

目录

1 实验目标与基本要求	3
1.1 实验目标	3
1.2 基本要求	3
2 主要知识点、重点与难点	3
2.1 主要知识点	3
2.2 重点	3
2.3 难点	4
3 实验过程设计	4
4 主要实现过程	4
4.1 爬虫准备	4
4.1.1 确定网站：搜狐股票	4
4.1.2 查看爬取元素的代码	5
4.1.3 精准确定爬取元素	7
4.2 获取 HTML 代码	7
4.2.1 伪装用户	7
4.2.2 使用 findelement 爬取相关表格	8
4.3 数据存储	9
4.3.1 创建数据库、表	10
4.3.2 数据插入数据库	11
4.3.3 txt 文件的生成	12
4.4 创建图形化界面	12
4.4.1 设计图形化界面	12
4.4.2 构建 MyFrame 类	14
4.4.3 构建窗口对象	16
4.4.4 按钮功能的实现	16

4.4.5	grid 构建	17
4.5	数据处理及可视化	18
4.5.1	数据处理	18
4.5.2	K 线图的绘制	18
5	效果展示	19
5.1	开始运行展示	19
5.2	界面展示	20
5.2.1	查询表格	21
5.2.2	查询 K 图	21
5.3	键入其他股票代码效果展示	22
5.3.1	更新查询 K 图效果展示	23
6	总结	24
6.1	关于爬虫	24
6.2	关于可视化界面	26
6.3	关于绘图	26
6.4	关于数据库	26
7	附录	26
7.1	股票代码 300117 的界面 html 源码	26
7.2	实验源码	45

1 实验目标与基本要求

1.1 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。

1.2 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；

2 主要知识点、重点与难点

2.1 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

2.2 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

2.3 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。

3 实验过程设计

- 1) 在互联网中寻找可以提取数据的网站；
- 2) 构造网络爬虫框架以及正则表达式；
- 3) 设计数据库表，建立数据库，将获取的数据存储到数据库中；
- 4) 设计数据库访问语句，可以以表格的形式展现数据；
- 5) 可以将一只股票 30 天的数据用 K 线图展现。

4 主要实现过程

4.1 爬虫准备

4.1.1 确定网站：搜狐股票

根据实验要求，本次实验不限制爬取对象，由于 Nasdaq 为国外网站，访问时不能直接出现数据，因此本次实验选取了搜狐证券进行爬虫操作。

网址：<https://q.stock.sohu.com/cn/>

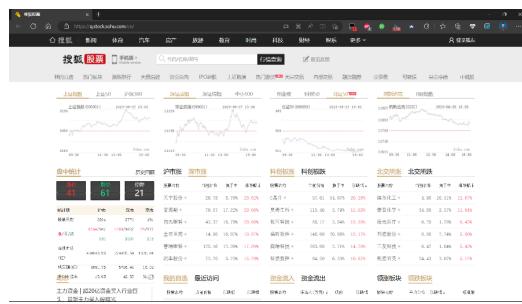


图 1: 搜狐股票网站

对于某只特定的股票，通过分析网址，只需要在上述的 url+ 该股票代码 + /lshq.shtml，就可以进入股票历史数据界面，以贵州茅台为例：

日期	开盘	收盘	涨跌额	涨跌幅%	成交量(手)	成交金额(万元)	换手率
2022-12-29	1757.92	1757.92	0.00	0.00%	3104039	3479775.52	15.58%
2023-01-02	1757.92	1757.92	0.00	0.00%	25506	449990.61	0.23%
2023-01-03	1744.00	1745.91	13.91	0.80%	27092	450447.70	0.23%
2023-01-04	1700.00	1701.28	1.28	0.74%	20582	352495.57	0.19%
2023-01-05	1721.00	1706.00	-15.00	-8.64%	20221	545912.60	0.18%
2023-01-06	1746.00	1728.47	-17.53	-0.99%	15216	321192.75	0.18%
2023-01-09	1760.11	1761.00	+0.89	0.05%	18868	546850.69	0.18%
2023-01-10	1740.00	1786.00	146.00	8.29%	258403	320400.20	0.18%
2023-01-11	1785.00	1786.00	+1.00	0.06%	18514	322010.75	0.18%
2023-01-12	1740.00	1785.00	35.00	2.51%	20467	532040.61	0.24%
2023-01-13	1720.00	1715.42	-4.58	-0.50%	21232	304452.69	0.17%
2023-01-16	1880.00	1723.00	-157.00	-8.75%	29842	304501.62	0.24%
2023-01-17	1747.26	1694.15	-53.10	-3.48%	51106	673265.70	0.18%
2023-01-18	1765.00	1745.50	-20.50	-1.48%	20209	512885.44	0.23%
2023-01-19	1780.00	1771.70	-18.30	-1.08%	28418	517115.63	0.23%
2023-01-20	1795.00	1789.99	-5.01	-0.28%	12505	242610.05	0.14%
2023-01-23	1805.00	1796.96	-8.04	-0.45%	14804	267825.19	0.18%
2023-01-24	1795.00	1796.96	+1.96	0.11%	11141	264444.64	0.14%

图 2: 贵州茅台股票历史数据

4.1.2 查看爬取元素的代码

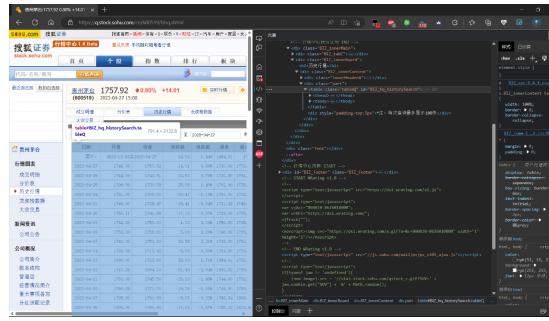
进入网站开发者模式（按下 F12）查看网页的源代码，点击元素，鼠标指向页面中的元素，通过这个工具确定需要爬取的部分的源代码

图 3: 查看源代码

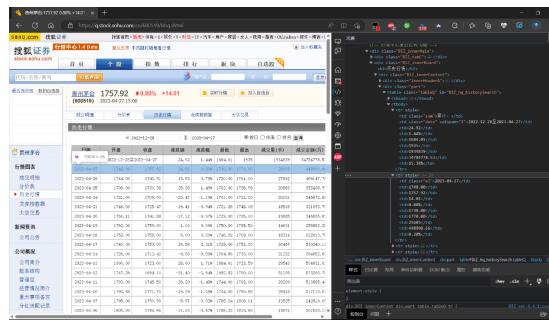
通过元素选取确定了数据表格的

$class = 'tableQ'$

$id = 'BIZ_hq_historySearch'$



(a) 确定数据表格相关属性



(b) 定位表格

图 4: 源代码审查

4.1.3 精准确定爬取元素

观察页面布局可以发现表格体的第一行是累计，而累计并不是本实验所需要的数据，因此确定爬取数据从第二行开始

贵州茅台						
行业表现		2020-09-02~2021-09-02				
日期	开盘	最高	最低	收盘	涨跌	成交量(手)
2020-09-02	1740.00	1757.82	14.01	1,694.15	+159.15	135879.51
2020-09-03	1740.00	1743.11	13.52	1,738.17	+105.06	25065.68
2020-09-04	1740.00	1743.11	13.52	1,738.17	+105.06	25065.68
2020-09-05	1720.00	1736.00	-78.47	1,736.00	+1.39	25065.68
2020-09-06	1740.00	1755.47	-14.41	1,748.00	+1.58	15014.75
2020-09-07	1780.00	1741.68	-17.12	1,676.17	-118.83	15065.68
2020-09-08	1750.00	1756.00	5.00	1,756.00	+1.00	15014.75
2020-09-09	1750.00	1756.00	5.00	1,756.00	+1.00	15014.75
2020-09-10	1780.00	1785.00	59.58	2,338.17	+563.09	50360.12
2020-09-11	1720.00	1712.42	-5.48	1,558.17	-162.82	25210.68
2020-09-12	1730.00	1742.00	12.00	1,742.00	+12.00	25065.68
2020-09-13	1740.00	1894.00	-98.41	1,894.00	+10.00	50310.75
2020-09-14	1790.00	1745.93	-58.23	1,488.17	-310.83	51036.44
2020-09-15	1790.00	1771.79	-18.28	1,698.17	-100.82	51036.44
2020-09-16	1790.00	1786.00	-4.00	1,786.00	+0.00	51036.44
2020-09-17	1790.00	1796.99	-17.99	1,796.99	+0.00	51036.44
2020-09-18	1882.00	1814.59	11.52	1,698.17	-183.17	51036.44
2020-09-19	1820.00	1830.07	-17.97	1,830.07	+100.00	15065.68
2020-09-20	1820.00	1830.07	-17.97	1,830.07	+100.00	15065.68
2020-09-21	1790.00	1898.00	18.00	1,898.17	+100.00	51036.44
2020-09-22	1790.00	1796.00	4.23	1,648.17	-151.87	15301.94
2020-09-23	1770.00	1761.80	-8.14	1,761.80	+10.80	51036.44
2020-09-24	1790.00	1778.82	-11.18	1,778.82	-11.18	51036.44
2020-09-25	1790.00	1778.82	-11.18	1,778.82	-11.18	51036.44
2020-09-26	1790.00	1774.89	-15.11	1,698.17	-99.82	51036.44

