the financial data of listed companies before 2018. If we want the latest financial data, we need to constantly update relevant data sheets. In order to improve this problem, we can grab the latest financial data of listed companies through some online databases. Therefore, users can conduct a more complete financial analysis.

Besides, our financial data is mainly obtained from Snowball. The amount of financial data of listed companies is limited, and users may not be able to find financial data of non-A shares. In order to improve this shortcoming, we can crawl more detailed and complete financial data through python, so as not to restrict A shares or U.S. stocks, hence, users can complete a more comprehensive research and analysis of an industry.