金融软件工程实验二

201870119 李锵

实验内容简述

本次实验的主要内容为：使用网络爬虫的基础技术爬取股票实时信息，对于爬取的数据进行存储并进行简单的计算。其中用到的库包括

```
import pandas as pd
import requests
import time
import re
```

一、爬虫部分——requests库

为了使网站返回正常的数据信息，首先为网络爬虫定义**请求表头"User-Agent"与"Referer"**

```
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) "
                          "Chrome/98.0.4758.102 Safari/537.36", "Referer":
"https://finance.sina.com.cn/"}
stock_ids = ["sh600010", "sh600519", "sh601318", "sh601857", "sh600036",
"sz000792", "sz002432", "sz002594",
             "sz000723", "sz002104"]
url = "https://hq.sinajs.cn/?format=text&list={}".format(','.join(stock_ids))
```

由于助教老师提供的股票数据网址API直接返回相关的查询数据，故无需利用BeautifulSoup对网页源码进行解析和定位，只要对爬虫获取的response数据进行简单的分割并存储即可。

```
def info_process(response):
    stock_info = response.text.strip().split("\n")
    cur_price = []
    Time = []
    for stock in stock_info:
        temp = stock.strip().split(",")
        cur_price.append(temp[3])
        date = temp[30] + ' ' + temp[31]
        Time.append(date)
    data = {Time[1]: cur_price}
    return pd.DataFrame(data)
```

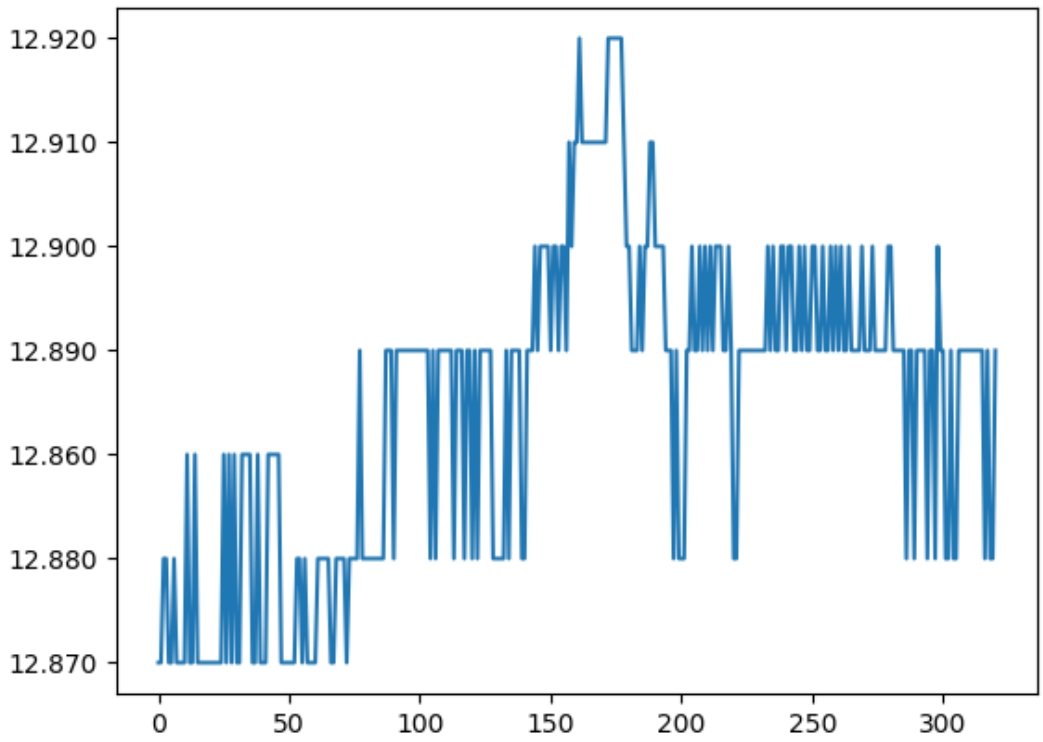
由于中国证券市场股票的价格一般每三秒变动一次，而本次爬虫实验只需要爬取十五分钟内的所有价格变动信息，可以忽略每次爬取间的数据处理时间，**通过*time*库简单设置get_url函数执行间隔为三秒。**但如果要在很长一段时间内持续爬取相关的股票变动数据，则我们需要考虑到数据处理部分的时间开销可能会对爬虫实际爬取的时间间隔造成影响，此问题可以通过Python的多线程*threading*库或者并发异步*asyncio*库等得到有效的解决。

