****

**Python数据处理编程实验报告**

班级：计科2105

学号：8202191123

姓名：谭哲文

2023年5月

计算机学院

实验1：网络爬虫开发及数据的可视化

# 实验目标与基本要求

### 实验目标

开发网络爬虫在东方财富、新浪财经或者纳斯达克等财经网站上爬取一只股票的每天的开盘价，收盘价，最高价，最低价等信息，并存储在数据库中，并开发GUI应用可视化。

### 基本要求

1. 掌握网络爬虫的开发方法；
2. 掌握Python开发数据库的GUI界面；
3. 掌握Matplotlib绘制股票的K线图；

# 主要知识点、重点与难点

### 主要知识点

1. 网络爬虫的基本知识；
2. 利用正则表达式对网页信息提取；
3. 数据库的访问和表中的数据操作；
4. Matplotlib库的使用。

### 重点

1. 网络爬虫框架的使用；
2. 正则表达式的使用；
3. 数据库存储数据。

### 难点

1. 利用正则表达式根据网页中的信息组织方式提取数据；
2. K线图的展现。

# 实验过程设计

1 .确定网站：搜狐证券

根据实验要求，本次爬虫实验并不限定爬虫网站，主要是为了锻炼学生的爬虫能力.我选择了搜获证券进行爬虫。

网址：<https://q.stock.sohu.com/cn>



如上图，为该网站的首页。

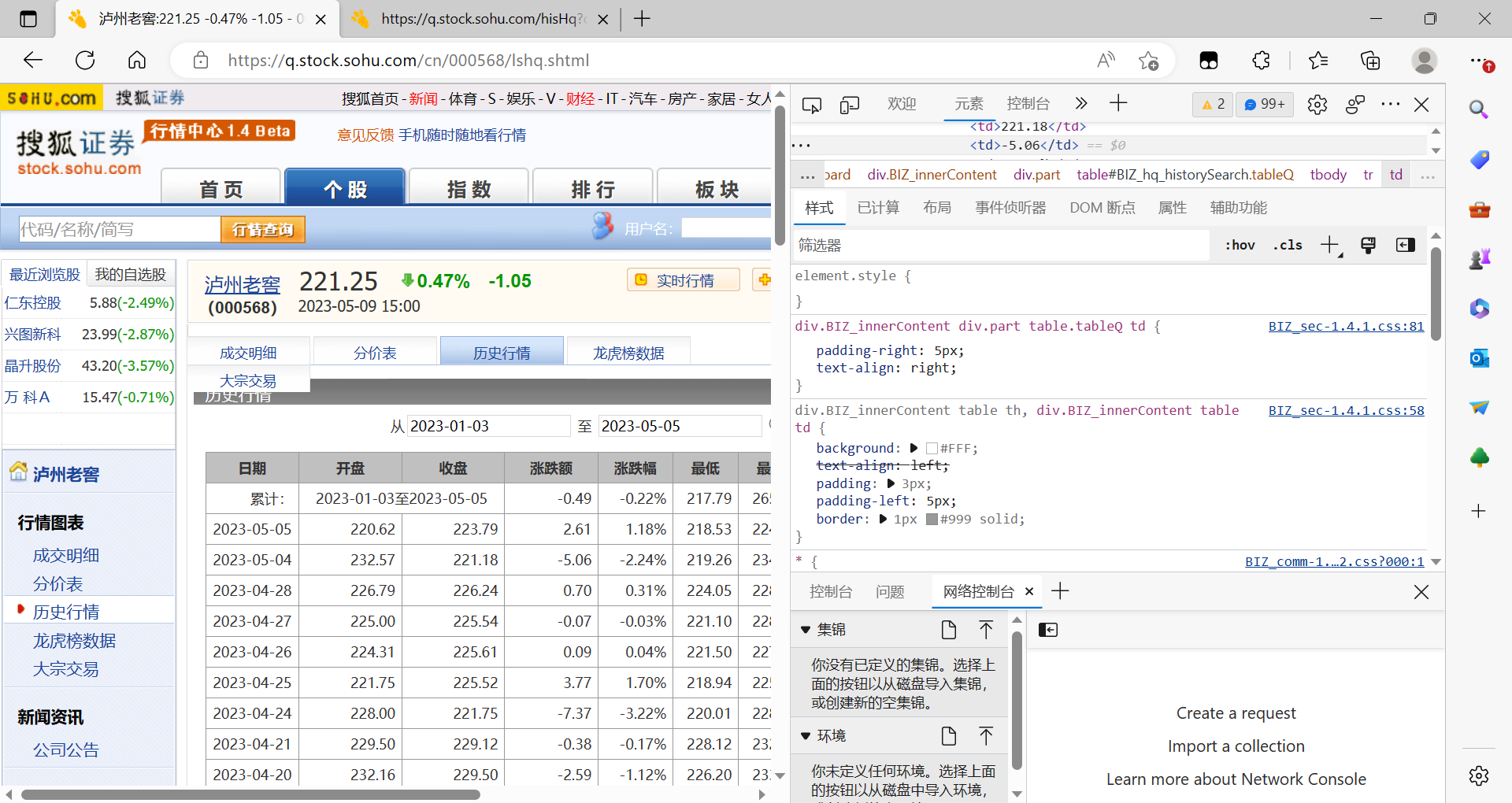
1.2 查看爬取元素的代码

选择左上方的泸州老窖，再在分类导航一栏中选择历史行情。

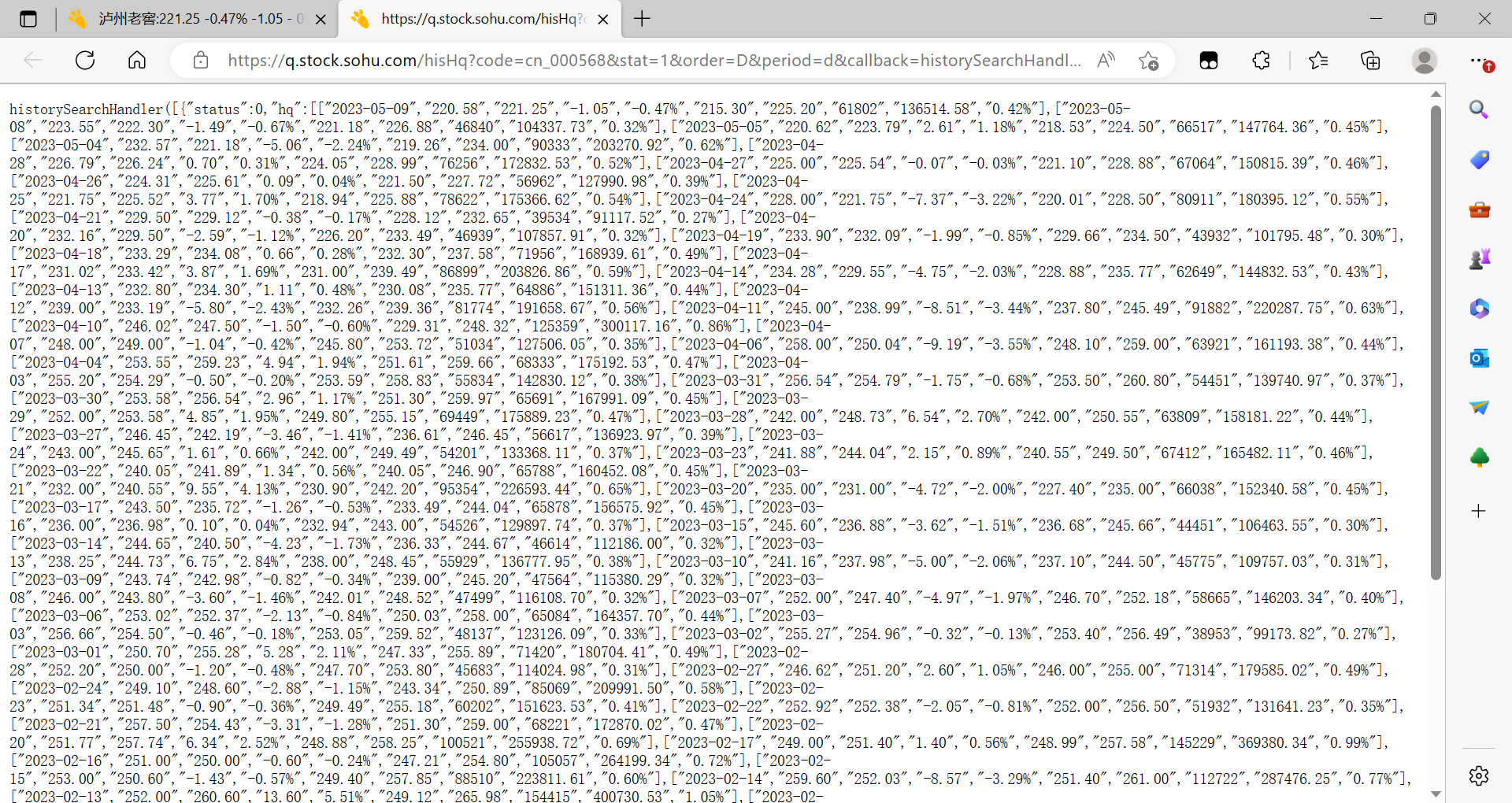
点击后的跳转页面如下图所示。



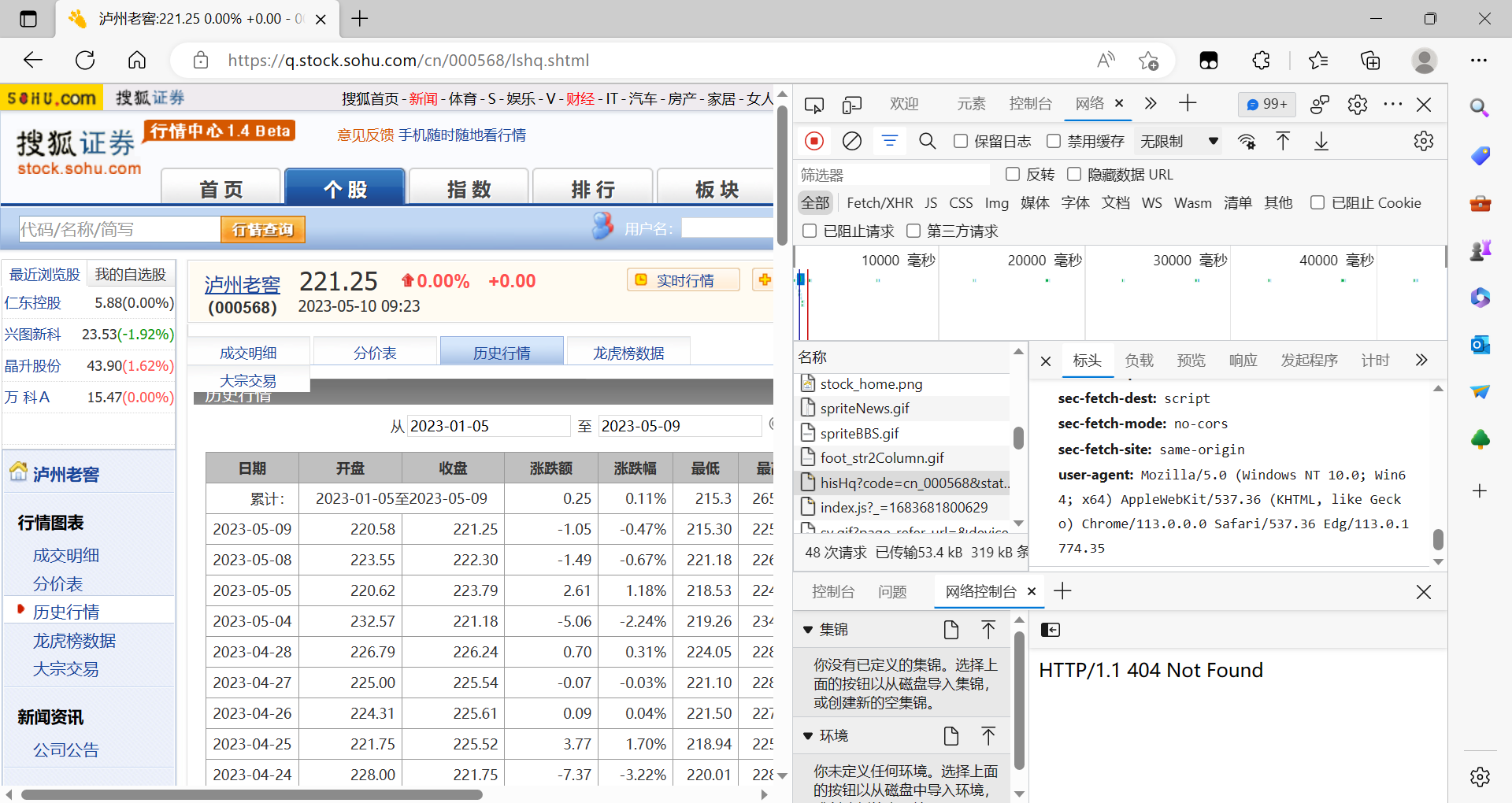
右键点击检查，点击网络，找到hisHq.



<https://q.stock.sohu.com/hisHq?code=cn_000568&stat=1&order=D&period=d&callback=historySearchHandler&rt=jsonp&0.7380906540612318>



1. 爬取网页

伪装成用户进行爬取，通过查看网络响应，找到其user-agent通过headers进行访问。

cookies = {

    'IPLOC': 'CN1200',

    'SUV': '230427083048JSFW',

    'clt': '1682555456',

    'cld': '20230427083056',

    'reqtype': 'pc',

    'gidinf': 'x099980109ee16eb87f14cc490001fc46aa6f8cd1299',

    'BIZ\_MyLBS': 'cn\_000568%2C%u6CF8%u5DDE%u8001%u7A96%7Ccn\_002647%2C%u4EC1%u4E1C%u63A7%u80A1%7Ccn\_688081%2C%u5174%u56FE%u65B0%u79D1%7Ccn\_688478%2C%u6676%u5347%u80A1%u4EFD%7Ccn\_000002%2C%u4E07%u79D1%uFF21',

