Python 程序设计实验报告

姓名：李尚坤 学号：20307130215 指导教师：秦亚杰

一、实验目的及要求

1、从网上获取一支股票 2021 年 2 月的价格信息，其中股票代码中连续三位与学号末三位相同，画出股票的日 K 线图；
2、假设从 2 月 1 日至 10 日的每一个交易日定投买入 1 手，买入价格为当天的收盘价，之后一直持有买入的股票。计算出总的本金投入、期末总资产、期末总收益、收益率和最大回撤率，并画出每日总资产与收益率曲线；
3、设计一个 GUI 界面，输入股票代码、起止时间、K 线类别，自动爬取并绘制相应的曲线。

二、实验设备与环境

macOS 11.2.3, PyCharm Edu 2020.3.4, Python Interpreter: Python 3.9

三、实验原理

1、基于 baostock 库的股票数据获取；
2、基于 pandas 库的数据分析处理；
3、基于 matplotlib 库的数据绘图（数据可视化）；
4、基于 mplfinance 库的金融数据可视化分析。

四、实验设计方案

（一）实验一：绘制日 K 线，DisplayKLine_StuNum.py

1、利用 import 引入程序所需要的库。实验一中需要用到的库有 baostock, pandas, mplfinance, matplotlib。

```
import baostock as bs  # 引入程序所需要的库
import pandas as pd
import mplfinance as mpf
import matplotlib as mpl
import matplotlib.pyplot as plt
import sys
```

2、设置绘图的参数，其中下图第一行设置负号显示，第二行设置中文字体为“Arial Unicode MS”。由于本实验操作基于 macOS 系统，因此当本程序在 Windows 系统上运行时可能会出现乱码，无法显示中文字体，此时应当更换中文字体的样式。

```
mpl.rcParams['axes.unicode_minus'] = False  # 设置负号显示
mpl.rcParams['font.sans-serif'] = ['Arial Unicode MS']  # 设置中文显示
```

3、定义一个函数 getStockData()。该函数输入为股票代码、起止时间、数据类型，然后该函数进行数据爬取获得股票的日期、开收盘价、最高最低价等信息，最后返回一个 DataFrame 的数据结构。

```
def getStockData(stockcode, startdate, enddate, periodtype):  # 用于获取股票信息并返回一个含有股票信息的DataFrame数据结构
    bs.login()  # 登陆系统
    rs = bs.query_history_k_data_plus(stockcode, "date,code,open,high,low,close",
                                      start_date=startdate,
                                      end_date=enddate,
                                      frequency=periodtype)
    data_list = []
    while (rs.error_code == '0') & rs.next():  # 获取一条记录，并将记录合并在一起
        data_list.append(rs.get_row_data())
    df = pd.DataFrame(data_list, columns=["date", "code", "open", "high", "low", "close"])  # 创建一个DataFrame数据结构
    df['open'] = df['open'].astype('float')   # 将获得的数据转换为浮点数类型
    df['high'] = df['high'].astype('float')
    df['low'] = df['low'].astype('float')
    df['close'] = df['close'].astype('float')
    df['date'] = pd.to_datetime(df['date'])  # 将date列数据类型转换为时间类型
    df.set_index(['date'], inplace=True)  # 将日期列作为行索引
    bs.logout()
    return df
```

4、定义一个函数 candle_draw()。分别设置图形颜色、图形风格、字体样式与基本参数等信息，使得该函数用于绘制 K 线图。特别需要注意的是，本实验在 macOS 系统中进行，因此设置的字体样式为“Arial Unicode MS”，若该程序在 Windows 系统中运行，则需要更改字体样式。

```
def candle_draw(data, fig1, ax1, title1):
    mc = mpf.make_marketcolors(     # 设置marketcolors
        up='red',                   # 设置收盘价大于等于开盘价时K线线柱的颜色
        down='green',               # 设置收盘价小于开盘价时K线线柱的颜色
        edge='i',                   # 设置K线线柱边缘的颜色，i表示继承自up和down的颜色，下同
        wick='i')                   # 设置上下影线的颜色
    s = mpf.make_mpf_style(         # 设置图形风格
        gridaxis='both',            # 设置网格线位置
        gridstyle='-.',             # 设置网格线线型
        y_on_right=False,           # 设置y轴位置是否在右
        marketcolors=mc)
    mpl.rcParams['lines.linewidth'] = 0.5
    font1 = {'family': 'Arial Unicode MS',  # 设置字体样式
             'weight': 'bold',
             'size': 10, }
    ax1.set_title(title1, font1)  # 设置标题
    kwargs = dict(type='candle',)  # 设置基本参数，其中type选择K线图
    mpf.plot(data, **kwargs, tight_layout=True, style=s, ax=ax1)
    fig1.canvas.draw_idle()
```