图 5: 分析表格元素

4.2 获得 HTML 代码

4.2.1 伪装用户

使用 request 库对网页源代码进行爬取，分析得知，网站的历史行情数据都是动态数据，无法通过静态方式爬取，因此使用 selenium 库进行数据爬取

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome()
4 driver.get('http://q.stock.sohu.com/cn/'+stack_code+'/lshq.shtml')
5 time.sleep(5)
```

调用 selenium 库

4.2.2 使用 findelement 爬取相关表格

在尝试这种方法之前，曾经尝试使用 beautifulsoup 库通过类定位爬取数据表格，但是最终失败了，改变策略使用 findelement，并调用 By.XPATH，通过 xpath 来定位数据表格的相关属性并爬取，在确定 xpath 的时候，使用了正则表达式来定位，在 chrome 的开发者模式下，对网页元素分析可以直接获得 xpath

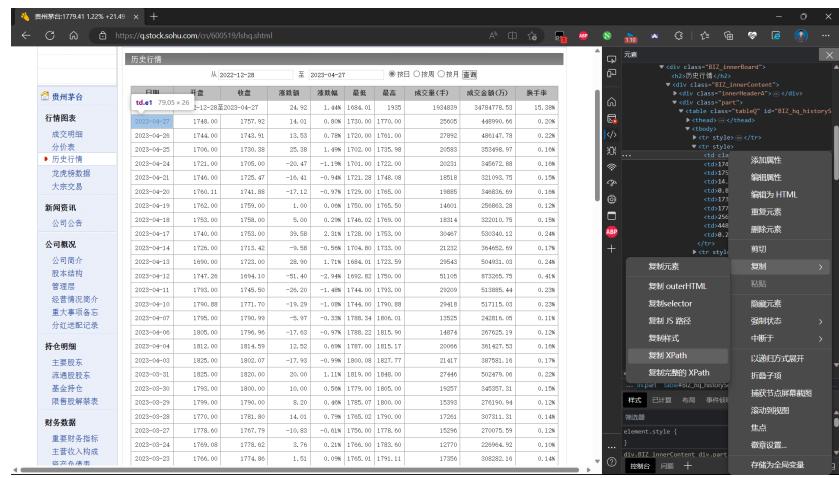


图 6: 获取元素 xpath

将获得的 xpath 放入 findelement 函数，然后将爬取的数据存放在 kv 字典中，再通过循环结构重复每一行的爬取操作。

```

1   f=open("data.txt","w",encoding="utf-8")
2
3   for i in range(2, 81):
4       kv={}
5       try:
6           kv["日期"] = driver.find_element(By.XPATH, '//*[@id="'
7               BIZ_hq_historySearch"]/tbody/tr['+str(i)+']/td[1]
8               ').text
9
10      with open('data.txt', 'a') as f:
11          f.write(str(kv) + '\n')
12
13      print(str(kv))
14
15  
```

```

kv[ "开盘价" ] = driver . find_element(By.XPATH, '//*[@@id
    ="BIZ_hq_historySearch"]/tbody/tr [ '+str(i)+']/td
    [ 2 ] ') . text
2
kv[ "最高价" ] = driver . find_element(By.XPATH, '//*[@@id
    ="BIZ_hq_historySearch"]/tbody/tr [ '+str(i)+']/td
    [ 7 ] ') . text
kv[ "最低价" ] = driver . find_element(By.XPATH, '//*[@@id
    ="BIZ_hq_historySearch"]/tbody/tr [ '+str(i)+']/td
    [ 6 ] ') . text
4
kv[ "收盘价" ] = driver . find_element(By.XPATH, '//*[@@id
    ="BIZ_hq_historySearch"]/tbody/tr [ '+str(i)+']/td
    [ 3 ] ') . text
kv[ "成交量" ] = driver . find_element(By.XPATH, '//*[@@id
    ="BIZ_hq_historySearch"]/tbody/tr [ '+str(i)+']/td
    [ 8 ] ') . text
6
print( "complete" )
except:
8
    print( "找不到元素" )
    continue
10
month_data . append( kv )
to_write = json . dumps( kv , ensure_ascii=False )
12
f . write( to_write )
f . close()

```

findelement 数据爬取

由于后续有绘制 K 线图的需求，在数据方面，本实验只需要存储日期、开盘价、最高价、最低价和收盘价等数据，此外再加入成交量，至此爬取数据部分完毕。

4.3 数据存储

本实验使用 MySQL 数据库，数据存储实现直接调用 pymysql 库，并定义存储函数，在存储函数中实现创建数据库、创建表、插入数据等操作.

4.3.1 创建数据库、表

在 python 中，通过 pymysql 库能够实现与数据库连接，并使用 python 代码实现 sql 语句的调用。

先在存储函数中生成游标对象，使用游标对象实现数据库的相关操作

```
1 cursor = my_database.cursor()
```

生成游标对象

然后通过游标对象创建数据库

```
1 sql_createDataBase = "create database if not exists  
2     stockData"  
3 cursor.execute(sql_createDataBase)  
4 sql_useDataBase = "USE stockData"
```

创建数据库

对数据表进行设计

```
1 create table if not exists data  
2 (  
3     date DATE,  
4     opening_price float ,  
5     closing_price float ,  
6     highest float ,  
7     lowest float  
8 )
```

数据表的设计

通过游标对象创建表

```
1 cursor.execute(sql_useDataBase)
2 sql_createTable = '''
3     create table if not exists data
4     (
5         date DATE,
6             opening_price float ,           closing_price float ,
7                 highest float ,
8                     lowest float
9     )
10    '''
11
12 cursor.execute(sql_createTable)
```

创建表

4.3.2 数据插入数据库

先写出 sql 的数据插入语句

```
1 Insert into data values('date','opening_price','
2                           closing_price','highest','lowest')
```

sql 数据插入语句

对于本实验数据库的数据表，只存储后续需要画 K 线图的日期、开盘价、收盘价、最高价、最低价，在定义存储函数的时候，将待存储的数据以字典类型作为变量存入数据库中，在字典列表中取出数据，使用 for 循环将数据拆分，分别传入数据

```
1 for item in data:
2     sql_Insert = '''Insert into data values(
3         {0},{1},{2},{3},{4})'''.format(item['日期'], item[
4             '开盘价'], item['收盘价'], item['最高价'], item['最低
5                 价'])
6
7 cursor.execute(sql_Insert)
```

数据插入数据库，frame

至此数据库、表创建完成，存储函数实现

4.3.3 txt 文件的生成

在 findelement 数据爬取代码段中，同时实现了将待存储数据转换成 json 数据格式并写入 data.txt 文件中，在后续的数据分析操作中便于 python 其他库读取

```
1 f=open("data.txt","w",encoding="utf-8")  
2  
3 for i in range(2, 81):  
4     kv={}  
5     try:  
6         ...  
7     except:  
8         ...  
9     month_data.append(kv)  
10    to_write = json.dumps(kv, ensure_ascii=False)  
11    f.write(to_write)  
12    f.close()
```

txt 文件存储

4.4 创建图形化界面

4.4.1 设计图形化界面

本实验使用 wxpython 库创建 GUI 界面。

设计了一个简单的界面：

界面的左侧是预留空间，用来打印输出某只股票的历史行情数据表格；

界面的右侧有一个文本框，用来输入股票代码；

文本框下方有三个按钮，分别是“查询 K 图”、“更新”和“查看表格”，实现功能如下：

查询 K 图：点击后，打开一个 html 文件，内部存有已经绘制好的 K 线图；

更新：先在文本框最终输入想要查询的股票代码，点击更新后，爬取数据并存入数据库，同时生成一个 xlsx 表格，用来正确性验证；

查看表格：点击后，在左侧预留空间打印输出某只股票的历史行情数据表格。

	datetime	open	close	low	high	trade_sum
1	2023-04-28	3.37	3.56	3.37	3.53	216482
2	2023-04-27	3.40	3.43	3.36	3.39	91171
3	2023-04-26	3.34	3.50	3.32	3.42	138342
4	2023-04-25	3.45	3.45	3.27	3.36	122378
5	2023-04-24	3.23	3.45	3.20	3.39	248542
6	2023-04-21	3.27	3.35	3.18	3.19	98079
7	2023-04-20	3.35	3.35	3.24	3.27	111765
8	2023-04-19	3.41	3.42	3.35	3.35	57144
9	2023-04-18	3.43	3.43	3.38	3.40	64232
10	2023-04-17	3.44	3.48	3.41	3.43	67008
11	2023-04-14	3.49	3.49	3.42	3.45	86886
12	2023-04-13	3.46	3.58	3.45	3.47	113233
13	2023-04-12	3.42	3.47	3.40	3.46	64489
14	2023-04-11	3.39	3.43	3.36	3.42	67523
15	2023-04-10	3.50	3.51	3.39	3.40	95986
16	2023-04-07	3.44	3.50	3.41	3.49	88527
17	2023-04-06	3.45	3.45	3.38	3.43	98005
18	2023-04-04	3.57	3.58	3.44	3.45	194467
19	2023-04-03	3.57	3.59	3.55	3.58	89009
20	2023-03-31	3.55	3.58	3.53	3.57	82424
21	2023-03-30	3.62	3.62	3.52	3.56	114574
22	2023-03-29	3.65	3.67	3.60	3.60	108687
23	2023-03-28	3.76	3.76	3.63	3.64	153336

查询K图
更新
查看表格

图 7: 图形化界面

4.4.2 构建 MyFrame 类

对于 MyFrame 类的设计，首先在类的初始化函数中，定义了一个 panel，用来存放界面的各个组件，然后在 panel 中定义了一个文本框，用来输入股票代码，文本框下方有三个按钮，分别是“查询 K 图”、“更新”和“查看表格”，代码实现如下：