    't': '1683170711989',

}

headers = {

    'authority': 'q.stock.sohu.com',

    'accept': '\*/\*',

    'accept-language': 'zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6',

    # 'cookie': 'IPLOC=CN1200; SUV=230427083048JSFW; clt=1682555456; cld=20230427083056; reqtype=pc; gidinf=x099980109ee16eb87f14cc490001fc46aa6f8cd1299; BIZ\_MyLBS=cn\_000568%2C%u6CF8%u5DDE%u8001%u7A96%7Ccn\_002647%2C%u4EC1%u4E1C%u63A7%u80A1%7Ccn\_688081%2C%u5174%u56FE%u65B0%u79D1%7Ccn\_688478%2C%u6676%u5347%u80A1%u4EFD%7Ccn\_000002%2C%u4E07%u79D1%uFF21; t=1683170711989',

    'referer': 'https://q.stock.sohu.com/cn/000568/lshq.shtml',

    'sec-ch-ua': '"Chromium";v="112", "Microsoft Edge";v="112", "Not:A-Brand";v="99"',

    'sec-ch-ua-mobile': '?0',

    'sec-ch-ua-platform': '"Windows"',

    'sec-fetch-dest': 'script',

    'sec-fetch-mode': 'no-cors',

    'sec-fetch-site': 'same-origin',

    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36 Edg/112.0.1722.58',

}

1. 获取数据：

response = requests.get(

    'https://q.stock.sohu.com/hisHq?code=cn\_000568&stat=1&order=D&period=d&0.25251595627443435',

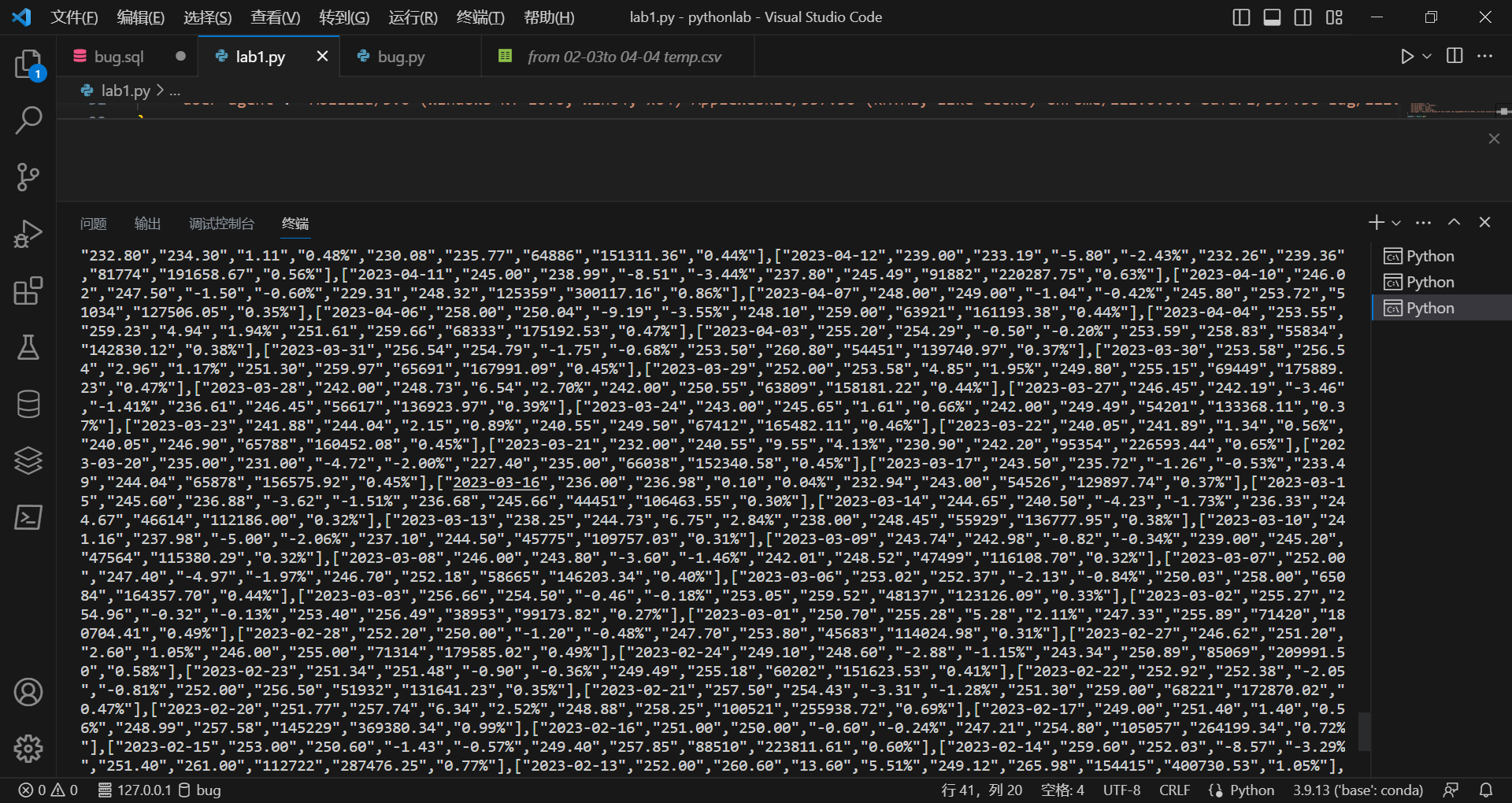
    cookies=cookies,

    headers=headers,

)

print(response.text)

print(type(response.text))