5、通过程序入口 if __name__ == ‘__main__’，开始执行整个程序。先通过 input() 函数输入学号的最后三位数字，然后在所有股票代码中查找含有这三个数字的股票代码并取第一个满足条件的股票代码。获取股票代码后，设置好起止时间，通过 getStockData()函数获得对应股票的价格信息。最后通过 candle_draw()函数绘制出 K 线图。

```python
if __name__ == '__main__':
    stockcode = input('请输入学号的最后三位: ')
    index = stockcode
    allStockLstFile = 'stock_basic.csv'
    with open(allStockLstFile, 'r', encoding='gbk') as fh:  # 从所有股票代码中找到包含输入3位数字的股票代码
        while True:
            line = fh.readline()
            if not line:
                break
            tmpinfo = line.strip('\n').split(',')
            if tmpinfo[4] == '1' and tmpinfo[5] == '1':  # 去掉股票代码中的'sh.'或'sz.', 并查看是否包含输入的3位数字, 防止用户输入字母
                if stockcode in tmpinfo[0][3:]:
                    stockcode = tmpinfo[0]
                    break

    if len(stockcode) != 9:  # 如果没有找到相应的股票代码
        print('未找到最后三位为{}的股票'.format(index))
        sys.exit()

    startdate = "2021-02-01"   # 设置起止时间
    enddate = "2021-02-28"
    stockData = getStockData(stockcode, startdate, enddate, 'd')  # 获取股票信息
    fig = plt.figure(figsize=(12, 7), dpi=100)  # 绘制K线
    left, width, bottom, top = 0.05, 0.9, 0.10, 0.80
    ax1 = fig.add_axes([left, bottom, width, top], gid='mygroup')
    title = "股票代码"+stockcode[3:]+"("+startdate+"~"+enddate+")日K线图"  # 设置图表标题
    candle_draw(stockData, fig, ax1, title)
    aa = fig.canvas.manager.toolbar
    fig.canvas.set_window_title('K线图')
    plt.show()
```

（二）实验二：绘制资产与收益率曲线，DisplayAssetProfit.py

1、利用 import 引入程序所需要的库。实验二中需要用到的库有 baostock, pandas, mplfinance, matplotlib, tkinter。

```python
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import mplfinance as mpf
import baostock as bs
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
```

2、设置绘图的参数，其中下图第一行设置中文字体为“Arial Unicode MS”，第二行设置负号显示，第三行设置曲线的宽度。由于本实验操作基于 macOS 系统，因此当本程序在 Windows 系统上运行时可能会出现乱码，无法显示中文字体，此时应当更换中文字体的样式。

```python
mpl.rcParams['font.sans-serif'] = ['Arial Unicode MS']  # 设置字体样式
mpl.rcParams['axes.unicode_minus'] = False  # 设置负号显示
mpl.rcParams['lines.linewidth'] = 1.2  # 设置曲线宽度
```