```
1 class MyFrame(wx.Frame):
2     data = []
3     column_names = []
4     stock_Code = ""
5
6     def __init__(self, data, column_names):
7         super().__init__(parent=None, title="股票数据显示界面",
8                         size=(600, 600))
9         self.data = data
10        self.column_names = column_names
11        self.Centre()
12        panel = wx.Panel(parent=self)
13        # self.message1 = wx.StaticText()
14        # self.message1.SetLabelText("请输入股票代码")
15        # self.message1.SetPosition((400,370))
16        self.number = wx.TextCtrl(panel, pos=(450, 370))
17        query_button = wx.Button(parent=panel, id=1, label='查询
18                                K图', pos=(450, 400))
19        post = wx.Button(parent=panel, id=2, label='更新', pos
20                         =(450, 435))
21        show = wx.Button(parent=panel, id=3, label='查看表格',
22                         pos=(450, 470))
23        self.Bind(wx.EVT_BUTTON, self.on_click, query_button)
24        self.Bind(wx.EVT_BUTTON, self.on_click, post)
25        self.Bind(wx.EVT_BUTTON, self.on_click, show)
26        # self.Bind(wx.EVT_TEXT, self.EvtText)
27        # 建立表格
```

```

1 def generate_xlsx(self):
2     self.grid = self.CreateGrid(self)
3     self.Bind(wx.grid.EVT_GRID_LABEL_LEFT_CLICK, self.
4         OnLabelLeftClick)
5
6     def on_click(self, event):
7         event_id = event.GetId()
8         print(event_id)
9         if event_id == 1:
10             print("查询K图")
11             webbrowser.open('K图.html')
12         elif event_id == 2:
13             self.stock_Code = self.number.GetValue()
14             print(self.number.GetValue())
15             update_xlsx(self, self.stock_Code)
16         elif event_id == 3:
17             self.generate_xlsx()
18
19     def OnLabelLeftClick(self, event):
20         print("RowIndex: {}".format(event.GetRow()))
21         print("ColIdx: {}".format(event.GetCol()))
22         print(self.data[event.GetRow()])
23         event.Skip()
24
25     def CreateGrid(self, parent):
26         grid = wx.grid.Grid(parent)
27         grid.CreateGrid(len(self.data), len(self.data[0]))
28
29         for row in range(len(self.data)):
30             for col in range(len(self.data[row])):
31                 grid.SetColLabelValue(col, self.column_names[col])

```

```

        grid . SetCellValue ( row , col , self . data [ row ] [ col ])
2      # 设置行和列自定调整
3      grid . AutoSize ()
4
5      return  grid

```

MyFrame 的构建

4.4.3 构建窗口对象

在窗口对象中，首先调用了 MyFrame 类的构造函数，然后调用了 Show() 方法显示窗口，实现代码如下：

```

1 class  App(wx.App):
2     data = []
3     column_names = []
4
5     def  show(self):
6         frame = MyFrame( self . data , self . column_names )
7         frame . Show ()
8
9     return  True

```

构建窗口对象

4.4.4 按钮功能的实现

界面一共有三个按钮，分别是“查询 K 图”、“更新”和“查看表格”，实现功能如下：

查询 K 图：点击后，打开一个 html 文件，内部存有已经绘制好的 K 线图；

更新：先在文本框最终输入想要查询的股票代码，点击更新后，爬取数据并存入数据库，同时生成一个 xlsx 表格，用来正确性验证；

查看表格：点击后，在左侧预留空间打印输出某只股票的历史行情数据表格。

代码实现如下：

```
def on_click(self, event):
    event_id = event.GetId()
    print(event_id)
    if event_id == 1:
        print("查询K图")
        webbrowser.open('K图.html')
    elif event_id == 2:
        self.stock_Code = self.number.GetValue()
        print(self.number.GetValue())
        update_xlsx(self, self.stock_Code)
    elif event_id == 3:
        self.generate_xlsx()
```

按钮功能实现

4.4.5 grid 构建

在 grid 构建中，首先调用了 CreateGrid() 方法，然后调用了 AutoSize() 方法，实现代码如下：

```
def CreateGrid(self, parent):
    grid = wx.grid.Grid(parent)
    grid.CreateGrid(len(self.data), len(self.data[0]))

    for row in range(len(self.data)):
        for col in range(len(self.data[row])):
            grid.SetColLabelValue(col, self.column_names[col])
            grid.SetCellValue(row, col, self.data[row][col])
    # 设置行和列自定调整
    grid.AutoSize()
```

grid 构建

4.5 数据处理及可视化

本实验中，主要的数据处理及可视化项目就是 K 线图的绘制，在绘制前需要做相应的数据处理

4.5.1 数据处理

本项目的数据处理主要是将绘制 K 线图所需的五个对应数据分别放进五个列表中。

```
def data_Pretreatment(month_data):
    2   for data in month_data:
        my_date = data.get("日期")
        4   my_open = data.get("开盘价")
        my_close = data.get("收盘价")
        6   my_high = data.get("最高价")
        my_low = data.get("最低价")
        8   date.append(my_date)
        opening_price.append(my_open)
        closing_price.append(my_close)
        10  highest.append(my_high)
        lowest.append(my_low)
    12
```

shadowbox

4.5.2 K 线图的绘制

本次实验的 K 线图绘制并没有使用 Matplotlib 库，而是通过 pyecharts 库来实现的，pyecharts 是一个用于生成 Echarts 图表的类库，它提供了一种使用 Python 构建 Echarts 图表的简单方法。pyecharts 支持 Python2.7+ 和 Python3.4+，并且兼容多种操作系统。pyecharts 的安装也非常简单，只需要在命令行中输入 pip install pyecharts 即可。

在 pyecharts 库中，有 Kline 类可以直接调用，只需要整合好一组数据输入 Kline 的一个实例对象就可以生成 K 线图

实现代码如下：

```
def draw_K():
    2     kline = Kline().set_global_opts(title_opts=opts.TitleOpts(
        title="K线图"))
    v1 = []
    4     size = len(opening_price) # 有多少条数据（多少天）
    for i in range(size-1, 0, -1):
        6         tmp = [opening_price[i], closing_price[i], lowest[i],
        highest[i]]
        v1.append(tmp) # 整合好一组数据存入v1中
    8     # print(v1)