根据爬取的数据进行绘图

```
import matplotlib.pyplot as plt


def fun_plot(i):
    final = pd.read_excel("data.xlsx", header=None)
    final = pd.DataFrame(final.values.T, columns=final.index)
    final.drop(final.head(3).index, inplace=True)
    for i in range(1, 11):
        value = final.iloc[:, i].values.tolist()
        plt.plot(value)
        plt.savefig("./{}.png".format(i))
        plt.show()
    return
```

效果如下



二、数据处理部分——pandas库

为了对股票实时价格数据进行有效的管理，选用了**Dataframe**作为数据存储的基本结构。作为一种表格型数据结构，Dataframe含有一组有序的列，每列可以是不同的值。DataFrame既有行索引，也有列索引，它可以看作是由Series组成的字典，不过这些Series公用一个索引。DataFrame的创建有多种方式，不过最重要的还是根据dict进行创建，以及读取csv或者txt文件来创建。

正则表达式

为了便于以后存储不同时间类型的Dataframe，先提取各股票的详细信息作为考虑到股票相关信息的结构为"*sz600010=包钢股份*"，故利用正则表达式**r's.(.)=(.)'**匹配股票的代码和名称信息。

```
def pd_form():
    response = requests.get(url, headers=headers)
    stock_info = response.text.strip().split("\n")
    stock_name = []
    stock_id = []
    for stock in stock_info:
        temp = stock.strip().split(",")
        ans = re.findall(r's.(.*)=(.*)', temp[0])
        Id = ans[0][0]
        name = ans[0][1]
        stock_id.append(Id)
        stock_name.append(name)
    Stock_data = {'Stock_Id': stock_id, 'Stock_Name': stock_name}
    data = pd.DataFrame(Stock_data)
    return data
```

数据存储的基本结构——Dataframe

通过字符串连接，我们得到了价格变动的时间点信息（例："*2022-02-08 14:05:00*"），但要通过时间对Dataframe进行筛选操作还需要利用pandas库中的to_datetime()函数将其转为标准的时间表示格式

```
final[0] = pd.to_datetime(final[0])  # date转为时间格式
```

筛选Dataframe中所有分钟数为(4+i)的股票价格信息（第一分钟为14:05，故第i分钟在实际表示中为4+i)

```
temp = final[(final[0].dt.minute == 4 + i)]
```

下面以15分钟内10支股票的收益率为例展示Dataframe的创建与拼接：

```
    for i in range(1, 11):
        rate.append(format(round((float(end[i - 1]) - float(begin[i - 1])) /
float(begin[i - 1]), 6), '.4%'))
    new_column = {"Rate": rate} # 为新的属性列先创建一个字典结构
    new_column = pd.DataFrame(new_column) # 将该字典结构转成dataframe结构
    minute_final = pd.concat([minute_final, new_column], axis=1) # 将新属性列与原
dataframe拼接在一起
```

文件IO——to_excel()函数

在使用了Dataframe对数据进行存储和计算等操作后，进行文件的输入输出就变得十分容易：

pandas库提供了**to_excel()**以及**to_csv()**函数将Dataframe保存到本地

to_excel()函数的具体参数如下：

```
DataFrame.to_excel(excel_writer, sheet_name='Sheet1', na_rep='',
float_format=None, columns=None, header=True, index=True, index_label=None,
startrow=0, startcol=0, engine=None, merge_cells=True, encoding=None,
inf_rep='inf', verbose=True, freeze_panes=None)
```

excel_writer	**文件路径或现有的ExcelWriter**
sheet_name	将包含DataFrame的表的名称
na_rep	缺失数据表示方式
float_format	格式化浮点数的字符串
columns	**要编写的列**
header	**写出列名。如果给定字符串列表，则假定它是列名称的别名。**
index	**写行名（索引）**
index_label	如果需要，可以使用索引列的列标签。如果没有给出，标题和索引为true，则使用索引名称。如果数据文件使用多索引，则需使用序列。
startrow	左上角的单元格行来转储数据框
startcol	左上角的单元格列转储数据帧
engine	使用写引擎 - 您也可以通过选项io.excel.xlsx.writer，io.excel.xls.writer和io.excel.xlsm.writer进行设置。
merge_cells	编码生成的excel文件。 只有xlwt需要，其他编写者本地支持unicode。
inf_rep	无穷大的表示(在Excel中不存在无穷大的本地表示)
freeze_panes	指定要冻结的基于1的最底部行和最右边的列

*加粗的参数是作者平常用的参数

三、实验结果文件说明

data.xlsx文件存储的是爬虫每三秒爬取的股票实时价格信息；

minute_data.xlsx文件存储的是每支股票每分钟均值和15分钟内价格变动率的信息。

stock.py为全部源代码

2022-03-01