3、定义一个函数 getStockData()。该函数输入为股票代码、起止时间、数据类型，随后该函数进行数据爬取获得股票的日期、开收盘价、最高最低价，成交量，复权状态，换手率等信息。然后计算资产和收益率等，最后进行相关整理后返回一个 DataFrame 的数据结构。

```python
def getStockData(stockcode, startdate, enddate, periodtype):
    bs.login()  # 登陆系统
    rs = bs.query_history_k_data_plus(stockcode, 'date, code, open, high, low, close, volume, amount, adjustflag, turn, pctChg',
                                      start_date=startdate,
                                      end_date=enddate,
                                      frequency=periodtype,
                                      adjustflag='3')
    # 获取对应股票的信息，volume为成交量，amount为成交额，adjustflag为复权状态，turn为换手率，pctChg为涨跌幅
    data_list = []
    while (rs.error_code == '0') & rs.next():  # 获取一条记录，并将多条记录整合在一起
        data_list.append(rs.get_row_data())
    capital = 0                                # 计算资产和收益率
    for i in range(len(data_list)):
        data = data_list[i]
        close = float(data[5])
        capital = capital + close*100
        tvalue = (close*(i+1)*100)
        profit = (tvalue/capital-1)*100
        data.append(capital)
        data.append(tvalue)
        data.append(profit)
    df = pd.DataFrame(data_list, columns=['date', 'code', 'open', 'high', 'low', 'close',
                                          'volume', 'amount', 'adjustflag', 'turn', 'pctChg',
                                          'capital', 'mktValue', 'profitRatio'])
    # 创建一个DataFrame数据结构
    df['open'] = df['open'].astype('float')  # 将获得的数据转换为浮点数类型
    df['high'] = df['high'].astype('float')
    df['low'] = df['low'].astype('float')
    df['close'] = df['close'].astype('float')
    df['volume'] = df['volume'].astype('float')
    df['date'] = pd.to_datetime(df['date'])  # 将date列数据类型转换为时间类型
    df.set_index(['date'], inplace=True)  # 将日期列作为行索引
    bs.logout()
    return df
```

4、定义一个函数 main()。该函数的功能为绘制资产和收益率曲线。该函数的输入为股票代码，已经计算并处理好的对应股票的相关信息，以及起止时间。首先设置好将要展示的图形界面，包括窗口大小、图标大小、曲线标题与分辨率等。然后绘制曲线，资产曲线为红色，收益率曲线为蓝色，且二者共享一个横轴。最后设计一个退出按钮，当点击“Quit”键后，程序退出。

```python
def main(stockcode, asset_info, s_dateStr, e_dateStr):
    linewidth_bak = mpl.rcParams['lines.linewidth']
    root = tk.Tk()
    root.geometry('1200x450+200+200')  # 设置窗口的大小与位置
    titlestr = stockcode + '资产与收益率曲线（' + s_dateStr + '~' + e_dateStr + '）'  # 设置曲线标题
    root.title(titlestr)
    fig = plt.figure(figsize=(8, 4), dpi=100)  # 设置图表的大小与分辨率
    left, width = 0.05, 0.90
    ax1 = fig.add_axes([left, 0.26, width, 0.70], gid='mygroup')
    ax2 = fig.add_axes([left, 0.05, width, 0.25], sharex=ax1, gid='mygroup')  # 共享ax1轴
    ax1.yaxis.grid(True, which='major')
    ax2.yaxis.grid(True, which='major')
    add_plot = [mpf.make_addplot(asset_info['mktValue'], color='red', width=0.5, ax=ax1),
                mpf.make_addplot(asset_info['profitRatio'], color='blue', width=1.2, ax=ax2)]
    mpf.plot(asset_info, ax=ax1, addplot=add_plot)
    canvas = FigureCanvasTkAgg(fig, master=root)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

    def QuitButton():  # 设计一个GUI交互界面，界面中点击"Quit"退出程序
        root.quit()
    button = tk.Button(root, text='Quit', command=QuitButton)
    button.pack(side=tk.BOTTOM)
    root.focus_force()
    root.mainloop()
    root.destroy()
    mpl.rcParams['lines.linewidth'] = linewidth_bak
```