    10    kline.add_yaxis(series_name="日K", y_axis=v1)
    kline.add_xaxis([s for s in date])
    12    kline.render("K图.html")
```

K 线图绘制

K 线图绘制函数的最终返回一个绘有月 K 线图的 html 文件，在“查询 K 图”的按钮事件中，使用 `webbrowser.open` 通过浏览器打开 html 文件
至此，本实验的主要实现过程基本阐述完毕

5 效果展示

效果展示分别展示控制台输出情况和界面的运行情况

5.1 开始运行展示

在 pycharm 代码界面，由于预设了股票代码 300117，在右键点击运行后，会首先爬取 300117 的相关数据，并生成对应的图、数据库、txt 文件
可以看到 txt 文件中已经以 json 格式存入了数据，数据库中也存入了数据，说明程序运行正常。

图 8: 控制台输出

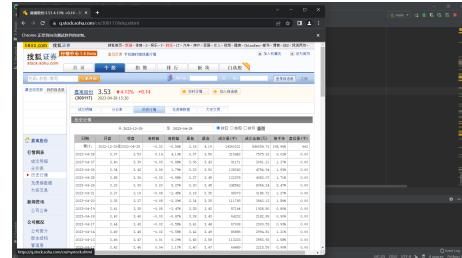


图 9: webdriver 爬取

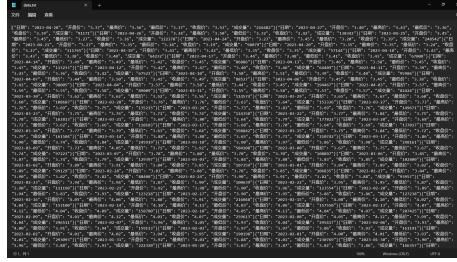


图 10: txt 文件生成

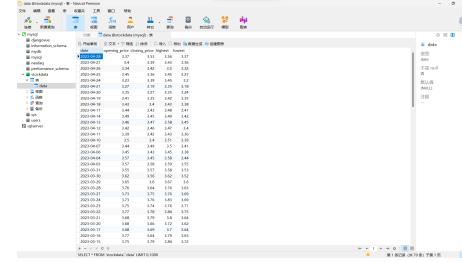


图 11: 数据库存储数据

5.2 界面展示

在界面弹出之前，程序预置爬取 300117 的数据，所以在界面弹出后，默认存储的是 300117 的数据，但是在初始界面中，是没有显示 300117 的数据的，需要点击“查看表格”按钮才会显示 300117 的数据，如图所示：

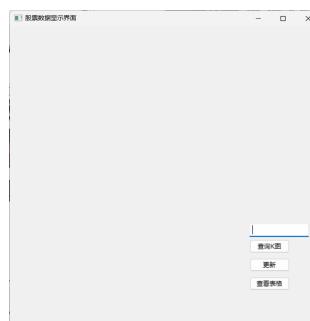


图 12: 初始界面显示

5.2.1 查询表格

首先进行表格查询，查看表格后与源数据进行比较，可以看到表格中的数据与源数据一致，说明表格查询功能正常

	datetime	open	close	low	high	trade_sum
1	2023-04-28	3.37	3.56	3.37	3.53	216482
2	2023-04-27	3.47	3.46	3.36	3.39	91171
3	2023-04-26	3.38	3.50	3.32	3.42	138342
4	2023-04-25	3.45	3.45	3.27	3.36	122378
5	2023-04-24	3.23	3.45	3.20	3.39	249542
6	2023-04-21	3.27	3.35	3.18	3.19	98079
7	2023-04-20	3.35	3.35	3.24	3.27	111765
8	2023-04-19	3.41	3.42	3.35	3.35	57144
9	2023-04-18	3.43	3.43	3.38	3.40	64232
10	2023-04-17	3.44	3.46	3.41	3.43	67008
11	2023-04-14	3.49	3.49	3.42	3.45	86886
12	2023-04-13	3.46	3.58	3.45	3.47	112233
13	2023-04-12	3.42	3.47	3.40	3.46	64489
14	2023-04-11	3.39	3.43	3.36	3.42	67523
15	2023-04-10	3.50	3.51	3.39	3.40	95986
16	2023-04-07	3.44	3.50	3.41	3.49	88527
17	2023-04-06	3.45	3.45	3.38	3.43	98005
18	2023-04-04	3.57	3.58	3.44	3.45	194467
19	2023-04-03	3.57	3.59	3.55	3.58	89009
20	2023-03-31	3.55	3.58	3.53	3.57	82424
21	2023-03-30	3.62	3.62	3.52	3.56	114574
22	2023-03-29	3.65	3.67	3.60	3.60	108687
23	2023-03-28	3.76	3.76	3.63	3.64	153336

图 13: 表格展示

5.2.2 查询 K 图

点击查询 K 图按钮后，控制台打印输出 1、查询 K 图，然后程序通过 webbrowser.open 打开浏览器，显示 K 线图，如图所示：



图 14: K 线图展示

至此，关于界面的基础展示已经展示完毕，下面将展示键入其他股票代码后的界面

5.3 键入其他股票代码效果展示

直接在文本框中键入其他股票代码，点击“更新”按钮，程序会对对应股票代码的历史行情数据进行重新爬取

#	datetime	open	close	low	high	trade_sum
1	2023-04-28	3.37	3.56	3.37	3.53	216482
2	2023-04-27	3.40	3.43	3.36	3.39	91171
3	2023-04-26	3.34	3.50	3.32	3.42	138342
4	2023-04-25	3.45	3.45	3.27	3.36	122378
5	2023-04-24	3.23	3.45	3.20	3.39	248542
6	2023-04-21	3.27	3.35	3.18	3.19	98079
7	2023-04-20	3.35	3.35	3.24	3.27	111765
8	2023-04-19	3.41	3.42	3.35	3.35	57144
9	2023-04-18	3.43	3.43	3.38	3.40	64232
10	2023-04-17	3.44	3.48	3.41	3.43	67008
11	2023-04-14	3.49	3.49	3.42	3.45	86886
12	2023-04-13	3.46	3.58	3.45	3.47	113233
13	2023-04-12	3.42	3.47	3.40	3.46	64489
14	2023-04-11	3.39	3.43	3.36	3.42	67523
15	2023-04-10	3.50	3.51	3.39	3.40	95986
16	2023-04-07	3.44	3.50	3.41	3.49	88527
17	2023-04-06	3.45	3.45	3.38	3.43	98005
18	2023-04-04	3.57	3.58	3.44	3.45	194467
19	2023-04-03	3.57	3.59	3.55	3.58	89009
20	2023-03-31	3.55	3.58	3.53	3.57	82424
21	2023-03-30	3.62	3.62	3.52	3.56	114574
22	2023-03-29	3.65	3.67	3.60	3.60	108687
23	2023-03-28	3.76	3.76	3.63	3.64	153336

图 15: 键入其他股票代码

在点击更新按钮后，程序重新调用 webdriver 爬取对应股票代码的数据，并将数据存入数据库和 txt 文件中，同时在这一过程中还验证了将数据写入 xlsx 文件的功能，如图所示：

可以看到，表格、xlsx 文件、数据库、txt 文件中的数据都已经更新，说明更新功能正常

图 16: 更新后表格展示

图 17: xlsx 文件展示

图 18: 数据库更新展示

图 19: txt 文件展示

5.3.1 更新查询 K 图效果展示

在重新爬取数据后，数据自动更新，同时绘制 K 线图的函数也被自动调用，点击查询 K 图按钮可以看到更新后的 K 线图，如图所示：

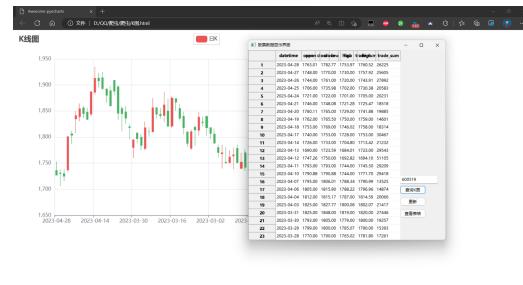


图 20: 更新 K 线图展示

至此，本实验的所有功能基本展示完毕

6 总结

本次实验涉及到了 python 爬虫、数据库、图像绘制技术以及 python 制作 GUI 界面等相关知识，综合性比较强，学习到的新知识也非常多

6.1 关于爬虫

在最开始爬取股票的历史行情数据的时候，并不知晓其历史行情数据采用动态方式，在最开始的时候获取数据使用了 request 库获取了静态的 html 文件，

```
# def getHtml(stack_code):
#     data = requests.get("https://q.stock.sohu.com/cn/" +
#     stack_code + "/lshq.shtml",
#     headers={"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"})
#     # 获得一个请求得到的静态网页
#     return data.text

# 将HTML文本转化成字典列表
# def getData(data):
#     month_data = []
#     soup = BeautifulSoup(data, "lxml") # data由getHtml()取得
#     # 爬取网页的工具
#     list = soup.find("div", class_="part").find("table",
#     class_="tableQ")
#     # 看html，已经把不需要的都删了，数据在innerbox的
#     table_bg001 border_box limit_sale下面
#     dataList = list.find_all("tr")[1:]
#     # 因为第一个tr后面是一堆th标签，不需要
#     f = open("data.txt", "w", encoding="utf-8")
```

```

#         for item in dataList:
2#             kv = {}
#             if isinstance(item, bs4.element.Tag):
4#                 tdList = item.find_all("td")
#                 # print(tdList)
6#                 kv["日期"] = tdList[0].text # 去掉td和/td, 取中间
    的内容
#                 kv["开盘价"] = tdList[1].text
8#                 kv["最高价"] = tdList[6].text
#                 kv["最低价"] = tdList[5].text
10#                kv["收盘价"] = tdList[2].text
#                kv["成交量"] = tdList[7].text
12#                month_data.append(kv)
#                to_write = json.dumps(kv, ensure_ascii=False)
14#                f.write(to_write)
#                # print(kv)
16#                f.close()
#                # print(month_data) # 字典列表
18#                return month_data

```

request 库调取

反复试验后无法获得相应数据，通过分析 html 的源代码（股票代码 300117 的界面 html 源码见附录），发现 tbody 标签中元素为空，分析确定静态文件中没有数据，因此使用 selenium 库中的 webdriver 模块获取动态数据，通过分析动态数据，最终成功获取数据，但是在获取数据的过程中，由于数据量较大，程序运行时间较长，所以在获取数据的过程中，需要设置等待时间，否则会出现获取不到数据的情况，最终通过设置等待时间解决了这一问题，同时在获取数据的过程中，由于数据量较大，所以需要设置浏览器的窗口大小，否则会出现数据不全的情况，最终通过设置浏览器窗口大小解决了这一问题，最终成功获取了数据

6.2 关于可视化界面

本实验使用了 wxpython 库来实现项目的 GUI 界面，在实现过程中，只搭建了了一个简单的框架，但是麻雀虽小五脏俱全，满足了基本功能

6.3 关于绘图

在绘图的时候，没有选择 matplotlib 库，选用了 pyecharts 库，其中的 Kline 类能够直接实现 K 线图的绘制。

但是在绘制的时候，发现 pyecharts 库的 Kline 类只能绘制一只股票的 K 线图，所以在绘制新股票的时候，需要重新读取元素绘制，最终实现了另一个 K 线图的绘制，同时覆盖了上一个元素的 K 线图，实现了 K 线图的更新。

6.4 关于数据库

本实验中，调用 pymysql 库实现了数据库的连接，通过数据库的连接，实现了数据的存储，同时在更新数据的时候，也实现了数据的更新，最终实现了数据的存储和更新。在实验过程中，由于数据库的连接需要设置 host、user、password、database 等参数，所以在实验过程中，需要设置这些参数，否则会出现连接不上数据库的情况，最终通过设置这些参数解决了这一问题，成功连接了数据库。此外还学习了 cursor 游标的使用，通过游标实现了数据的插入和更新。

7 附录

7.1 股票代码 300117 的界面 html 源码

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//  
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.  
dtd">
```

```

2   <html xmlns="http://www.w3.org/1999/xhtml">
3     <head>
4       <meta http-equiv="Content-Type" content="text/html; charset=
5         gb2312" />
6       <title>贵州茅台(600519) - 历史行情 - 股票行情中心 - 搜狐证券
7         </title>
8       <meta name="Keywords" content="贵州茅台,600519,历史行情">
9       <meta name="Description" content="贵州茅台(600519)的历史行
10      情, 提供贵州茅台(600519)的历史行情等信息">
11      <link type="text/css" rel="stylesheet" href="//s1.biz.itc.cn
12        /cn/css/BIZ_comm-1.4.2.css?000" media="screen" />
13      <link type="text/css" rel="stylesheet" href="//k.sohu.com/
14        static/finance/pc/qstock/v0.0.12/css/BIZ_sec-1.4.1.css"
15        media="screen" />
16
17      <script type="text/javascript">
18        /* 文件生成时写入区域 START */
19        biz_Code = "cn_600519";
20        //正常状态: 0, 选中状态: 1, 无效置灰状态: -1
21        //上市股票
22        biz_leftMenuConfig = [[0],[0, 0, 0, 0, 0,
23          0],[0],[0, 0, 0, 0, 0],[0, 0, 0,
24          0],[0, 0, 0, 0, 0]];
25
26        biz_leftMenuConfig[1][3]=1;
27        biz_middMenuConfig =
28          [0,0,0,1,-1,0];
29        /* 文件生成时写入区域 END */
30
31      </script>
32      <!-- 头部js START -->
33      <script type="text/javascript" src="//s1.biz.itc.cn/cn/
34        script/lib/jquery-1.7.2.js"></script>
35      <script type="text/javascript" src="//s4.biz.itc.cn/cn/
36        script/mystock/JCalendar-1.js"></script>
37      <script type="text/javascript">

