中国人民大學

RENMIN UNIVERSITY OF CHINA

《Python 数据分析与机器学习》

课程作业报告

姓　　名：何欣桐

学　　号：0000000000

学　　院：商学院

专　　业：工商管理类

授课教师：余　力

2022 年 3 月

目录

摘要

本项目基于《Python 数据分析与机器学习》课程所学知识与网络相关教程资源，对数据集中的基金数据，进行各类数据分析，主要包括：使用 requests、bs4、re、selenium 爬取天天基金网上的基金总表和单个基金的相关信息（代码、名称、价格、类型等），使用 numpy、pandas 对得到的数据进行分析处理（筛选类型、提取需要的信息进行计算等），使用 matplotlib、tkinter、wordcloud 进行数据可视化（云图绘制、曲线绘制、直方图绘制等）。共爬取了 1 张所有基金的总简表、3000 余条基金价格文件，2 份主题基金分类文件，导出 1 张基金价格趋势图，1 张基金名称云图，1 张基金连续涨跌天数直方图。

通过本次课程的学习，我对 Python 的了解更加深入，它能帮助我们解决很多问题，爬虫、可视化工具使我们的数据分析效率大大提升，特别是今年美国大学生数学建模比赛的 C 题，是一个关于如何获得最大利益的问题，其问题背景就是机器学习和量化交易，这和本次课程的作业也很相似，但也更深入，这充分说明了 AI+金融是当前的热点问题。但由于自身原因，未能完成更有难度的一些作业任务，希望今后仍然保持好奇，继续深入学习机器学习相关知识，并努力和自己所学专业相结合，成为学科交叉型人才。

正文

1　数据准备与主要样例函数介绍

根据课程所给的 BigData_course.py 样例文件，可适当修改所给样例函数，使其可以下载数据总简表和各个基金的具体数据。

1.1 crawl_whole_info_table()

该函数从网上抓取基金信息总表。其步骤主要包括：设置 session 信息便于进入网站，随后对于访问的返回信息进行处理读取，将每一项基金存入 fund_list 列表，创建 pandas 数据表 fund_df，最后保存到文件 **data/wholeInfo.xlsx** 中。

实际上，一个基金在网上的总表属性包含 25 项，样例函数**只取前 17 项**属性进行保存（代码、名称、英文、日期、单位净值、累积净值、日增长、近 1 周、近 1 月、近 3 月、近 6 月、近 1 年、近 2 年、近 3 年、今年来、成立来、成立时间）

此外，还需要注意的是，其中有一处遍历循环['hh', 'gp', 'zq', 'zs']: 地址中'ft=zq'代表爬取债券型基金，可根据情况更改为 pg、gp、hh、zs、QDII、LOF（偏股型、股票型、混合型、指数型、QDII 型、LOF 型），所以对最后获得的总表**删除重复值。**