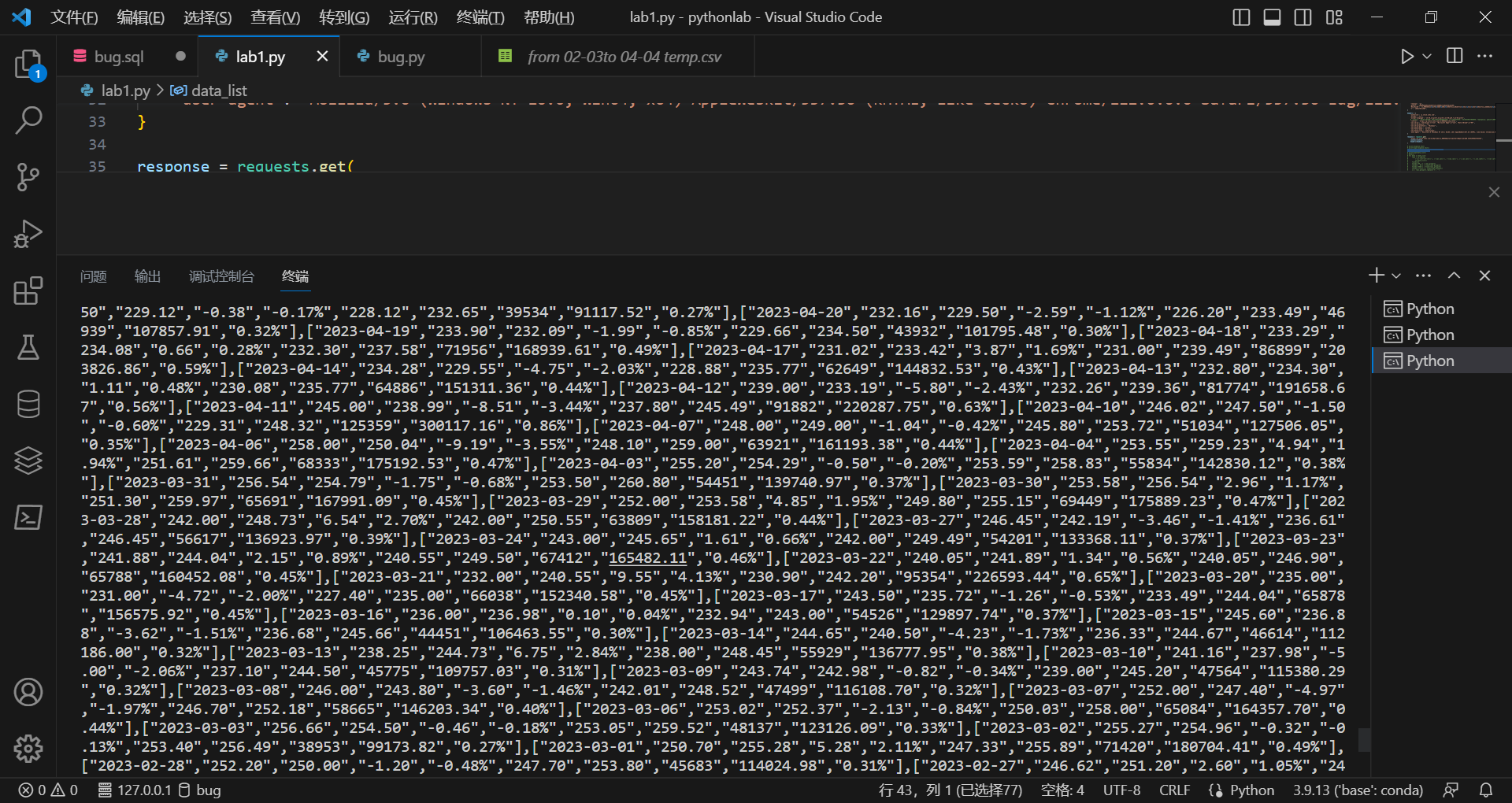


通过查询response.text的type，发现其为str类型，用json.loads将其转化为列表型，因为其有其他赘余部分，我们用

data\_list = json.loads(response.text[18:-121])

print(response.text[18:-121])

来获取符合列表部分字符串以符合转化形式。



在获取到html代码后，需要通过正则表达式的处理提取出我们所需要的信息。

data=[]

for data in data\_list:

    rows = re.search(

        r'\[\'(\w+-\w+-\w+)\'\, \'(\w\*.\w\*)\'\, \'(\w\*.\w\*)\'\, \'(.\w\*.\w\*)\'\, \'(.\w\*.\w\*%)\'\, \'(\w+.\w+)\'\, \'(\w+.\w+)\'\, \'(\w+)\'\, \'(\w+.\w+)\'\, \'(\w\*.\w\*%)\'\]',

        str(data\_list))

    fields={}

    fields['日期'] = rows.group(1)

    fields['开盘'] = float(rows.group(2))

    fields['收盘'] = float(rows.group(3))

    fields['涨跌额'] = float(rows.group(4))

    s1 = rows.group(5).replace('%', '')

    fields['涨跌幅'] = float(s1)

    fields['最低'] = float(rows.group(6))

    fields['最高'] = float(rows.group(7))

    fields['成交量'] = int(rows.group(8))

    fields['成交金额'] = float(rows.group(9))

    s2 = rows.group(10).replace('%', '')

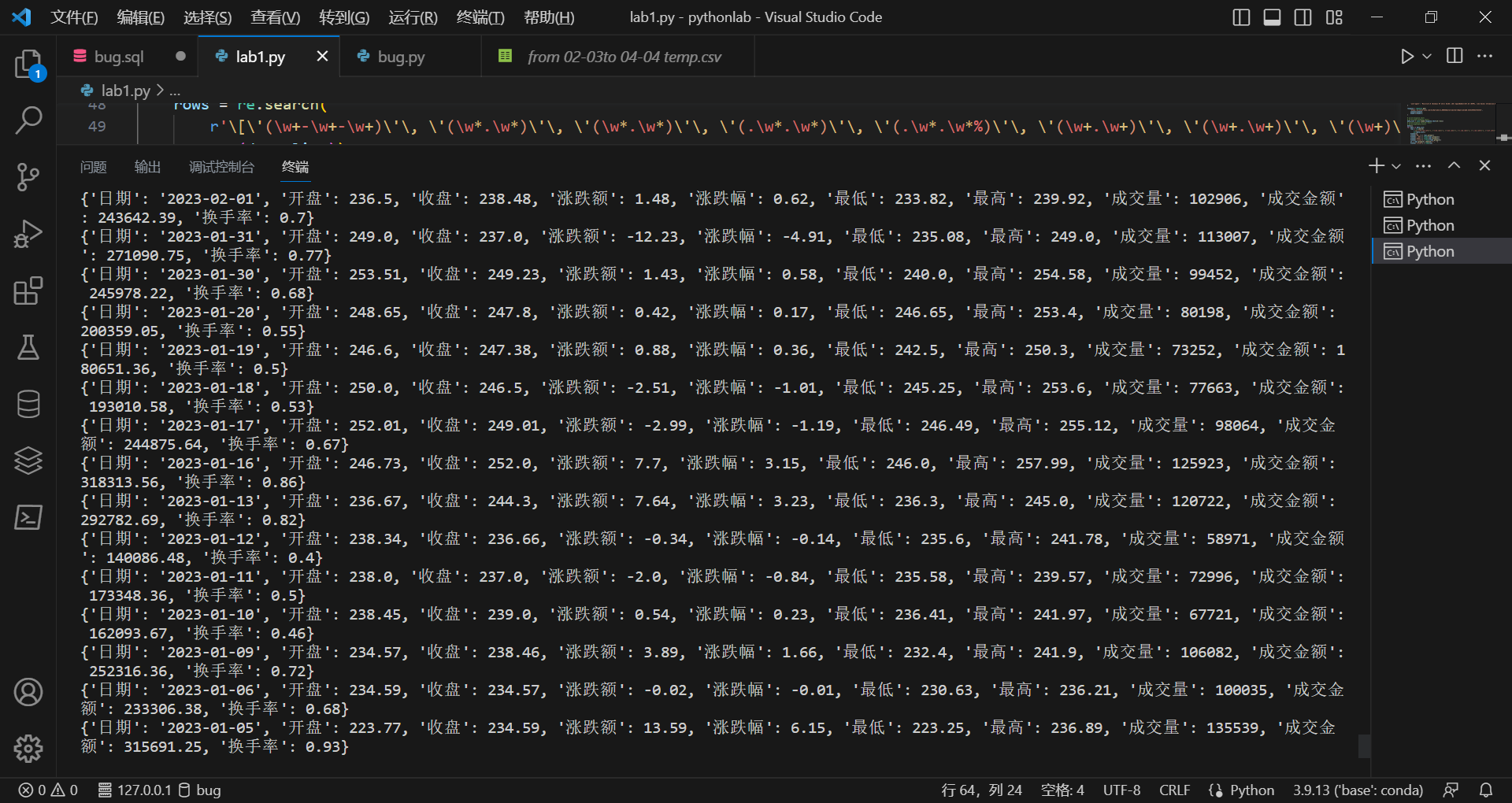
    fields['换手率'] = float(s2)

    data.append(fields)

根据HTML的内容编写正则表达式时，要注意如涨跌额这样的数据是带有“%”的，应该去掉，否则无法转化为浮点数。

fields['涨跌额'] = float(rows.group(4))

s1 = rows.group(5).replace('%', '')

然后将这些数据加入列表，构成字典列表。

（3）数据保存到数据库：

首先在数据库中创建表

-- Active: 1678891157340@@127.0.0.1@3306@bug

create database if not exists bug;    -- 不区分大小写

use bug;

drop table if exists HistoricalQuote;

/\* Table: HistoricalQuote                                       \*/

/\*==============================================================\*/

create table HistoricalQuote

(

   HDate                date not null,

   Open                 decimal(8,4),

   Close                decimal(8,4),

   changeamount         decimal(8,4),

   changerate           decimal(8,4),

   Low                  decimal(8,4),

   High                 decimal(8,4),

   Volume               bigint,

   Volumeamount         decimal(14,4),

   turnoverrate         decimal(8,4),

   primary key (HDate)

);

SELECT \* FROM HistoricalQuote;

为了方便修改和主代码我将函数部分放在了bug.py文件里，后续使用只用导入即可。

from bug import insert\_hisq\_data,  findall\_hisq\_data, findselect\_hisq\_data ,fetchData,pot\_candlestick\_ohlc

以下为将数据插入数据库的代码部分：

def insert\_hisq\_data(row):

    """在股票历史价格表中传入数据"""