```

```

26     var BIZ_menu_config = { nav: 1 };
27     commet_obj = {};
28     var loadEvents = function(){
29         //行情
30         var ml1 = new jaw.commets();
31         commet_obj = ml1;
32         ml1.append("hq2", 25, PEAK.getHqURL(2));
33         ml1.handler();
34
35         var url = PEAK.BIZ.HOST + "/hisHq?code=" + biz_Code + "&stat=" +
36             "1&order=D&period=d&callback=historySearchHandler&rtn=jsonp";
37         jaw.evalScript({url: url});
38
39         var jCalendar = new JCalendar();
40         jCalendar.bind("BIZ_lshq_sd", "bottom");
41         jCalendar.bind("BIZ_lshq_ed", "bottom");
42     }
43 </script>
44 <script type="text/javascript" src="//k.sohu.com/static/
45     finance/pc/qstock/v0.0.10/js/main/autocomplete.min.js"></
46     script>
47 <script type="text/javascript" src="//k.sohu.com/static/
48     finance/pc/qstock/v0.0.10/js/main/main-1.4.7.min.js"></
49     script>
50 <script type="text/javascript" src="//k.sohu.com/static/
51     finance/pc/qstock/v0.0.10/js/main/hq_sec-1.4.min.js"></
52     script>
53 <style style="text/css">
54     #calendar_container { width:160px; border:1px solid #06C}
55     #calendar { border-collapse:collapse; background-color:#FFF;
56         width:160px; height:120px; margin:0px auto}
57     #calendar td {font-size:12px; text-align:center; vertical-
58         align:middle; font-family:"宋体"}
59     #calendar thead {background-color:#999; color:#FFF}

```

```

#calendar thead tr td{font:bold; height:20px;}
52 #calendar caption {background-color:#06C; height:20px; padding
:3px 0 0 2px; *padding:5px 0 0 3px}
#calendar a{color:#F90; margin:0 5px; text-decoration:none}
54 #calendar #prev_month,#calendar #next_month {font-size:18px;
margin:0}
#calendar #c_today {background-color:#036; color:#FFF}
#calendar span.arrowL {background:#EEF5FF; width:14px; height
:14px; border:1px #07C solid; position:absolute; top:3px;
left:12px; cursor:pointer}
#calendar span.arrowL em {*font-size:1px; width:0; height:0;
border:5px #EEF5FF solid; border-right:5px #07C solid;
position:absolute; top:2px; *top:0; left:0; *border:5px #
EEF5FF solid; *border-right:4px #07C solid}
56 #calendar span.arrowR {background:#EEF5FF; width:14px; height
:14px; border:1px #07C solid; position:absolute; top:3px;
left:132px; cursor:pointer}
#calendar span.arrowR em {*font-size:1px; width:0; height:0;
border:5px #EEF5FF solid; border-left:5px #07C solid;
position:absolute; top:2px; *top:1px; left:4px; *left:3px; *
border:4px #EEF5FF solid; *border-left:5px #07C solid}
#calendar .over {background-color:#CCC}
#calendar .keydate {color:#06F}
60 </style>

62

64 <!-- 头部js END -->
</head>

66 <body>

68 <!-- 页头 START -->
70 <!-- 搜狐通用页眉A START -->
<div id="criterionNav" class="Area_w">
72 <a target="_blank" href="//www.sohu.com/" id="sohu_logo"></a>
<a target="_blank" href="//stock.sohu.com/" id="sohu_sec_logo"></a>
74
76    <div id="criterionNav_right" class="Area">
77        <ul class="right">
78            <li><a target="_top" href="//www.sohu.com/">搜狐首页
79                </a></li>
80            <li></li>
81            <li class="red"><a target="_top" href="//news.sohu.com/">新闻</a></li>
82            <li></li>
83            <li><a target="_top" href="//sports.sohu.com/">体育
84                </a></li>
85            <li></li>
86            <li><a target="_top" href="//s.sohu.com/">S</a></li>
87            <li></li>
88            <li><a target="_top" href="//yule.sohu.com/">娱乐</a>
89                </li>
90            <li></li>
91            <li><a target="_top" href="//v.sohu.com/">V</a></li>
92            <li></li>
93            <li class="red"><a target="_top" href="//business.sohu.com/">财经</a></li>
94            <li></li>
95            <li><a target="_top" href="//it.sohu.com/">IT</a></li>
96            <li></li>
97            <li><a target="_top" href="//auto.sohu.com/">汽车</a>
98                </li>
99            <li></li>
100           <li><a target="_top" href="//house.sohu.com/">房产</a></li>
```

```
96    <li></li>
97    <li><a target=_top href="/home.sohu.com/">家居</a
98        ></li>
99    <li></li>
100   <li><a target=_top href="/women.sohu.com/">女人</a
101      ></li>
102   <li></li>
103   <li><a target=_top href="/tv.sohu.com/">视频</a
104      ></li>
105   <li></li>
106   <li><a target=_top href="/v.blog.sohu.com/">播客
107      </a></li>
108   <li></li>
109   <li><a target=_top href="/www.chinaren.com/">
110      ChinaRen</a></li>
111   <li></li>
112   <li><a target=_top href="/login.mail.sohu.com/">
113      邮件</a></li>
114   <li></li>
115   <li><a target=_top href="/blog.sohu.com/">博客</a
116      ></li>
117   <li></li>
118   <li><a target=_top href="/comment2.news.sohu.com/
119      ">我说两句</a></li>
120   <li class=end><a target=_top href="/www.sogou.
        com/">搜狗</a></li>
121   </ul>
122   </div>
123
124 </div>
125 <!-- 搜狐通用页眉A END -->
```

```

122    <!-- 行情中心页眉 START -->
123    <div class="BIZ_header">
124        
127        <map name="BIZ_logo">
128            <area shape="rect" coords="0,0,110,46" href="//stock
129                .sohu.com/" target="_blank"></area>
130            <area shape="rect" coords="110,0,200,46" href="//q.
131                stock.sohu.com/" target="_blank"></area>
132        </map>
133        <div id="BIZ_ver">
134            <!-- <a style="padding-right:30px;color:#f00" href="//
135                stock.sohu.com/20130428/n374361532.shtml" target="
136                _blank">*声明：由于系统调整，暂停美股港股行情服务</a
137                > -->
138            <a target="_blank" href="//q.stock.sohu.com/feedback
139                .html">意见反馈</a>
140            <a target="_blank" href="//stock.sohu.com/upload/
141                stock_mobile.html" style="color:#18479B;">手机随
142                时随地看行情</a>
143        </div>
144
145        <!-- 顶部功能栏 START -->
146        <ul id="BIZ_fnbarA" class="BIZ_fnbarA">
147            <li class="e1" c=0><a href="javascript:addBookmark()
148                ;">加入收藏夹</a></li>
149            <li class="e2" c=1><a href="javascript:setHomepage()
150                ;">设为首页</a></li>
151        </ul>
152        <!-- 顶部功能栏 END -->
153
154        <!-- 行情中心主导航 START -->
155        <style type="text/css">

```

```

144     div.BIZ_header div.BIZ_nav ul li{margin-right:0}
145     </style>
146     <div id="BIZ_nav" class="BIZ_nav">
147         <ul style="width:900px; margin-left:135px">
148             <li>首 页<a href="/index.shtml">首 页</a></li>
149             <li>个 股<a href="/cn/000002/index.shtml">个 股
150                 </a></li>
151             <li>指 数<a href="/cn/zs.shtml">指 数</a></li>
152             <li>排 行<a href="/cn/ph.shtml">排 行</a></li>
153             <li>板 块<a href="/cn/bk.shtml">板 块</a></li>
154             <li>自选股<a style="background: url(//stock.sohu.
155                 com/upload/mystock2012/html/skin/images/new2.
156                 gif) no-repeat" href="/cn/mystock.shtml">自选
157                 股</a></li>
158             <!-- <li>千股千评<a href="//t.stock.sohu.com">千股
159                 千评</a></li>-->
160             <!-- <li>炒股大赛<a href="//q.stock.sohu.com/cgds/"'
161                 target="_blank" style="color:#f60">炒股大赛</a
162                 ></li>-->
163         </ul>
164         <div class="BIZ_update_info" style="display:none"></
165             div>
166             <div class="BIZ_nav_border"></div>
167         </div>
168     </div>
169     <!-- 行情中心页眉 END -->
170
171     <!-- 页头 END -->
172
173     <!-- 行情中心主栏 START -->
174     <style type="text/css">
175         div#FEP_searchbar{left:15px}
176         div.BIZ_bar_wrapper div.BIZ_bar div.BIZ_login ul.off li.
177             fn{width:50px;}
178         div.BIZ_bar_wrapper div.BIZ_bar div.BIZ_login ul.on li.