5、通过程序入口 if __name__ == '__main__'，开始执行整个程序。主程序中首先设置要投资股票的代码（实验中选取代码 sh.600215）和投资起止时间（2021-02-01 到 2021-02-10），随后调用函数 getStockData()获得对应股票的信息。然后计算出该目标数据，分别打印出来。最后调用函数 main()画出总资产与收益率曲线。

```
if __name__ == '__main__':
    stockcode = 'sh.600215'  # 设置要处理的股票代码
    s_dateStr = '2021-02-01'  # 设置投资的起止时间
    e_dateStr = '2021-02-10'
    stockData_day = getStockData(stockcode, s_dateStr, e_dateStr, 'd')  # 获取对应股票的数据
    totalCapital = stockData_day['capital'][-1]  # 计算总本金投入
    totalmktValue = stockData_day['mktValue'][-1]  # 计算总资产
    totalProfit = totalmktValue-totalCapital  # 计算总收益
    profitRatio = stockData_day['profitRatio'][-1]  # 计算收益率
    maxloseRatio = stockData_day['profitRatio'].min()  # 计算最大回撤率
    if maxloseRatio > 0:
        maxloseRatio = 0
    print('总本金投入：{}'.format(totalCapital))  # 打印出数据
    print('期末总资产：{}'.format(totalmktValue))
    print('期末总收益：{}'.format(totalProfit))
    print('收益率：{:.5}%'.format(profitRatio))
    print('最大回撤率：{:.5}%'.format(maxloseRatio))
    main(stockcode, stockData_day, s_dateStr, e_dateStr)  # 画出资产与收益率曲线
```

（三）实验三：设计 GUI 界面自动爬取股票信息，InputStockInfo.py，GUI_ShowKLine.py

1、本实验由两个程序组成，首先编写 InputStockInfo.py 程序，其功能是设计一个 GUI 界面获取用户输入的信息。利用 import 引入程序所需要的库，并定义最终需要返回的数据 stockInfo。InputStockInfo.py 所用到的库为 tkinter。

```
import tkinter as tk
from tkinter import ttk
stockInfo = ['', '', '', '']
```

2、定义 InputStockInfo.py 中的 main()函数。该函数的第一部分主要为设计 GUI 界面的输入框。首先设置界面的标题和大小，然后分别设置输入股票代码、开始时间、截止时间的输入框，分别设置输入框的大小与位置。最后结合 tk.StringVar()与 ttk.Combobox()两个组件设置一个可供用户选择的组合框，该组合框中的可选项分别为“日 K 线”、“周 K 线”以及“月 K 线”，可以供用户选择不同的 K 线图种类。

```
def main():
    root = tk.Tk()
    root.title('输入股票代码和起止时间')  # 界面的标题
    root.geometry('470x200')  # 界面大小
    label1 = tk.Label(root, text='股票代码(如:600215):')  # 设置输入框
    label1.place(x=20, y=5, width=200, height=25)
    editbox1 = tk.Entry(root)
    editbox1.place(x=225, y=5, width=200, height=25)
    label2 = tk.Label(root, text='开始时间(如2020-03-01):')
    label2.place(x=20, y=45, width=200, height=25)
    editbox2 = tk.Entry(root)
    editbox2.place(x=225, y=45, width=200, height=25)
    label3 = tk.Label(root, text='截止时间(如2021-03-01):')
    label3.place(x=20, y=85, width=200, height=25)
    editbox3 = tk.Entry(root)
    editbox3.place(x=225, y=85, width=200, height=25)
    label4 = tk.Label(root, text='K线类型:')
    label4.place(x=20, y=125, width=200, height=25)
    comvalue = tk.StringVar()  # 可以随时根据用户的选择更新窗体中的值
    comboxlist = ttk.Combobox(root, textvariable=comvalue)  # 创建一组合框，可选择K线种类
    comboxlist['values'] = ('日K线', '周K线', '月K线')  # 设置组合框中的值
    comboxlist.place(x=225, y=125, width=200, height=25)
```

3、main()函数的第二部分是定义函数 myConfirm()以及 myCancel()。myConfirm()函数在 GUI 界面中“确定”被点击后被执行，可以获取输入框中用户输入的内容并将其转换后填充进 stockInfo 中，而且在填充时将最后一个输入框中“日 K 线”、“周 K 线”、“月 K 线”分别转换为“day”、“week”、“month”。myCancel()函数在 GUI 界面中“取消”被点击后被执行，直接退出 GUI 界面

```
    def myConfirm():  # GUI界面中点击"确定"后执行，获取将输入框中的数据填充进stockInfo中
        global stockInfo
        var1 = editbox1.get()
        var2 = editbox2.get()
        var3 = editbox3.get()
        var4 = comboxlist.get()
        if var1 != '' and var2 != '' and var3 != '' and var4 != '':
            if var4 == '日K线':
                var4 = 'day'
            elif var4 == '周K线':
                var4 = 'week'
            elif var4 == '月K线':
                var4 = 'month'
            stockInfo = [var1, var2, var3, var4]
            root.quit()

    def myCancel():  # GUI界面中点击"取消"后执行，直接退出GUI界面
        global stockInfo
        stockInfo = ['', '', '']
        root.quit()
```