    # 1.建立数据库连接

    # 打开数据库连接

    connection = pymysql.connect(host='localhost',

                                 user='root',

                                 password='tzw20010126',

                                 database='bug',

                                 charset='utf8')

    try:

        # 2.创建游标对象

        with connection.cursor() as cursor:

            # 3.执行SQL操作

            sql = 'insert into historicalquote ' \

                  '(HDate,Open,Close,changeamount,changerate,Low,High,Volume,Volumeamount,turnoverrate)' \

                  'values  (%(日期)s,%(开盘)s,%(收盘)s,%(涨跌额)s,%(涨跌幅)s,%(最低)s,%(最高)s,%(成交量)s,%(成交金额)s,%(换手率)s)'

            affectedcount = cursor.execute(sql, row)

            print('影响的数据行数:{0}'.format(affectedcount))

            # 4.提交数据库事务

            connection.commit()

        # with 代码块结束  5.关闭游标

    except pymysql.DatabaseError as error:

        # 4.回滚数据库事务

        connection.rollback()

        print(error)

    finally:

        # 6.关闭数据库连接

        connection.close()

在将数据导入数据库中保存后，我们可以借助SQL语言进行数据查询。我们通过select语句选择我们需要的对应股票和对应时间段。

def findselect\_hisq\_data(startmonth, startday, endmonth, endday):

    """在股票代码查询其股票历史数据"""

    # 1.建立数据库连接

    connection = pymysql.connect(host='localhost',

                                 user='root',

                                 password='tzw20010126',

                                 database='bug',

                                 charset='utf8')

    # 要返回的数据

    data = []

    starttime = '2023-'+startmonth+'-'+startday

    print('starttime'+starttime)

    endtime = '2023-'+endmonth+'-'+endday

    print('endtime'+endtime)

    try:

        # 2.创建游标对象

        with connection.cursor() as cursor:

            # # 3.执行SQL操作

            sql = """select ' \

                  'HDate, Open, Close, changeamount, changerate, Low, High, Volume, Volumeamount, turnoverrate ' \

                  'from historicalquote where  HDate between '%s' and '%s' """\

                  % (starttime, endtime)  # 注意要加引号，不然无法查询数据

            cursor.execute(sql)

            # 3.执行SQL操作

            sql = 'select ' \

                  'HDate, Open, Close, changeamount, changerate, Low, High, Volume, Volumeamount, turnoverrate ' \

                  'from historicalquote where   hdate between %s and %s'

            cursor.execute(sql, [starttime, endtime])

            # 4.提取结果集

            result\_set = cursor.fetchall()

            # print(result\_set)

            for row in result\_set:

                fields = {}

                fields['日期'] = str(row[0])

                fields['开盘'] = float(row[1])

                fields['收盘'] = float(row[2])

                fields['涨跌额'] = float(row[3])

                fields['涨跌幅'] = float(row[4])

                fields['最低'] = float(row[5])

                fields['最高'] = float(row[6])

                fields['成交量'] = float(row[7])

                fields['成交金额'] = float(row[8])

                fields['换手率'] = float(row[9])

                data.append(fields)

        print(data)

        # with 代码块结束  5.关闭游标

    except pymysql.DatabaseError as error:

        print('数据查询失败' + error)

    finally:

        # 6.关闭数据库连接

        connection.close()

    # 返回字典列表

    return data

def findall\_hisq\_data():

    # 1.建立数据库连接

    connection = pymysql.connect(host='localhost',

                                 user='root',

                                 password='tzw20010126',

                                 database='bug',

                                 charset='utf8')

    # 要返回的数据

    data = []

    try:

        # 2.创建游标对象

        with connection.cursor() as cursor:

            # 3.执行SQL操作

            sql = """

                  select

                  HDate, Open, Close, changeamount, changerate, Low, High, Volume, Volumeamount, turnoverrate

                  from historicalquote

                  """

            cursor.execute(sql)

            # 4.提取结果集

            result\_set = cursor.fetchall()

            print("查询成功")

            for row in result\_set:

                fields = {}

                fields['日期'] = row[0]

                fields['开盘'] = float(row[1])

                fields['收盘'] = float(row[2])

                fields['涨跌额'] = float(row[3])

                fields['涨跌幅'] = float(row[4])

                fields['最低'] = float(row[5])

                fields['最高'] = float(row[6])

                fields['成交量'] = float(row[7])

                fields['成交金额'] = float(row[8])

                fields['换手率'] = float(row[9])

                data.append(fields)

        print(data)

        # with 代码块结束  5.关闭游标

    except pymysql.DatabaseError as error:

        print('数据查询失败' + error)

    finally:

        # 6.关闭数据库连接

        connection.close()

    # 返回字典列表

    return data

使用fetchData（）函数查询到的数据保存至文件中，方便后续使用pandas库读取。

def fetchData(startmonth, startday, endmonth, endday):

    data = findselect\_hisq\_data( startmonth, startday, endmonth, endday)

    print('data='+str(data))

    # 列名

    colsname = ['日期', '开盘', '最高', '最低', '收盘', '涨跌额', '成交量', '成交金额']

    # 临时数据文件名

    datafile = '  from '+startmonth+'-'+startday+'to '+endmonth+'-'+endday+' '+'temp.csv'

    # 写入数据到临时数据文件

    with open(datafile, 'w', newline='', encoding='utf-8') as wf:

        writer = csv.writer(wf)

        writer.writerow(colsname)

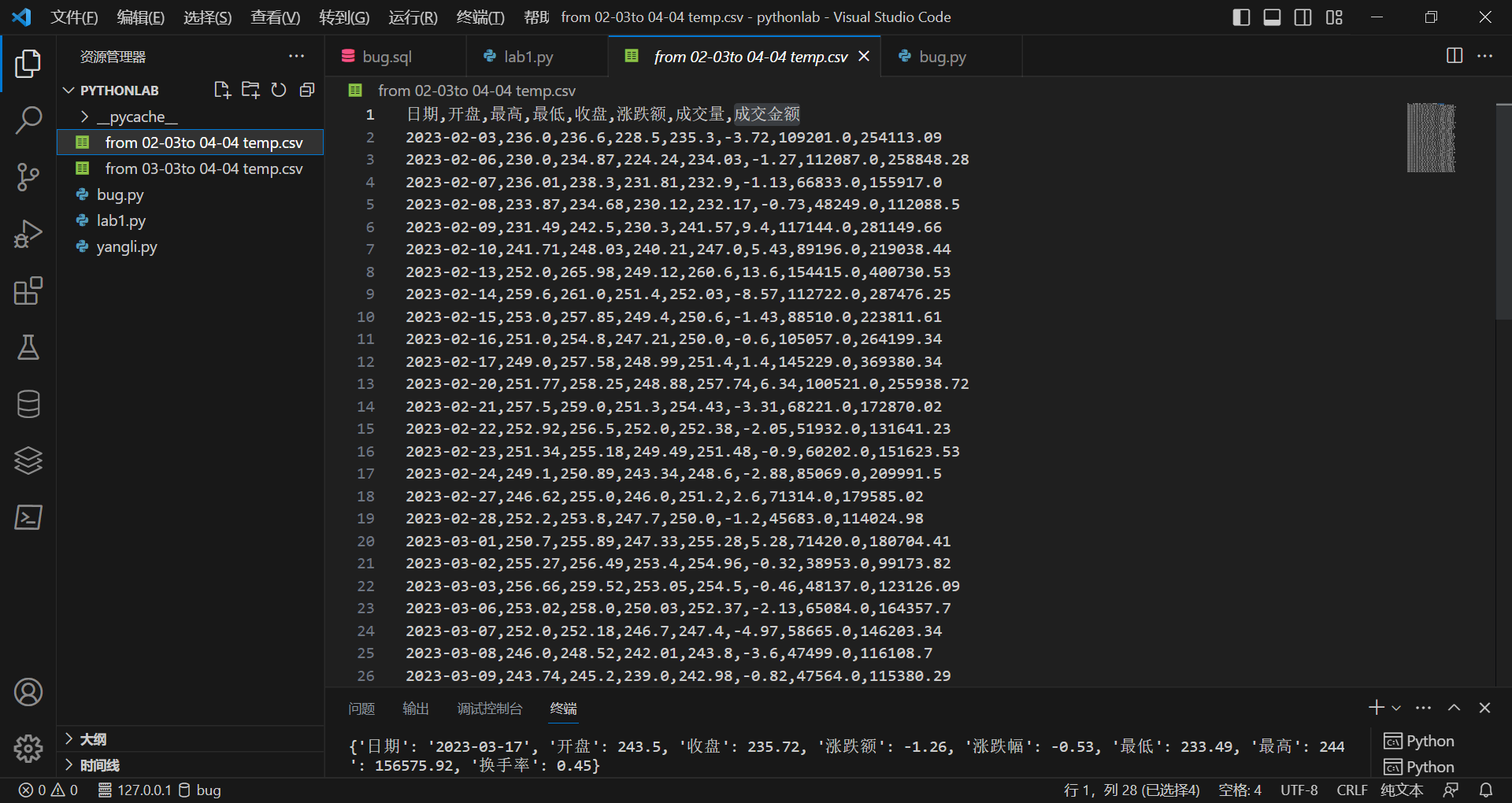
        for quotes in data:

            row = [quotes['日期'], quotes['开盘'], quotes['最高'],

                   quotes['最低'], quotes['收盘'], quotes['涨跌额'], quotes['成交量'], quotes['成交金额']]

            writer.writerow(row)

    return datafile



2.创建图形化界面

2.1设计图形化界面

我设计了三个窗口，第一个窗口是开始界面，询问是否开始，第二个界面则是功能界面，将有4个下拉列表来确定查询的时间范围。一个文本框和2个功能按钮: 第一个是显示表格，展示获取到的数据；第二个则是展示股票在那段时间的阴阳图。

第三个窗口，通过线程切换，关掉主功能窗口，跳转到图片展示窗口。

2.2构建MyFrame类

2.2.1 第一个窗口

class MyFrame(wx.Frame):

    def \_\_init\_\_(self):

        super().\_\_init\_\_(parent=None, title='股票行情查询', size=(300, 400))

        self.Centre()  # 设置窗口居中

        panel = wx.Panel(parent=self)

        fnt1 = wx.Font(pointSize=15, family=wx.ROMAN, style=wx.NORMAL, weight=wx.NORMAL)

        fnt = wx.Font(pointSize=20, family=wx.ROMAN, style=wx.NORMAL, weight=wx.NORMAL)