```

```

    caption { margin-right:10px }
170 div.BIZ_bar_wrapper div.BIZ_bar div.BIZ_login{ left :auto ;
    right :10px; background: url('//i1.itc.cn/20120920/2
    bb1_e4c60ac2_b96d_b596_71aa_d67fed8c8861_1.png) no-
    repeat ; _background : transparent ; _filter : progid :
    DXImageTransform . Microsoft . AlphaImageLoader ( enabled= ,
    true ,sizingMethod='crop' ,src='//i1.itc.cn/20120920/2
    bb1_e4c60ac2_b96d_b596_71aa_d67fed8c8861_1.png' ) }

</style>
172 <div class="BIZ_bar_wrapper">
    <div id="BIZ_bar" class="BIZ_bar">
        <!--
174     <span class="BIZ_user"></span>
        行情中心登陆元素 START -->
        <div id="BIZ_login" class="BIZ_login"></div>
        <!-- 行情中心登陆元素 END -->

180     <!-- 搜索&Suggest START -->
        <div id="FEP_searchbar" class="searchbar suggestRoot
            clearfix">
182         <form action="javascript:void(0)" id="searchForm
            ">
            <ul id="FEP_searchList" class="searchList
                clearfix">
184             <li class="e1"><input id="searchInput"
                type="text" autoComplete="off"
                disableautocomplete /></li>
                <li class="e2"><input id="FEP_searchBtn"
                    type="submit" class="suggest_btn"
                    value="" /></li>
            </ul>
186         </form>
188         <div id="suggestDiv" class="suggestLists" style=
            "display: none; "></div>
    </div>

```

```

190         <!-- 搜索&Suggest END -->
191         </div>
192         <div class="BIZ_bar_border"></div>
193     </div>
194     <!--<div class="flash" style="width:980px; margin:0 auto 10px
195         ">
196         <a href="//q.stock.sohu.com/cgds/index.do" target="
197             _blank"></a>
199     </div>-->
200
201     <!-- 行情中心主栏 END -->
202
203     <div class="str2Column clearfix">
204         <div class="str2ColumnL">
205             <!-- 行情中心主菜单 START -->
206             <div class="BIZ_menu_shadow">
207                 <div id="BIZ_stock_list" class="BIZ_stock_list">
208                     <div class="BIZ_tabA">
209                         <ul class="clearfix" id="FTag">
210                             <li id="ft0" class="current" c="BIZ_MyLBS"><span
211                                 >最近浏览股</span></li>
212                             <li id="ft1" c="BIZ_Mystock"><span><a href="//q.
213                                 stock.sohu.com/cn/mystock.shtml">我的自选股</
214                                 a></span></li>
215                         </ul>
216                     </div>
217                     <table id="BIZ_MyLBS">
218                         <tr><td width="50px"><span>&nbsp;</span></td><td></
219                             td></tr>
220                         <tr><td><span>&nbsp;</span></td><td></td></tr>
221                         <tr><td><span>&nbsp;</span></td><td></td></tr>
222                         <tr><td><span>&nbsp;</span></td><td></td></tr>
223                         <tr class="last"><td><span>&nbsp;</span></td><td></td></tr>

```

```

    </table>
218   <table id="BIZ_Mystock" style="display:none">
219     <tr><td width="50px"><span>&ampnbsp</span></td><td></td></tr>
220     <tr><td><span>&ampnbsp</span></td><td></td></tr>
221     <tr><td><span>&ampnbsp</span></td><td></td></tr>
222     <tr><td><span>&ampnbsp</span></td><td></td></tr>
223     <tr class="last"><td><span>&ampnbsp</span></td><td></td></tr>
224   </table>
225 </div>
226
227 <div class="BIZ_menu_border">
228   <div id="BIZ_menu" class="BIZ_menu">
229     <script>biz_Name = "贵州茅台";
230     var status = "0";</script>
231     <div class="part first">
232       <ul>
233         <li><a href="//q.stock.sohu.com/cn/600519/index.
234           shtml"><b>贵州茅台</b></a></li>
235       </ul>
236     </div>
237     <div class="part">
238       <h3>行情图表</h3>
239       <ul>
240         <li class="tuijian_li" style="display:none"><a
241           href="//q.stock.sohu.com/qp/index.html?
242             cn_600519">实时行情</a><span class="tuijian">
243               推荐</span></li>
244         <li><a href="//q.stock.sohu.com/cn/600519/cjmx.
245           shtml">成交明细</a></li>
246         <li><a href="//q.stock.sohu.com/cn/600519/fjb.
247           shtml">分价表</a></li>
248         <li><a href="//q.stock.sohu.com/cn/600519/lshq.
249           shtml">历史行情</a></li>

```

```
244      <li><a href="http://q.stock.sohu.com/cn/600519/lhb.shtml">龙虎榜数据</a></li>
246      <li><a href="http://q.stock.sohu.com/cn/600519/dzjy.shtml">大宗交易</a></li>
248      </ul>
250      </div>
252      <div class="part">
254          <h3>新闻资讯</h3>
256          <ul>
258              <!-- <li><a href="http://q.stock.sohu.com/cn/600519/news_gs.shtml">公司新闻</a></li> -->
260              <li><a href="http://q.stock.sohu.com/cn/600519/gsgg.shtml">公司公告</a></li>
262              <!-- <li><a href="http://q.stock.sohu.com/cn/600519/news_gg.shtml">个股研究</a></li> -->
264              <!-- <li><a href="http://q.stock.sohu.com/cn/600519/news_hy.shtml">行业新闻</a></li> -->
266              <!-- <li><a href="http://q.stock.sohu.com/cn/600519/news_xg.shtml">相关新闻</a></li> -->
268              <!-- <li><a href="http://q.stock.sohu.com/cn/600519/news.shtml">个股新闻</a></li> -->
270              <!-- <li><a href="http://q.stock.sohu.com/cn/600519/pl.shtml">分析师评论</a></li> -->
272          </ul>
274      </div>
276      <div class="part">
278          <h3>公司概况</h3>
280          <ul>
282              <li><a href="http://q.stock.sohu.com/cn/600519/gsjj.shtml">公司简介</a></li>
284              <li><a href="http://q.stock.sohu.com/cn/600519/gbjg.shtml">股本结构</a></li>
286              <li><a href="http://q.stock.sohu.com/cn/600519/glcc.shtml">管理层</a></li>
```

```
266      <li><a href="/q.stock.sohu.com/cn/600519/jyqk .  
267          shtml">经营情况简介</a></li>  
268      <li><a href="/q.stock.sohu.com/cn/600519/bw .  
269          shtml">重大事项备忘</a></li>  
270      <li><a href="/q.stock.sohu.com/cn/600519/fhsp .  
271          shtml">分红送配记录</a></li>  
272      </ul>  
273  </div>  
274  <div class="part">  
275      <h3>持仓明细</h3>  
276      <ul>  
277          <li><a href="/q.stock.sohu.com/cn/600519/zyg .  
278              shtml">主要股东</a></li>  
279          <li><a href="/q.stock.sohu.com/cn/600519/ltdg .  
280              shtml">流通股股东</a></li>  
281          <li><a href="/q.stock.sohu.com/cn/600519/jjcc .  
282              shtml">基金持仓</a></li>  
283          <li><a href="/q.stock.sohu.com/cn/600519/xsjj .  
284              shtml">限售股解禁表</a></li>  
285      </ul>  
286  </div>  
287  <div class="part last">  
288      <h3>财务数据</h3>  
289      <ul>  
290          <li><a href="/q.stock.sohu.com/cn/600519/cwzb .  
291              shtml">重要财务指标</a></li>  
292          <li><a href="/q.stock.sohu.com/cn/600519/srgc .  
293              shtml">主营收入构成</a></li>  
294          <li><a href="/q.stock.sohu.com/cn/600519/zcfz .  
295              shtml">资产负债表</a></li>  
296          <li><a href="/q.stock.sohu.com/cn/600519/xjll .  
297              shtml">现金流量表</a></li>  
298          <li><a href="/q.stock.sohu.com/cn/600519/lr .  
299              shtml">利润表</a></li>  
300          <li><a href="/q.stock.sohu.com/cn/600519/yjyg .
```

```

288           shtml">业绩预告</a></li >
289     <li><a href="http://q.stock.sohu.com/cn/600519/cwbg.
290           shtml">财务报告</a></li >
291   </ul>
292 </div>

292           </div>
293           </div>
294         </div>
295         <!-- 行情中心主菜单 END -->
296       </div>
297       <div class="str2ColumnR">
298         <!-- 行情中心报价区域 START -->
299       <div class="BIZ_IS_price_shadow">
300         <div class="BIZ_IS_price_border">
301           <div id="BIZ_IS_price_A1" class="BIZ_IS_priceA">
302             <div class="BIZ_IS_price_id">
303               <a id="BIZ_IS_Name" href="http://q.stock.
304                 sohu.com/cn/600519/index.shtml">贵州茅台
305               </a>
306             <span> ( 600519 ) </span>
307           </div>
308           <!-- price START -->
309             <ul class="BIZ_IS_price_A">
310               <li class="e1" c=2></li >
311               <li class="e2" c=3></li >
312               <li class="e3" c=4></li >
313             </ul>
314             <div class="BIZ_IS_price_TS" id="
315               BIZ_time"></div>
316             <!-- 加入自选股功能栏 START -->
317             <div class="BIZ_fnbarB">
318               <ul>
319                 <li class="e1"><a href="index.