4、main()函数的第三部分是设计“确定”与“取消”按钮。该部分利用 tk.Button()分别设计“确定”与“取消”按钮。点击“确定”后执行 myConfirm()，点击“取消”后执行 myCancel()。

```
    button1 = tk.Button(root, text='确定', command=myConfirm)  # 设计"确定"按钮
    button1.place(x=95, y=165, width=40, height=25)
    button2 = tk.Button(root, text='取消', command=myCancel)  # 设计"取消"按钮
    button2.place(x=330, y=165, width=40, height=25)
    root.focus_force()
    editbox1.focus_force()
    root.mainloop()
    root.destroy()
    return stockInfo
```

5、然后编写 GUI_ShowKLine.py 程序。利用 import 引入程序所需要的库，GUI_ShowKLine.py 中需要用到的库有 baostock，pandas，mplfinance，matplotlib，tkinter，同时还需要引入 InputStockInfo 以调用其中的 main()函数。

```
import baostock as bs
import pandas as pd
import mplfinance as mpf
import matplotlib as mpl
import matplotlib.pyplot as plt
import InputStockInfo
```

6、设置绘图的参数，其中下图第一行设置中文字体为“Arial Unicode MS”，第二行设置负号显示。由于本实验操作基于 macOS 系统，因此当本程序在 Windows 系统上运行时可能会出现乱码，无法显示中文字体，此时应当更换中文字体的样式。

```
mpl.rcParams['font.sans-serif'] = ['Arial Unicode MS']
mpl.rcParams['axes.unicode_minus'] = False
```

7、定义一个函数 getStockData(),用于获取股票的信息，由于该函数的编写与实验一和实验二中的 getStockData()编写方式相似，在此便不再赘述，代码如下。

```python
def getStockData(stockcode, startdate, enddate, periodtype):  # 用于获取股票信息并返回一个含有股票信息的DataFrame数据结构
    bs.login()
    rs = bs.query_history_k_data_plus(stockcode, 'date, code, open, high, low, close',  # 获取股票信息
                                      start_date=startdate,
                                      end_date=enddate,
                                      frequency=periodtype)
    data_list = []
    while (rs.error_code == '0') & rs.next():  # 获取一条记录，并将多条记录整合在一起
        data_list.append(rs.get_row_data())
    df = pd.DataFrame(data_list, columns=['date', 'code', 'open', 'high', 'low', 'close'])  # 创建一个DataFrame数据结构
    df['open'] = df['open'].astype('float')  # 将获得的数据转换为浮点数类型
    df['high'] = df['high'].astype('float')
    df['low'] = df['low'].astype('float')
    df['close'] = df['close'].astype('float')
    df['date'] = pd.to_datetime(df['date'])  # 将date列数据类型转换为时间类型
    df.set_index(['date'], inplace=True)  # 将日期列作为行索引
    bs.logout()
    return df
```

8、定义一个函数 candle_draw()，用于绘制 K 线图。由于该函数的编写与实验一中的 candle_draw()编写方式相似，在此便不再赘述，代码如下。

```python
def candle_draw(data, fig1, ax1, title1):  # 画出K线图
    mc = mpf.make_marketcolors(up='red', down='green', edge='i', wick='i')
    s = mpf.make_mpf_style(gridaxis='both', gridstyle='-.', y_on_right=False, marketcolors=mc)
    mpl.rcParams['lines.linewidth'] = 0.5
    font1 = {'family': 'Arial Unicode MS', 'weight': 'bold', 'size': 10, }
    ax1.set_title(title1, font1)
    kwargs = dict(type='candle', )
    mpf.plot(data, **kwargs, tight_layout=True, style=s, ax=ax1)
    fig1.canvas.draw_idle()
```