        # 创建垂直方向的Box布局管理器

        self.sizer = wx.BoxSizer(wx.VERTICAL)

        image0\_sbitmap = wx.StaticBitmap(panel, -1)

        self.statictext = wx.StaticText(parent=panel, label="Let's start?", style=wx.ALIGN\_CENTER\_HORIZONTAL)

        self.statictext.SetFont(fnt)

        b1 = wx.Button(parent=panel, label='Yes')

        # b1 = wx.ToggleButton(panel, -1, 'Yes')

        self.Bind(wx.EVT\_BUTTON, self.on\_click, b1)

        b2 = wx.Button(parent=panel, label='No')

        self.Bind(wx.EVT\_BUTTON, self.on\_click1, b2)

        b1.SetFont(fnt1)

        b2.SetFont(fnt1)

        b3 = wx.BitmapButton(panel, -1)

        self.Bind(wx.EVT\_BUTTON, self.on\_click, b3)

        # 添加静态文本和按钮到Box布局管理器

        self.sizer.Add(100, 10, proportion=1, flag=wx.CENTER | wx.FIXED\_MINSIZE)

        self.sizer.Add(self.statictext, proportion=1, flag=wx.CENTER | wx.FIXED\_MINSIZE)

        self.sizer.Add(image0\_sbitmap, proportion=2, flag=wx.CENTER | wx.FIXED\_MINSIZE)

        self.sizer.Add(b1, proportion=1, flag=wx.CENTER | wx.EXPAND)

        self.sizer.Add(b2, proportion=1, flag=wx.CENTER | wx.EXPAND)

        self.sizer.Add(b3, proportion=3, flag=wx.CENTER | wx.EXPAND)

        panel.SetSizer(self.sizer)

    def on\_click(self, event):

        # 跳转到新界面

        self.Close()

        self.Destroy()

        workthread = threading.Thread(target=workthread\_body(), name='WorkThread')

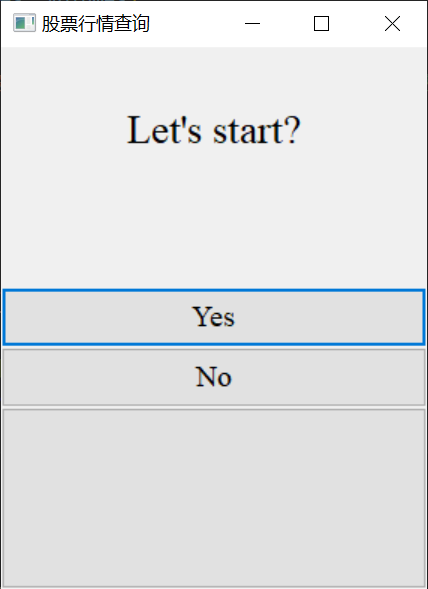
        workthread.start()

    def on\_click1(self, event):

        wx.MessageBox("感谢使用！QWQ")

        self.Close()

        self.Destroy()



2.2.2 第二个窗口

class MyFrame1(wx.Frame):

    def \_\_init\_\_(self):

        super().\_\_init\_\_(parent=None, title='股票行情查询', size=(800, 200))

        self.Centre()  # 设置窗口居中

        self.SetMinSize((750, 310))

        #self.Maximize(True)

        self.panel = wx.Panel(parent=self)

        image0\_sbitmap = wx.StaticBitmap(self.panel, -1)

        # hbox1 = wx.BoxSizer(wx.HORIZONTAL)

        statictext = wx.StaticText(parent=self.panel, label='欢迎使用')

        fnt = wx.Font(pointSize=15, family=wx.ROMAN, style=wx.NORMAL, weight=wx.NORMAL)

        statictext.SetFont(fnt)

        # 预定义变量

        self.text = ['', '', '', '']  # 初始月份，初始日期，结束月份，结束日期

        # self.tablename = ['']  # 数据库表名

        self.data = []  # 爬取的数据

        self.savepath = ''

        self.savepath1 = 'his\_bar' + self.savepath

        statictext2 = wx.StaticText(self.panel, label='         开始月份')

        list2 = ['', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']

        self.ch2 = wx.Choice(self.panel, -1, choices=list2)

        self.Bind(wx.EVT\_CHOICE, self.on\_choice2, self.ch2)

        statictext3 = wx.StaticText(self.panel, label='         开始日期')

        list3 = ['', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31']

        self.ch3 =wx.Choice(self.panel, -1, choices=list3)

        self.Bind(wx.EVT\_CHOICE, self.on\_choice3, self.ch3)

        # hbox2.Add(statictext2, 1, flag=wx.LEFT | wx.RIGHT | wx.FIXED\_MINSIZE, border=5)

        # hbox2.Add(ch2, 1, flag=wx.ALL | wx.FIXED\_MINSIZE)

        statictext4 = wx.StaticText(self.panel, label='         结束月份')

        list4 = ['', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']

        self.ch4 = wx.Choice(self.panel, -1, choices=list4)

        self.Bind(wx.EVT\_CHOICE, self.on\_choice4, self.ch4)

        statictext5 = wx.StaticText(self.panel, label='         结束日期')

        list5 = ['', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31']

        self.ch5 = wx.Choice(self.panel, -1, choices=list5)

        self.Bind(wx.EVT\_CHOICE, self.on\_choice5, self.ch5)

        b1 = wx.Button(parent=self.panel, label='查询数据')  # 按钮控件

        b1.Bind(wx.EVT\_BUTTON, self.showdata, b1)  # 绑定事件

        b2 = wx.Button(parent=self.panel, label='展示图表')  # 按钮控件

        b2.Bind(wx.EVT\_BUTTON, self.showfig, b2)  # 绑定事件

        self.showPanel = wx.Panel(parent=self.panel, size=(1200, 800))  # 展示面板

        self.gridpanel = None   #网格面版

        self.kpanel = None      # 图表面版

        # 设置布局

        # 创建水平方向的Box布局管理器对象

        hbox = wx.BoxSizer(wx.HORIZONTAL)

        # 添加控件到水平Box布局管理

        hbox.Add(statictext2, proportion=1, flag=wx.EXPAND)

        hbox.Add(self.ch2, proportion=1, flag=wx.ALIGN\_LEFT)

        hbox.Add(statictext3, proportion=1, flag=wx.EXPAND)

        hbox.Add(self.ch3, proportion=1, flag=wx.ALIGN\_LEFT)

        hbox.Add(statictext4, proportion=1, flag=wx.EXPAND)

        hbox.Add(self.ch4, proportion=1, flag=wx.ALIGN\_LEFT)

        hbox.Add(statictext5, proportion=1, flag=wx.EXPAND)

        hbox.Add(self.ch5, proportion=1, flag=wx.ALIGN\_LEFT)

        hbox1 = wx.BoxSizer(wx.HORIZONTAL)

        hbox1.Add(b1, proportion=1, flag=wx.ALIGN\_CENTER)

        hbox1.Add(b2, proportion=1, flag=wx.ALIGN\_CENTER)

        # 创建垂直方向的Box布局管理对象

        vbox = wx.BoxSizer(wx.VERTICAL)

        # 添加控件到垂直Box布局管理器

        vbox.Add(statictext, proportion=1, flag=wx.ALIGN\_CENTER\_HORIZONTAL)

        vbox.Add(image0\_sbitmap, proportion=2, flag=wx.ALIGN\_CENTER\_HORIZONTAL)

        vbox.Add(hbox, proportion=1, flag=wx.ALIGN\_LEFT)

        vbox.Add(hbox1, proportion=1, flag=wx.ALIGN\_CENTER)

        vbox.Add(self.showPanel, proportion=10, flag=wx.EXPAND)

        self.panel.SetSizer(vbox)



2.2.3第三个窗口

class MyFrame2(wx.Frame):

    def \_\_init\_\_(self,  savapath, savepath1, savepath2, savepath3, savepath4):

        super().\_\_init\_\_(parent=None, title='股票行情图像', pos=(2000, 2000), size=(2500, 1500))

        self.bmps = [wx.Bitmap(savapath, wx.BITMAP\_TYPE\_PNG),

                     wx.Bitmap(savapath, wx.BITMAP\_TYPE\_PNG),

                     wx.Bitmap(savepath2, wx.BITMAP\_TYPE\_PNG),

                     wx.Bitmap(savepath1, wx.BITMAP\_TYPE\_PNG),

                     wx.Bitmap(savepath3, wx.BITMAP\_TYPE\_PNG),

                     wx.Bitmap(savepath4, wx.BITMAP\_TYPE\_PNG)]

        fnt = wx.Font(pointSize=10, family=wx.ROMAN, style=wx.NORMAL, weight=wx.NORMAL)

        self.Maximize(True)

        self.Centre()  # 设置窗口居中

        self.panel = wx.Panel(parent=self)

        # 创建垂直方向的Box布局管理器对象

        hbox = wx.BoxSizer(wx.HORIZONTAL)

        image0\_sbitmap = wx.StaticBitmap(self.panel, -1)

        # hbox1 = wx.BoxSizer(wx.HORIZONTAL)

        statictext = wx.StaticText(parent=self.panel, label='欢迎使用QWQ')

        fnt1 = wx.Font(pointSize=15, family=wx.ROMAN, style=wx.NORMAL, weight=wx.NORMAL)

        statictext.SetFont(fnt1)

        label1 = "OHLC柱状图"

        b1 = wx.Button(parent=self.panel, id=1, label='蜡烛图')

        b12 = wx.Button(parent=self.panel, id=2, label='多日图')

        b13 = wx.Button(parent=self.panel, id=3, label='均线图')

        b2 = wx.Button(self.panel, id=4, label='折线图')

        b3 = wx.Button(self.panel, id=5, label=label1)

        b4 = wx.Button(self.panel, id=6, label='返回')

        self.Bind(wx.EVT\_BUTTON, self.on\_click, id=1, id2=6)

        b1.SetFont(fnt)

        b12.SetFont(fnt)

        b13.SetFont(fnt)

        b2.SetFont(fnt)

        b3.SetFont(fnt)

        b4.SetFont(fnt)

        self.image = wx.StaticBitmap(self.panel, -1, self.bmps[0])