```

```
318         shtml">实时行情</a></li>
319     <li class="e2"><a href="
320         javascript :addMyStock( ) ; ">加
321         入自选股</a></li>
322     </ul>
323     <div id="BIZ_myStockList" class="e2"
324         style="display :none ; "></div>
325     </div>
326     <!-- 加入自选股功能栏 END -->
327
328     <!-- price END -->
329         </div>
330     </div>
331     </div>
332     <!-- 行情中心报价区域 END -->
333     <div class="BIZ_innerMain">
334         <div class="BIZ_tabC">
335             <ul id="BIZ_tabC">
336                 <!--
337                 <li><a href="/cn/600519/index_kp.shtml">
338                     实时行情</a></li>
339                 <!--
340                 <li><a href="/cn/600519/cjmx.shtml">成交
341                     明细</a></li>
342                 <li><a href="/cn/600519/fjb.shtml">分价
343                     表</a></li>
344                 <li class="current"><a href="/cn/600519/
345                     lshq.shtml">历史行情</a></li>
346                 <li><a href="/cn/600519/lhb.shtml">龙虎
347                     榜数据</a></li>
348                 <li><a href="/cn/600519/dzjy.shtml">大宗
349                     交易</a></li>
350             </ul>
351         </div>
352         <div class="BIZ_innerBoard">
```

```
344 <h2>历史行情</h2>
345 <div class="BIZ_innerContent">
346   <div class="innerHeaderA">
347     <form class="formA" name="historyHqForm" action="javascript
348       : void(0)">
349       <input type="hidden" name="code"
350         value="cn_000002">
351       从<input class="ipt" type="text" name="sd" id="BIZ_lshq_sd" value="" size="20">
352       至 <input class="ipt" type="text" name="ed" id="BIZ_lshq_ed" value="" size="20">
353       <input type="radio" checked name="t" value="d">按日 <input type="radio" name="t" value="w">按周 <input type="radio" name="t" value="m">按月
354       <input type="button" value="查询" onclick="historyHqSearch(this.form)">
355     </form>
356   </div>
357   <div class="part">
358     <table class="tableQ" id="BIZ_hq_historySearch">
359       <thead>
360         <tr style="display:none">
361           <td class="sum">累计:</td><td class="date" colspan=2></td><td></td><td></td>
```

```

><td></td><td></td>
      <td></td><td></td>
</tr>
<tr style="display:none">
      <td class="e1"></td><td>
      ></td><td></td><td></td>
      >105.91</td><td>
      >105.91</td><td></td>
      ><td></td><td></td>
</tr>
<tr class="bgGray1">
      <th class="e1">日期</th>
      <th class="e2">开盘</th>
      <th class="e3">收盘</th>
      <th class="e4">涨跌额</th>
      <th class="e5">涨跌幅</th>
      <th class="e6">最低</th>
      <th class="e7">最高</th>
      <th class="e8">成交量(手)</th>
      <th class="e9">成交金额(万)</th>
      <th>换手率</th>
</tr>
</thead>
<tbody></tbody>
</table>
<div style="padding-top:5px">*注：每次查询最多显示100条</div>
</div>
</div>
</div>

```

```

382         </div>
383     </div>
384     <div class="foot"></div>
385   </div>
386
387   <!-- 行情中心页脚 START -->
388   <div id="BIZ_footer" class="BIZ_footer">
389     <a href="javascript:void(0)" onClick="this.style.
390       behavior='url(#default#homepage)';this.setHomePage
391       ('//www.sohu.com');return false;">设置首页</a>
392     - <a href="//q.stock.sohu.com/sitemap.shtml" target="
393       _blank">站点地图</a>
394     - <a href="//pinyin.sogou.com/" target="_blank">搜狗输入
395       法</a>
396     - <a href="//up.sohu.com/" target="_blank">支付中心</a>
397     - <a href="//hr.sohu.com" target=_blank>搜狐招聘</a>
398     - <a href="//ad.sohu.com/" target=_blank>广告服务</a>
399     - <a href="//sohucallcenter.blog.sohu.com/" target="
400       _blank">客服中心</a>
401     - <a href="//corp.sohu.com/s2006/contactus/" target="
402       _blank">联系方式</a>
403     - <a href="//www.sohu.com/about/privacy.html" target="
404       _blank">保护隐私权</a>
405     - <a href="//corp.sohu.com/" target="_blank">About SOHU
406       </a>
407     - <a href="//corp.sohu.com/indexcn.shtml" target="_blank
408       ">公司介绍</a>
409     <br />Copyright <span class="cr">&copy; </span> 2022 Sohu
410       .com Inc. All Rights Reserved. 搜狐公司 <span class=""
411       unline"><a href="//corp.sohu.com/s2007/copyright/"
412       target="_blank">版权所有</a></span>
413     <br />搜狐不良信息举报电话：010-62728061 举报邮箱：<a
414       href="mailto:jubao@contact.sohu.com">jubao@contact.
415       sohu.com</a>
416   </div>

```

```
404    <!-- START WRating v1.0 -->
405    <!--
406    <script type="text/javascript" src="https://dsl.wrating.com/
407        a1.js">
408    </script>
409    <script type="text/javascript">
410        var vjAcc="860010-0626010000";
411        var wrUrl="https://dsl.wrating.com/";
412        vjTrack("");
413    </script>
414    <noscript></noscript>
416    —>
417    <!-- END WRating v1.0 -->
418
419    <script type="text/javascript" src="//js.sohu.com/mail/pv/
420        pv_v203_ajax.js"></script>
421    <!--
422    <script type="text/javascript">
423        if(typeof jaw != 'undefined'){
424            (new Image).src = '//stat.stock.sohu.com/qstock_v.gif?
425                SUV=' + jaw.cookie.get("SUV") + '&' + Math.random();
426        }
427    </script>
428    —>
429
430    <!-- 行情中心页脚 START -->
431
432    <script type="text/javascript" src="//k.sohu.com/static/
433        finance/pc/tongji/tongji.js"></script>
434
435    </body>
436    </html>
```

股票代码 300117 的界面 html 源码

在表格标签 tbody 中没有数据，但是在浏览器中却显示了数据，这是因为浏览器会自动补全表格，所以在爬取数据的时候，需要注意这一点。

7.2 实验源码

```
1 import requests ,re ,json
2 import bs4
3 from bs4 import BeautifulSoup
4 import pandas as pd
5 import pymysql
6 # import matplotlib.pyplot as plt
7 from pyecharts.charts import Bar , Kline , EffectScatter , Line
8 from pyecharts import options as opts
9 import wx
10 import wx.grid
11 from selenium import webdriver
12 from selenium.webdriver.common.by import By
13 import time
14 import webbrowser
15
16
17 # 获取网页HTML代码
18 from textdistance import Overlap
19
20
21 # def getHtml(stack_code):
22 #     data = requests.get("https://q.stock.sohu.com/cn/" +
23 #                         stack_code + "/lshq.shtml",
24 #                         headers={"user-agent": "Mozilla/5.0 (Windows NT 10.0;
25 #                         Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
26 #                         /112.0.0.0 Safari/537.36 Edg/112.0.1722.58"})
```

```
25      #     # 获得一个请求得到的静态网页
26      #     return data.text
27
28
29      # 将HTML文本转化成字典列表
30      # def getData(data):
31      #     month_data = []
32      #     soup = BeautifulSoup(data, "lxml") # data由getHtml()
33      #         取得
34      #         # 爬取网页的工具
35      #         list = soup.find("div", class_="part").find("table",
36      #             class_="tableQ")
37      #         # 看html, 已经把不需要的都删了, 数据在innerbox的
38      #         table_bg001 border_box limit_sale下面
39      #         dataList = list.find_all("tr")[1:]
40      #         # 因为第一个tr后面是一堆th标签, 不需要
41      #         f = open("data.txt", "w", encoding="utf-8")
42      #         for item in dataList:
43      #             kv = {}
44      #             if isinstance(item, bs4.element.Tag):
45      #                 tdList = item.find_all("td")
46      #                 # print(tdList)
47      #                 kv["日期"] = tdList[0].text # 去掉td和/td, 取
48      #                     中间的内容
49      #                 kv["开盘价"] = tdList[1].text
50      #                 kv["最高价"] = tdList[6].text
51      #                 kv["最低价"] = tdList[5].text
52      #                 kv["收盘价"] = tdList[2].text
53      #                 kv["成交量"] = tdList[7].text
54      #                 month_data.append(kv)
55      #             to_write = json.dumps(kv, ensure_ascii=False)
56      #             f.write(to_write)
57      #             # print(kv)
58      #             f.close()
59      #     # print(month_data) # 字典列表
```

```

#      return month_data
57
def getData(stack_code):
59    month_data = []
60    driver = webdriver.Chrome()
61    driver.get('http://q.stock.sohu.com/cn/'+stack_code+'/'
62              'lshq.shtml')
63    time.sleep(5)
64
65    f=open("data.txt","w",encoding="utf-8")
66
67    for i in range(2, 81):
68        kv={}
69        try:
70            kv["日期"] = driver.find_element(By.XPATH, '//*[@'
71                        '@id="BIZ_hq_historySearch"]/tbody/tr['
72                        '+str(i)+']/td[1]').text
73            kv["开盘价"] = driver.find_element(By.XPATH, ''
74                        '//*[@[@id="BIZ_hq_historySearch"]/tbody/tr['
75                        '+str(i)+']/td[2')).text
76            kv["最高价"] = driver.find_element(By.XPATH, ''
77                        '//*[@[@id="BIZ_hq_historySearch"]/tbody/tr['
78                        '+str(i)+']/td[7')).text
79            kv["最低价"] = driver.find_element(By.XPATH, ''
80                        '//*[@[@id="BIZ_hq_historySearch"]/tbody/tr['
81                        '+str(i)+']/td[6')).text
82            kv["收盘价"] = driver.find_element(By.XPATH, ''
83                        '//*[@[@id="BIZ_hq_historySearch"]/tbody/tr['
84                        '+str(i)+']/td[3')).text
85            kv["成交量"] = driver.find_element(By.XPATH, ''
86                        '//*[@[@id="BIZ_hq_historySearch"]/tbody/tr['
87                        '+str(i)+']/td[8')).text
88            print("complete")
89        except:
90            print("找不到元素")