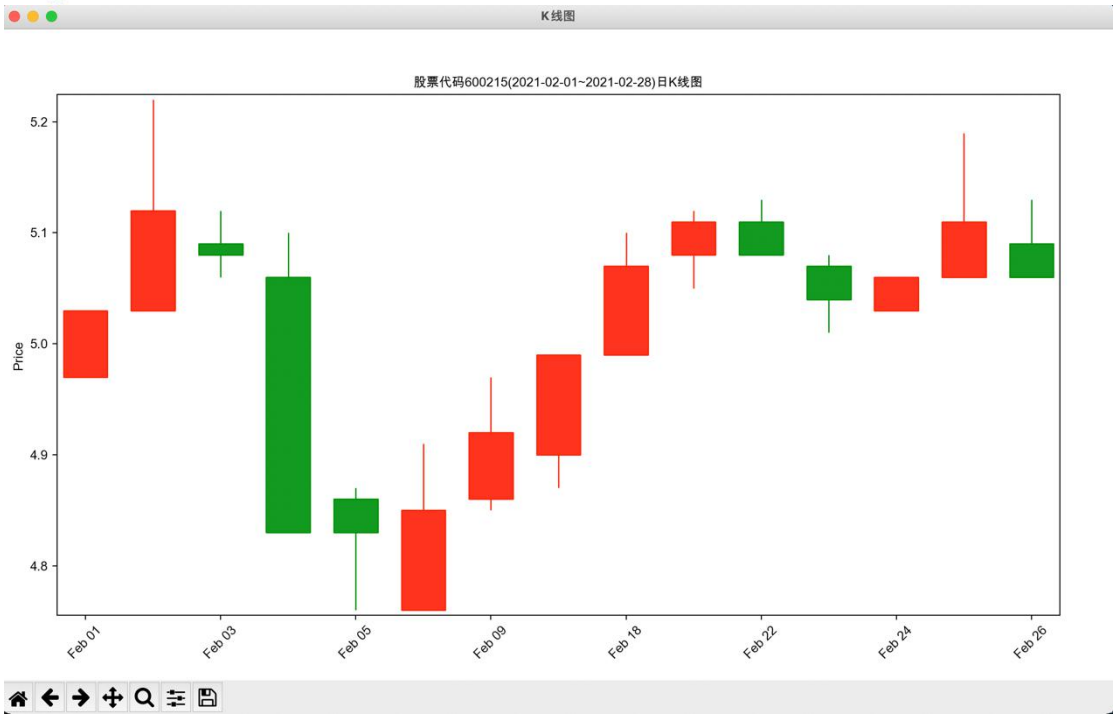
9、通过程序入口 if __name__ == ‘__main__’，开始执行整个程序。首先通过 InputStockInfo.main()调用 InputStockInfo.py 中的 main()函数，获得用户输入的相关信息。然后对这些信息分别进行处理，补全股票代码，获得开始与截止时间以及 K 线种类。最后画出 K 线图。

```python
if __name__ == '__main__':
    rslt = InputStockInfo.main()  # 执行InputStockInfo.py中的main()
    if rslt[0] != '':
        if rslt[0].startswith('6'):  # 将股票代码补充完整
            stockcode = 'sh.'+rslt[0]
        else:
            stockcode = 'sz.'+rslt[0]
        startdate = rslt[1]   # 获得开始时间
        enddate = rslt[2]     # 获得截止时间
        periodtype = rslt[3]  # 获得K线种类
        if periodtype == 'day':
            StockData = getStockData(stockcode, startdate, enddate, 'd')
            title1 = '股票代码：' + stockcode[3:] + '(' + startdate + '~' + enddate + ')日K线图'
        elif periodtype == 'week':
            StockData = getStockData(stockcode, startdate, enddate, 'w')
            title1 = '股票代码：' + stockcode[3:] + '(' + startdate + '~' + enddate + ')周K线图'
        else:
            StockData = getStockData(stockcode, startdate, enddate, 'm')
            title1 = '股票代码：' + stockcode[3:] + '(' + startdate + '~' + enddate + ')月K线图'
    fig = plt.figure(figsize=(10, 7), dpi=100)
    left, width, bottom, top = 0.05, 0.9, 0.10, 0.80
    ax1 = fig.add_axes([left, bottom, width, top], gid='mygroup')
    candle_draw(StockData, fig, ax1, title1)
    fig.canvas.set_window_title('K线图')
    plt.show()
```