        # 添加标控件到Box布局管理器

        # vbox.Add(self.finnalresult, proportion=1, flag=wx.CENTER)

        # vbox.Add(self.tcfin, proportion=1, flag=wx.CENTER)

        hbox.Add(b1, proportion=1, flag=wx.CENTER)

        hbox.Add(b12, proportion=1, flag=wx.CENTER)

        hbox.Add(b13, proportion=1, flag=wx.CENTER)

        hbox.Add(b2, proportion=1, flag=wx.CENTER)

        hbox.Add(b3, proportion=1, flag=wx.CENTER)

        #hbox.Add(b4, proportion=1, flag=wx.CENTER)

        #self.panel.SetSizer(hbox)

        vbox = wx.BoxSizer(wx.VERTICAL)

        vbox.Add(statictext, proportion=1, flag=wx.ALIGN\_CENTER\_HORIZONTAL)

        vbox.Add(image0\_sbitmap, proportion=2, flag=wx.ALIGN\_CENTER\_HORIZONTAL)

        vbox.Add(hbox, proportion=2, flag=wx.CENTER)

        vbox.Add(b4, proportion=1, flag=wx.CENTER)

        vbox.Add(self.image, proportion=1, flag=wx.CENTER | wx.FIXED\_MINSIZE)

        self.panel.SetSizer(vbox)

    def on\_click(self, event):

        event\_id = event.GetId()

        if event\_id == 1:

            self.image.SetBitmap(self.bmps[1])

            self.panel.Layout()

        elif event\_id == 4:

            self.image.SetBitmap(self.bmps[2])

            self.panel.Layout()

        elif event\_id == 5:

            self.image.SetBitmap(self.bmps[3])

            self.panel.Layout()

        elif event\_id == 2:

            self.image.SetBitmap(self.bmps[4])

            self.panel.Layout()

        elif event\_id == 3:

            self.image.SetBitmap(self.bmps[5])

            self.panel.Layout()

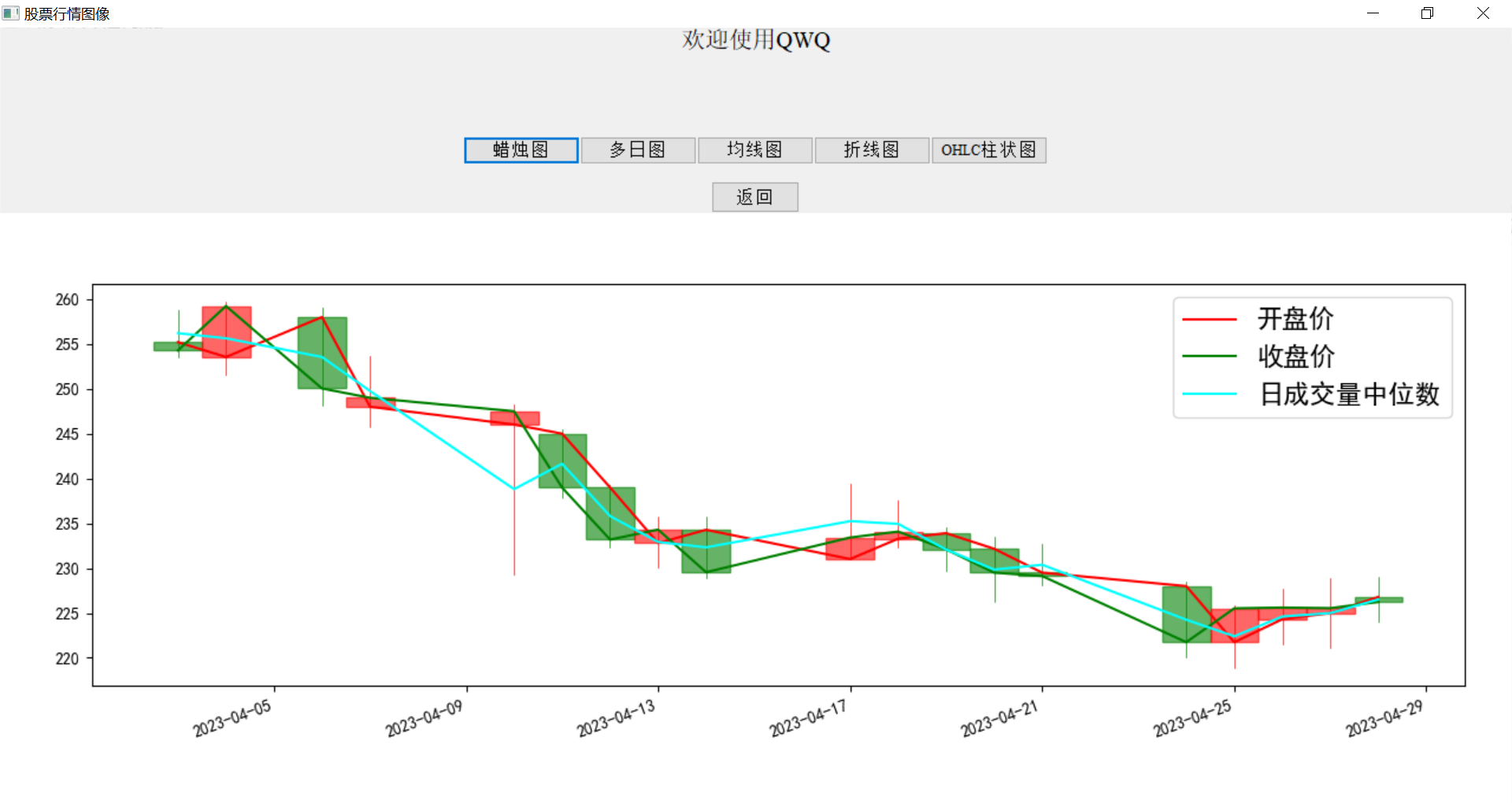
        else:

            self.Close()

            self.Destroy()

            workthread = threading.Thread(target=workthread\_body(), name='WorkThread')

            workthread.start()



2.3功能键设计

# 查询图表按钮

    def showfig(self, event):

        # 设置中文字体

        plt.rcParams['font.family'] = ['SimHei']

        plt.rcParams['axes.unicode\_minus'] = False

        datafile = fetchData(self.text[0], self.text[1], self.text[2], self.text[3])

        data = findselect\_hisq\_data(self.text[0], self.text[1], self.text[2], self.text[3])

        self.savepath = self.text[0]+'-'+self.text[1] + 'to' + self.text[2] +'-'+ self.text[3] +' '+ '.png'

        # 成交量折线图

        # 从date中提取成交量数据

        volume\_map = map(lambda it: it['成交量'], data)

        # 蒋volume\_map转换为成交量列表

        volume\_list = list(volume\_map)

        # 从data中提取日期数据

        data\_map = map(lambda it: it['日期'], data)

        # 将date\_map转换为日期列表

        date\_list = list(data\_map)

        # 设置图表大小

        plt.figure(figsize=(14, 4))

2.4爬虫记录股票表格设计

# 创建表格

    def creategridpanel(self, parent):

        panel = wx.Panel(parent, size=(1000, 600))

        grid = wx.grid.Grid(panel, name='grid', size=(1000, 600))

        self.initGrid()

        box = wx.BoxSizer()

        box.Add(grid, 1, flag=wx.ALL | wx.EXPAND, border=0)

        panel.SetSizer(box)

        return panel

    # 初始化表格

    def initGrid(self):

        grid = self.FindWindowByName('grid')

        table = MyGridTable(self.data)

        grid.SetTable(table, True)

        # 设置行列宽

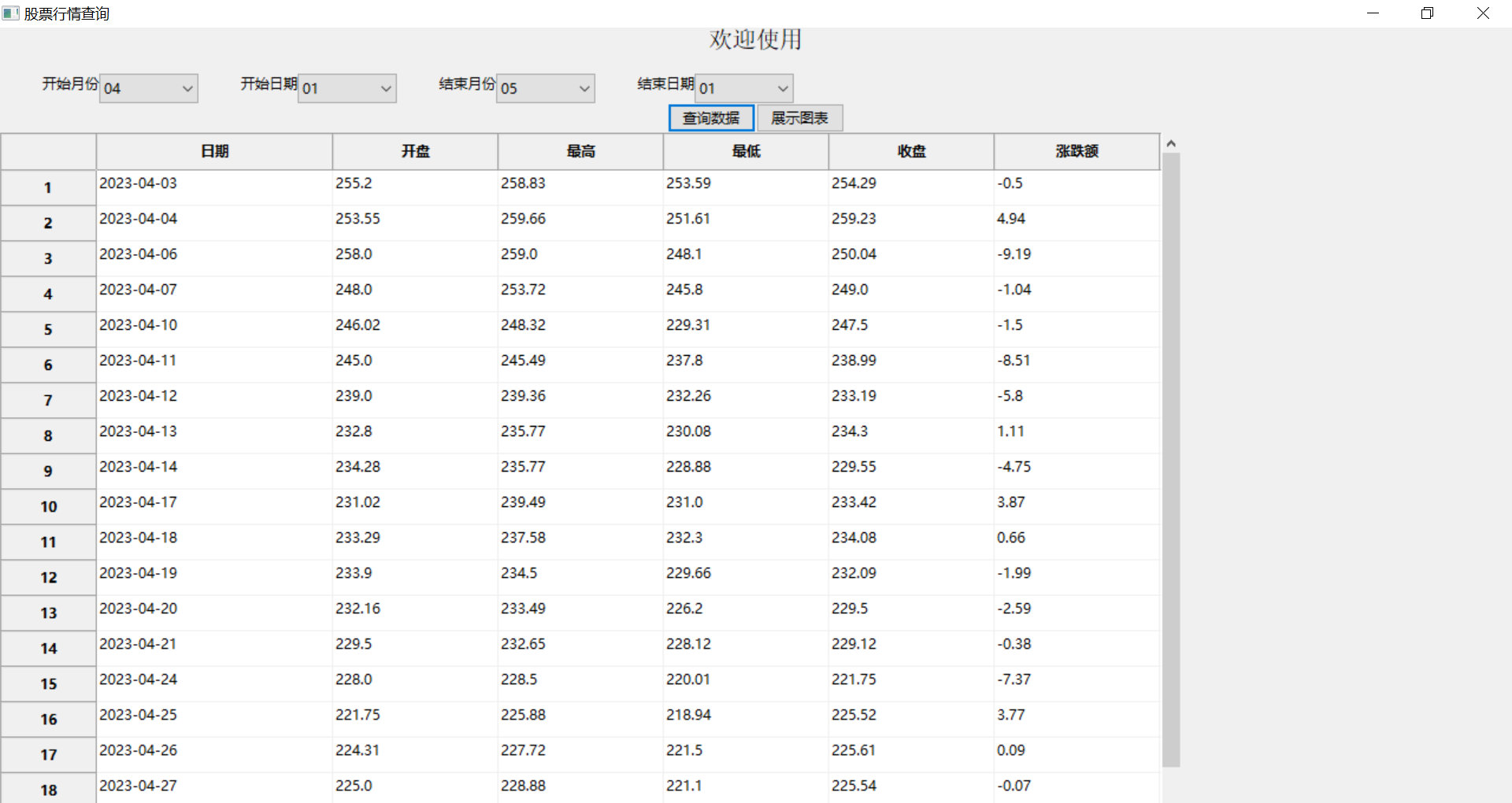
        rowsizeinfo = wx.grid.GridSizesInfo(30, [])