```

```
        continue
79         month_data.append(kv)
    to_write = json.dumps(kv, ensure_ascii=False)
81         f.write(to_write)
f.close()
# driver.close()
driver.quit()
85     return month_data

87 # 从字典列表取出数据存入数据库中
def Storage(data):
    # 连接数据库
    my_database = pymysql.connect(
        host='localhost',
        user='root',
93         password='986370165',
    )
    # 生成游标对象
    cursor = my_database.cursor()
    # 创建数据库
    sql_createDataBase = "create database if not exists
stockData"
    cursor.execute(sql_createDataBase)
    sql_useDataBase = "USE stockData"
    # 创建表
    cursor.execute(sql_useDataBase)
    sql_createTable = '''create table if not exists data(
        date DATE,
        opening_price float,
        closing_price float,
        highest float,
        lowest float)
    ''',
    cursor.execute(sql_createTable)
    # 从字典列表中取出数据存入数据库中
```

```

    for item in data:
        sql_Insert = '''Insert into data values
        ({0},{1},{2},{3},{4})'''.format(item['日期'], item
                                         ['开盘价'], item['收盘价'], item['最高价'], item[
                                         '最低价'])
        cursor.execute(sql_Insert)
    cursor.close()
    my_database.commit()
    my_database.close()
    # print(my_database)

date = []
opening_price = []
closing_price = []
highest = []
lowest = []

# 为画K线图做数据处理，将五个数据分别放进五个列表中
def data_Pretreatment(month_data):
    for data in month_data:
        my_date = data.get("日期")
        my_open = data.get("开盘价")
        my_close = data.get("收盘价")
        my_high = data.get("最高价")
        my_low = data.get("最低价")
        date.append(my_date)
        opening_price.append(my_open)
        closing_price.append(my_close)
        highest.append(my_high)
        lowest.append(my_low)

def draw(date,lowest):

```

```

145     bar1 = Bar()
146     bar1.add_xaxis(date)
147     bar1.add_yaxis("最低价", lowest)
148     bar1.set_series_opts(
149         # 是否显示标签
150         label_opts=opts.LabelOpts(is_show=False)
151         , markpoint_opts=opts.MarkPointOpts(
152             data=[opts.MarkPointItem(type_="max", name="max")
153                 ,
154                 opts.MarkPointItem(name="min", type_="min"
155                     )]
156         ),
157         markline_opts=opts.MarkLineOpts(data=[opts.
158             MarkLineItem(name="average", type_="average")]))
159     bar1.set_global_opts(
160         xaxis_opts=opts.AxisOpts(
161             axislabel_opts=opts.LabelOpts(rotate=-60,
162                 font_size=10),
163         ),
164         yaxis_opts=opts.AxisOpts(
165             name="价格: (元/股)",
166         ),
167     )
168     bar1.render("最低价.html")

169 def draw_K():
170     kline = Kline().set_global_opts(title_opts=opts.
171         TitleOpts(title="K线图"))
172     v1 = []
173     size = len(opening_price) # 有多少条数据 (多少天)
174     for i in range(size-1, 0, -1):
175         tmp = [opening_price[i], closing_price[i], lowest[i]
176             , highest[i]]

```

```

    v1.append(tmp) # 整合好一组数据存入v1中
175   # print(v1)

177   kline.add_yaxis(series_name="日K", y_axis=v1)
178   kline.add_xaxis([s for s in date])
179   kline.render("K图.html")

181
182 def store_in_dataframe(month_data):
183     my_list = []
184     for item in month_data:
185         tmp = list(item.values())
186         my_list.append(tmp)
187     # print(my_list)
188     my_dataframe = pd.DataFrame(my_list,
189         columns=["datetime", "open", "close", "low", "high",
190                  "trade_sum"])
191     print(my_dataframe)
192     my_dataframe.to_excel("数据.xlsx")
193     return my_dataframe

194
195 def back_testing_plot(table_name, indicator_name_list):
196     # data preparation
197     da = pd.DataFrame(data=table_name)
198     # da['trade_sum'] = da['trade_sum'].apply(lambda vol:
199         vol if vol > 0 else 0)
200     date = da["datetime"].apply(lambda x: str(x)).tolist()
201     k_plot_value = da.apply(lambda record: [record['open'],
202                             record['close'],
203                             record['low'],
204                             record['high']],
205                             axis=1).tolist()

```

```

207     # K chart
208     kline = Kline()
209     kline.add("Back_testing_Result", date, k_plot_value)
210
211     indicator_lines = Line()
212     for indicator_name in indicator_name_list:
213         indicator_lines.add(indicator_name,
214             date,
215             da[indicator_name].tolist(),
216             mark_point=["max", "min"],
217             )
218
219     # trading volume bar chart
220     bar = Bar()
221     print(type(max(da["trade_sum"])))
222     bar.add("trade_sum", date, da["trade_sum"],
223         tooltip_tragger="axis",
224         is_legend_show=False,
225         is_yaxis_show=False,
226         yaxis_max=5 * max(da["trade_sum"]),
227         )
228
229     # buy and sell
230     v1 = date[10]
231     v2 = da['high'].iloc[10]
232     es = EffectScatter("buy")
233     es.add("buy", [v1], [v2])
234     v1 = date[18]
235     v2 = da['high'].iloc[18]
236     es.add("sell", [v1], [v2], symbol="pin", )
237
238     overlap = Overlap()
239     overlap.add(kline)
240     overlap.add(indicator_lines, )
241     overlap.add(bar)
242     overlap.add(es)
243     overlap.render(path='高级图.html')

```

```

243
244     class MyFrame(wx.Frame):
245         data = []
246         column_names = []
247         stock_Code = ""
248
249         def __init__(self, data, column_names):
250             super().__init__(parent=None, title="股票数据显示界面", size=(600, 600))
251             self.data = data
252             self.column_names = column_names
253             self.Centre()
254             panel = wx.Panel(parent=self)
255             # self.message1 = wx.StaticText()
256             # self.message1.SetLabelText("请输入股票代码")
257             # self.message1.SetPosition((400,370))
258             self.number = wx.TextCtrl(panel, pos=(450, 370))
259             query_button = wx.Button(parent=panel, id=1, label='查询K图', pos=(450, 400))
260             post = wx.Button(parent=panel, id=2, label='更新', pos=(450, 435))
261             show = wx.Button(parent=panel, id=3, label='查看表格', pos=(450, 470))
262             self.Bind(wx.EVT_BUTTON, self.on_click, query_button)
263             self.Bind(wx.EVT_BUTTON, self.on_click, post)
264             self.Bind(wx.EVT_BUTTON, self.on_click, show)
265             # self.Bind(wx.EVT_TEXT, self.EvtText)
266             # 建立表格
267
268         def generate_xlsx(self):
269             self.grid = self.CreateGrid(self)
270             self.Bind(wx.grid.EVT_GRID_LABEL_LEFT_CLICK, self.OnLabelLeftClick)

```

```

271
272     def on_click(self, event):
273         event_id = event.GetId()
274         print(event_id)
275         if event_id == 1:
276             print("查询K图")
277             webbrowser.open('K图.html')
278         elif event_id == 2:
279             self.stock_Code = self.number.GetValue()
280             print(self.number.GetValue())
281             update_xlsx(self, self.stock_Code)
282         elif event_id == 3:
283             self.generate_xlsx()

284
285     def OnLabelLeftClick(self, event):
286         print("RowIndex: {}".format(event.GetRow()))
287         print("ColIdx: {}".format(event.GetCol()))
288         print(self.data[event.GetRow()])
289         event.Skip()

290
291     def CreateGrid(self, parent):
292         grid = wx.grid.Grid(parent)
293         grid.CreateGrid(len(self.data), len(self.data[0]))

294
295         for row in range(len(self.data)):
296             for col in range(len(self.data[row])):
297                 grid.SetColLabelValue(col, self.column_names
298                               [col])
299                 grid.SetCellValue(row, col, self.data[row][
300                               col])
301
302         # 设置行和列自定调整
303         grid.AutoSize()

304
305         return grid

```

```
305     class App(wx.App):
306         data = []
307         column_names = []
308
309         def show(self):
310             frame = MyFrame(self.data, self.column_names)
311             frame.Show()
312             return True
313
314
315         def update_xlsx(app, stock_code):
316             month_data = getData(stock_code)
317             Storage(month_data)
318             store_in_dataframe(month_data)
319             my_list = []
320             for item in month_data:
321                 tmp = list(item.values())
322                 my_list.append(tmp)
323             app.data = my_list
324             update(month_data)
325
326
327         def update(month_data):
328             date.clear()
329             opening_price.clear()
330             closing_price.clear()
331             highest.clear()
332             lowest.clear()
333             data_Pretreatment(month_data)
334             draw_K()
335
336
337         data=getData("300117") # 字典列表
```

```
339     date.sort(key=lambda k: k[‘日期’])
340     Storage(data)
341     store_in_dataframe(data)
342     data_Pretreatment(data)
343     draw_K()
344
345     # print(data)
346     app = App()
347     my_list = []
348     for item in data:
349         print(item)
350         tmp = list(item.values())
351         my_list.append(tmp)
352     app.data = my_list
353     app.column_names = [“datetime”, ‘open’, ‘close’, ‘low’, ‘
354         high’, “trade_sum”]
355     print(app.data)
356     app.show()
357     app.MainLoop()
```

实验源码