五、实验结果

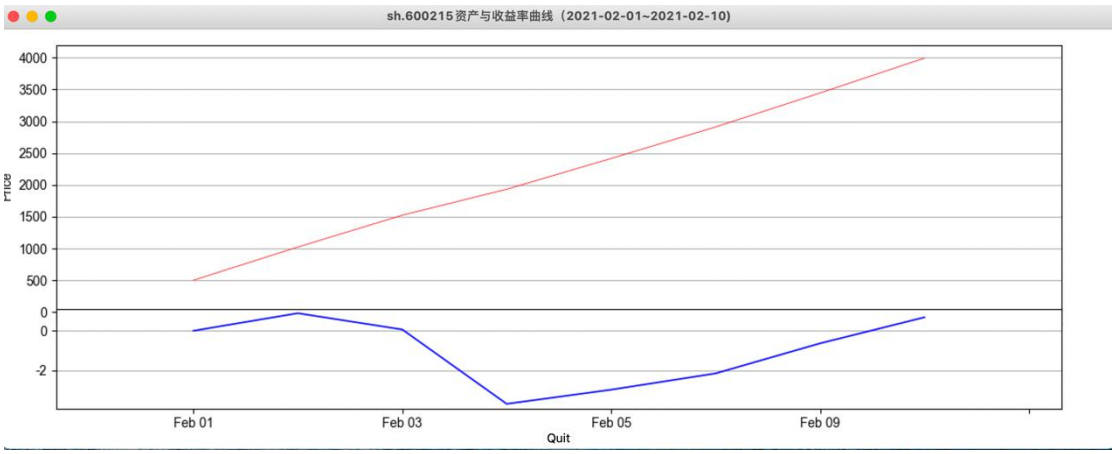
（一）实验一：运行 DisplayKLine_StuNum.py

```
请输入学号的最后三位：215
login success!
logout success!
```