        grid.SetRowSizes(rowsizeinfo)

        colsizeinfo = wx.grid.GridSizesInfo(0, [200, 140, 140, 140, 140, 140, 140, 180, 220, 180, 180])

        grid.SetColSizes(colsizeinfo)

        return grid



2.5各类图的绘制

# pot\_hisvolume(date\_list, volume\_list)

        dates = date\_list

        volumes = volume\_list

        # 绘制线段

        plt.plot(dates, volumes)

        plt.setp(plt.gca().get\_xticklabels(), rotation=20, horizontalalignment='right')

        plt.title('股票历史成交量', fontsize=20)  # 添加图表标题

        plt.ylabel('成交量', fontsize=20)  # 添加y轴标题

        plt.xlabel('交易日期', fontsize=20)  # 添加x轴标题

        self.savepath2 = 'pot' + self.savepath

        plt.savefig(self.savepath2)

        def pot\_his\_bar(date\_list, p\_list, ylabel):

            """绘制OHLC柱状图"""

            # 绘制柱状图

            plt.bar(date\_list, p\_list)

            plt.title('股票历史成交量')  # 添加图表标题

            plt.ylabel(ylabel)  # 添加y轴标题

            plt.xlabel('交易日期')  # 添加x轴标题

        # 从data中提取日期数据

        date\_map = map(lambda it: it['日期'], data)

        # 将date\_map转换为日期列表

        date\_list = list(date\_map)

        # 从data中提取开盘价数据

        open\_map = map(lambda it: it['开盘'], data)

        # 将open\_map转换为开盘价列表

        open\_list = list(open\_map)

        # 从data中提取最高价数据

        high\_map = map(lambda it: it['最高'], data)

        # 将high\_map转换为最高价列表

        high\_list = list(high\_map)

        # 从data中提取最低价数据

        low\_map = map(lambda it: it['最低'], data)

        # 将low\_map转换为最低价列表

        low\_list = list(low\_map)

        # 从data中提取收盘价数据

        close\_map = map(lambda it: it['收盘'], data)

        # 将close\_map转换为收盘价列表

        close\_list = list(close\_map)

        # 设置图表大小

        plt.figure(figsize=(12, 5))

        plt.subplot(411)

        pot\_his\_bar(date\_list, open\_list, '开盘价')

        plt.setp(plt.gca().get\_xticklabels(), rotation=15, horizontalalignment='right')

        plt.subplot(412)

        pot\_his\_bar(date\_list, close\_list, '收盘价')

        plt.setp(plt.gca().get\_xticklabels(), rotation=15, horizontalalignment='right')

        plt.subplot(413)

        pot\_his\_bar(date\_list, high\_list, '最高价')

        plt.setp(plt.gca().get\_xticklabels(), rotation=15, horizontalalignment='right')

        plt.subplot(414)

        pot\_his\_bar(date\_list, low\_list, '最低价')

        plt.setp(plt.gca().get\_xticklabels(), rotation=15, horizontalalignment='right')

        plt.tight\_layout()  # 调整布局

        plt.savefig('his\_bar' + self.savepath)

        self.savepath1 = 'his\_bar' + self.savepath

        """绘制阴阳烛图1"""

        # 从CSV文件中读取数据到DataFrame数据结构中

        quotes = pandas.read\_csv(datafile,

                                 index\_col=0,

                                 parse\_dates=True,

                                 infer\_datetime\_format=True)

        # 绘制一个子图，并设置子图大小

        fig, ax = plt.subplots(figsize=(15, 5))

        # 调整子图参数SubplotParams

        fig.subplots\_adjust(bottom=0.2)

        mpl\_finance.candlestick\_ohlc(ax, zip(mdates.date2num(quotes.index.to\_pydatetime()),

                                             quotes['开盘'], quotes['最高'],

                                             quotes['最低'], quotes['收盘']),

                                     width=1, colorup='r', colordown='g', alpha=0.6)

        ax.xaxis\_date()

        ax.autoscale\_view()

        quotes['日成交量中位数'] = (quotes['最高'] + quotes['最低']) / 2

        plt.plot(quotes.index, quotes['开盘'], 'r-', label='开盘价')

        plt.plot(quotes.index, quotes['收盘'], 'g-', label='收盘价')

        # plt.plot(quotes.index, quotes['最高'], 'b-', label='最高价')

        # plt.plot(quotes.index, quotes['最低'], 'y-', label='最低价')

        plt.plot(quotes.index, quotes['日成交量中位数'], 'cyan', label='日成交量中位数')

        plt.legend(loc='best', fontsize=16)

        plt.setp(plt.gca().get\_xticklabels(), rotation=20, horizontalalignment='right')

        plt.savefig(self.savepath)

        """绘制阴阳烛图2"""

        # 从CSV文件中读取数据到DataFrame数据结构中

        quotes1 = pandas.read\_csv(datafile,

                                  index\_col=0,

                                  parse\_dates=True,

                                  infer\_datetime\_format=True)

        # print(quotes)

        quotes1.reset\_index(inplace=True)

        quotes1 = quotes1[['日期', '开盘', '收盘', '最低', '最高', '成交量']]

        quotes1.columns = ['datetime', 'open', 'close', 'low', 'high', 'volume']

        quotes1["datetime"] = pandas.to\_datetime(quotes1["datetime"])

        quotes1.set\_index("datetime", inplace=True)

        s = mpf.make\_mpf\_style(base\_mpf\_style='yahoo', rc={'font.family': 'SimHei'})

        mpf.plot(

            data=quotes1,

            type="candle",

            # title="蜡烛图与三五日均线图",

            ylabel="金额(￥)",

            style=s,

            figsize=(14, 5),

            mav=(3, 5),  # 整数，或包含整数的列表/元组

            volume=True,

            savefig='蜡烛图2' + self.savepath,

            ylabel\_lower="成交量",

            figscale=2

        )

        self.savepath3 = '蜡烛图2' + self.savepath

        """绘制阴阳烛图3"""

        s1 = mpf.make\_mpf\_style(base\_mpf\_style='default', rc={'font.family': 'SimHei'})

        mpf.plot(

            data=quotes1,

            type="candle",

            # title="蜡烛图+两日均线图",

            ylabel="金额(￥)",

            style=s1,

            figsize=(15, 5),

            mav=(2),  # 整数，或包含整数的列表/元组

            volume=True,

            savefig='蜡烛图3' + self.savepath,

            ylabel\_lower="成交量",

            figscale=2

        )

        self.savepath4 = '蜡烛图3' + self.savepath

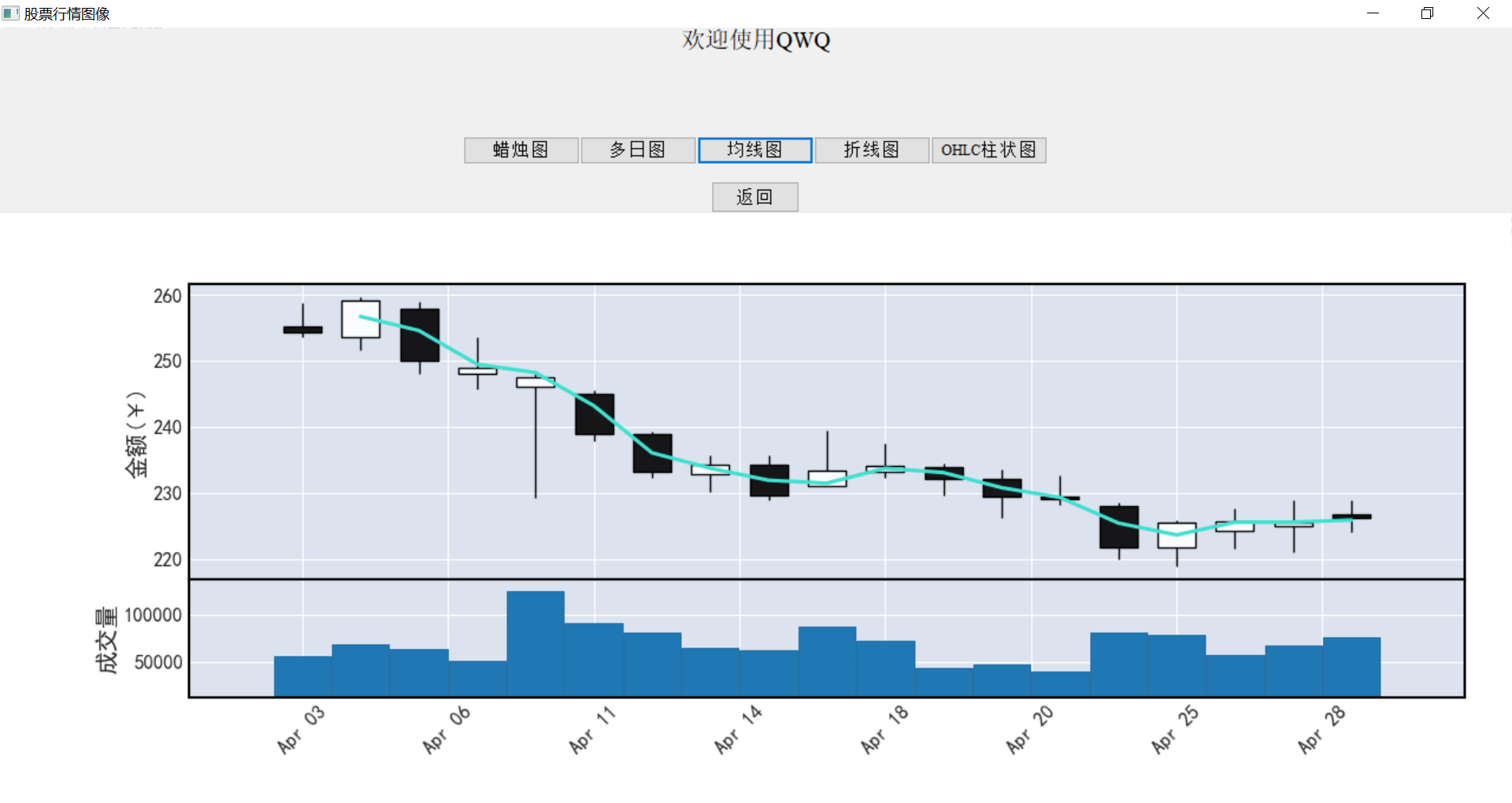
        self.Close()