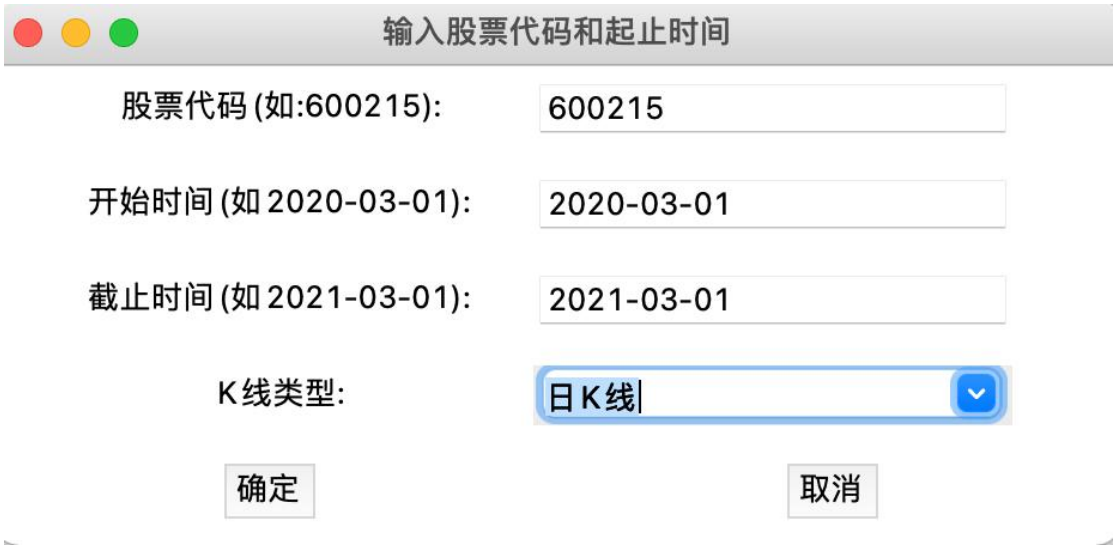


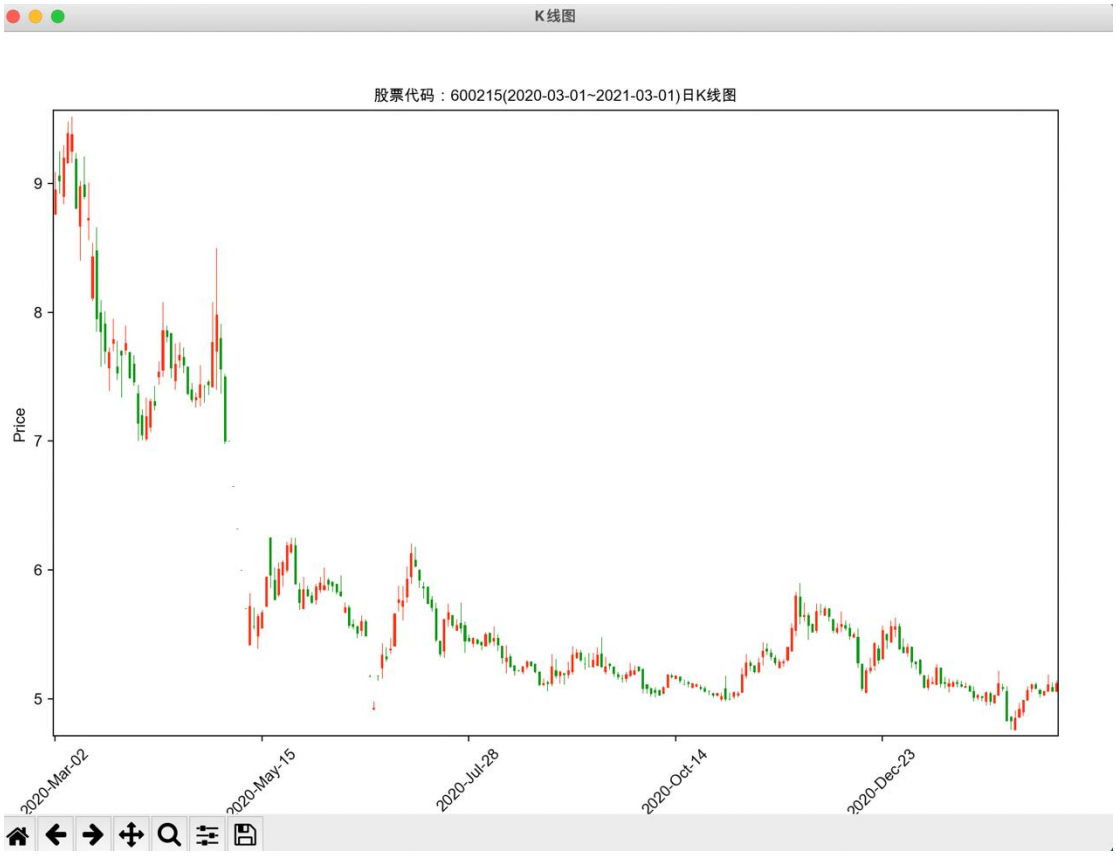
（二）实验二：运行 DisplayAssetProfit.py

```
login success!
logout success!
总本金投入：3965.0
期末总资产：3992.0
期末总收益：27.0
收益率：0.68096%
最大回撤率：-3.6889%
```



（三）实验三：运行 GUI_ShowKLine.py





六、实验小结

首先，通过这一次实验，我对 Python 程序设计这一门技术掌握得更加牢固，同时对 matplotlib、mplfinance、pandas 以及 tkinter 这几个常用的 Python 库有了更深刻的理解，并学会了初步运用这些库；其次，这次实验让我对数据可视化也有了更深刻的理解；最后，通过这一次编程，我自己的各方面能力都有了较大的提高，而且学会了将编程应用于生活生产中，这让我受益匪浅。