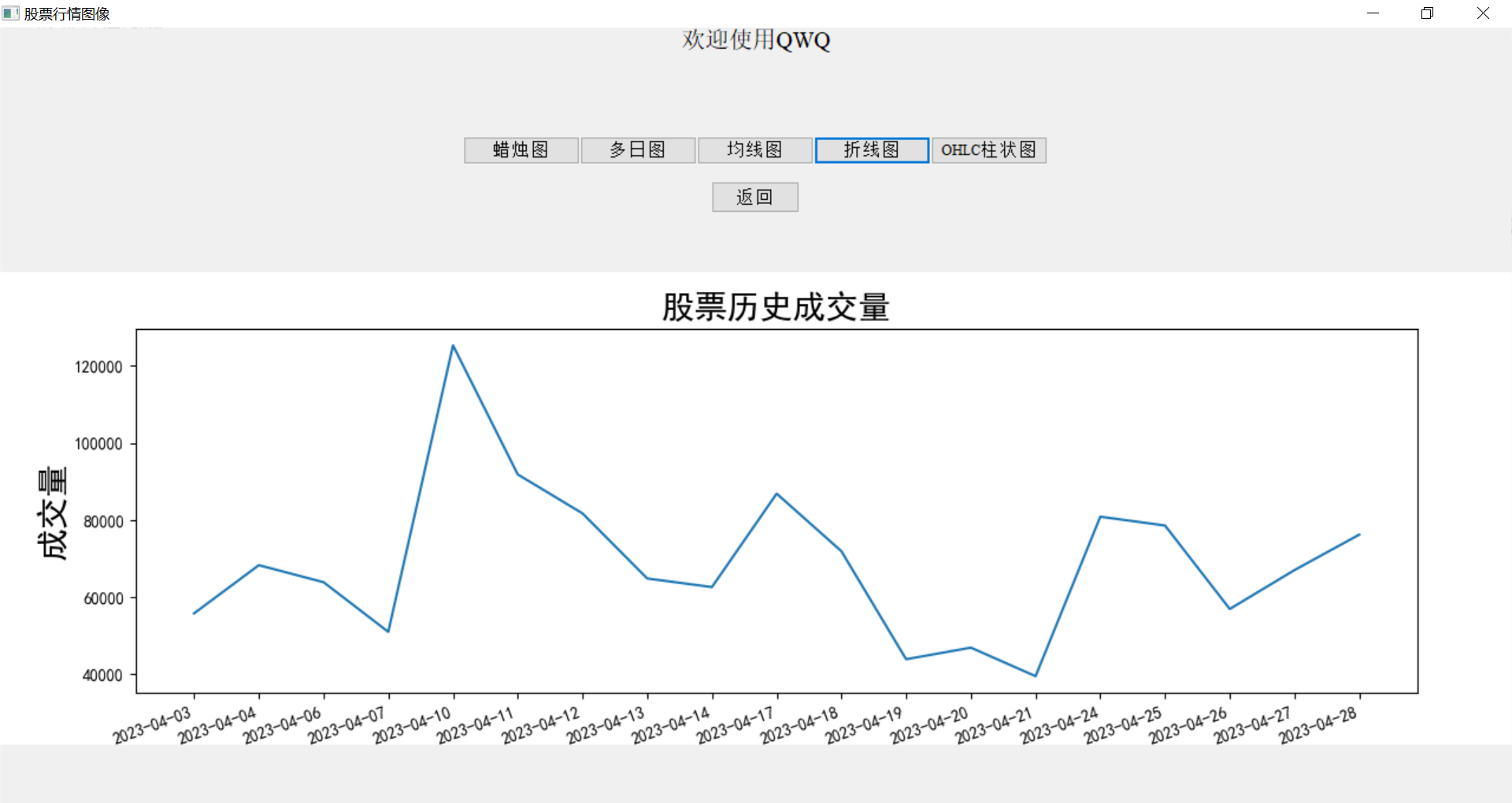
        self.Destroy()

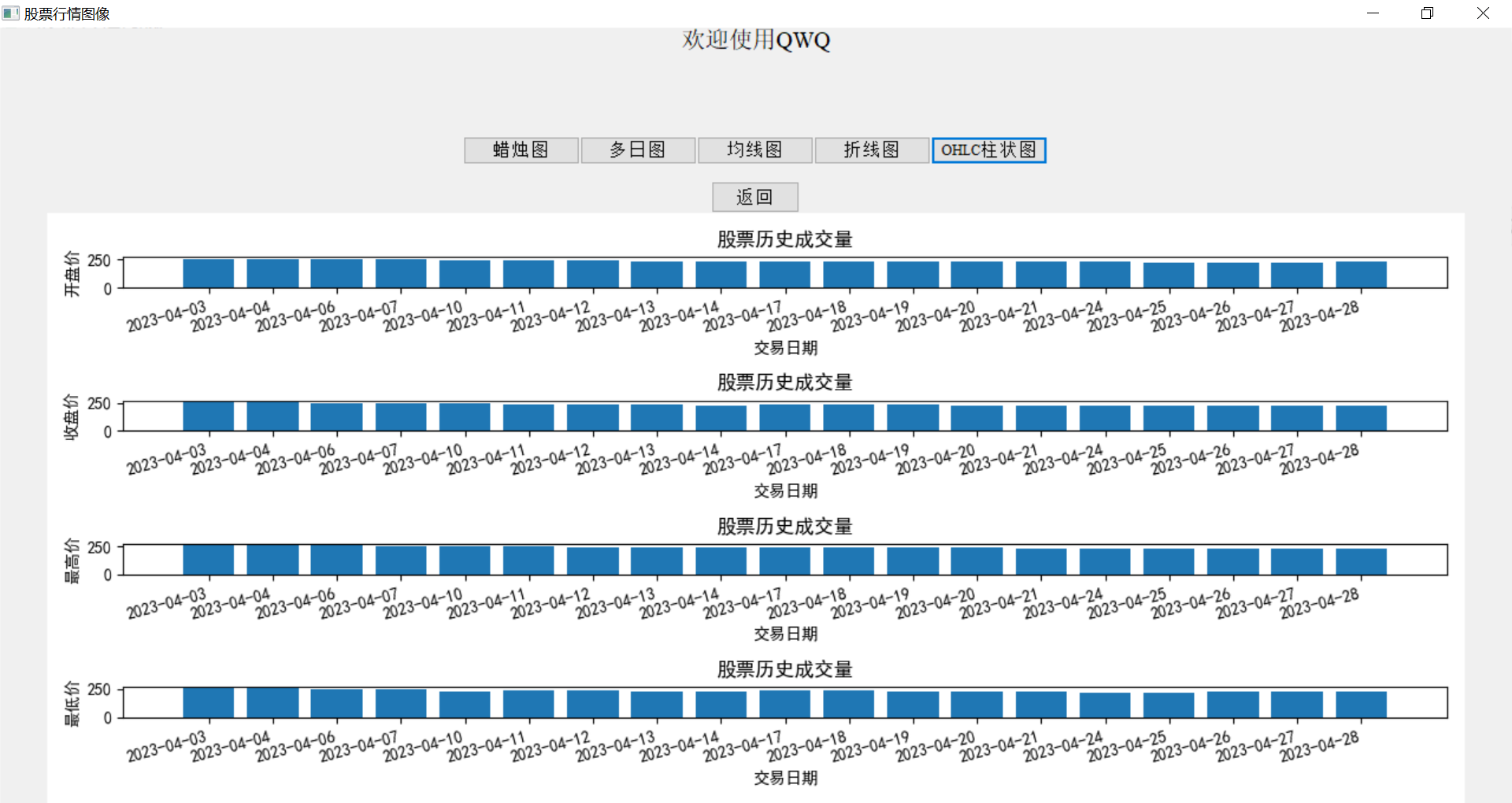
        frame = MyFrame2(self.savepath, self.savepath1, self.savepath2, self.savepath3, self.savepath4)

        frame.Show()









# 不足与问题：

因为爬取数据字符串长度的不同, 未能设计选取股票的功能, 希望日后进行改进.

# 实验收获：

1. 学会了基本的html知识，懂得利用浏览器开发者工具获取需要的网络数据。
2. 实践了爬虫的使用，能够使用爬虫进行静态和动态数据的爬取。
3. 实践了在Python语言中进行sql语句的嵌入执行。