利用该函数保存总简表的源代码如下。

```
if "网上抓取基金信息总表 ----------":
    def crawl_whole_info_table():
        if "1、网上抓取----------":
            fund_list = []
            try:
                session = requests.session()
                session.headers["Accept"] = \
"text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8"
                session.headers["Accept-Encoding"] = "gzip, deflate, br"
                session.headers["Accept-Language"] = "zh-CN,zh;q=0.9"
                session.headers["Connection"] = "keep-alive"
                session.headers["Upgrade-Insecure-Requests"] = "1"
                session.headers["User-Agent"] = \
                    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
```

```
Gecko) Chrome/63.0.3239.132 Safari/537.36"
                    session.headers["Host"] = "fund.eastmoney.com"
                    session.headers["Referer"] =
'http://fund.eastmoney.com/data/fundranking.html'
                    for fund_type in ['hh', 'gp', 'zq', 'zs']:
                        # 上面地址中'ft=zq'代表爬取债券型基金，可根据情况更改为 pg、gp、hh、
zs、QDII、LOF。(偏股型、股票型、混合型、指数型、QDII 型、LOF 型)
                        url =
"http://fund.eastmoney.com/data/rankhandler.aspx?op=ph&ft=%s&rs=&gs=0&qdii=&tabSubt
ype=,,,,,&pn=10000" % (fund_type)
                        response = session.get(url, verify=False, timeout=60)
                        response.encode = 'utf8'
                        response = response.text
                        response = response[response.find('var rankData = {datas:['):
response.find(']')]
                        response = response.replace("var rankData = {datas:[", "")
                        for i in response.split('"'):
                            if len(i.split(',')) < 10:
                                continue
                            order = i.split(',')
                            for j in range(4, 16):
                                if "." in order[j]:
                                    order[j] = float(order[j])
                                else:
                                    order[j] = 0
                            # 代码位数补齐至六位
                            if len(order[0]) < 6:
                                order[0] = "0" * (6 - len(order[0])) + order[0]
                            fund_list.append(order)
                except Exception as ex:
                    logging.exception(str(ex))
            if "2、创建 pandas 数据表----------------":
                title = ["代码", "名称", "英文", "日期", "单位净值", "累积净值", "日增长", "近 1 周", "近 1
月", "近 3 月", "近 6 月", "近 1 年", "近 2 年", "近 3 年",
                         "今年来", "成立来", "成立时间", "未知", "成立来 2", "折前手续费", "手续费", "折
数", "手续费 2", "折数 2", "未知 2"]
                fund_df = pd.DataFrame(fund_list, columns=title)
                # 只取到成立时间（含）以前的属性
                fund_df = fund_df.iloc[:, :17]
```

```
            # 按照代码升序调整
            fund_df = fund_df.sort_values(by='代码', axis=0,
ascending=True).reset_index(drop=True)
            # 保留第一次出现的那条数据
            fund_df = fund_df.drop_duplicates(keep='first')
        if "3、保存到文件 ":
            file = u'data/wholeInfo111.xlsx'
            fund_df.to_excel(file, index=False, encoding='gbk')
            return fund_df
    if 1:
        df = crawl_whole_info_table()
        print(df)
```

最后一次爬取总简表的时间为 **2022 年 2 月 16 日**，代码运行结果如下。

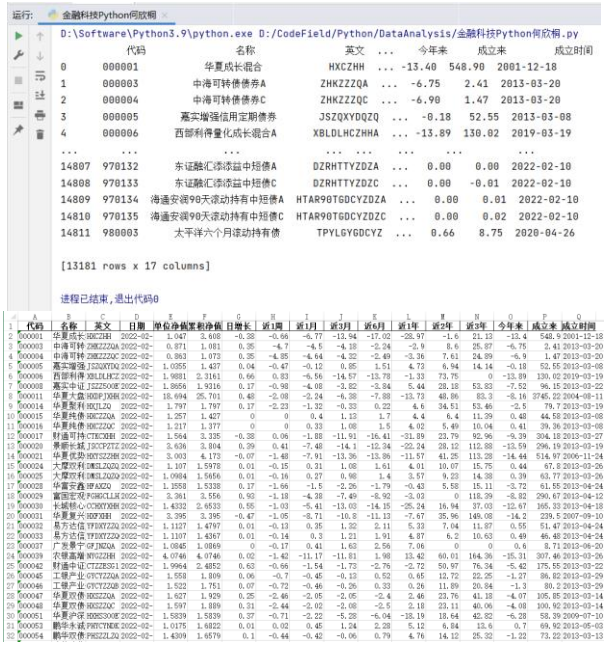

图 1 总简表的运行与导出结果

查看 excel 文件，共有 **13181** 条数据，可以对获得的数据进行验证：如果不考虑重复元素，直接导出文件，使用 excel 进行重复项删除操作，根据如下提示，可知在代码中进行的去重复操作是正确合理的。

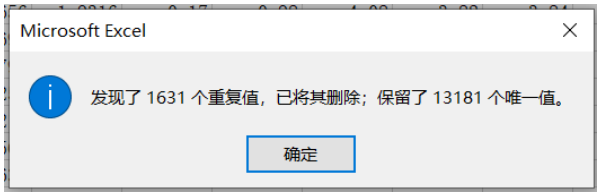

图 2 Excel 处理方法与结果验证

1.2 crawl_one_fund_price(code, per=49, sdate='', edate='')

该函数根据基金代码、每页获取交易日数量、开始日期、结束日期爬取该基金设定时间内的所有交易日数据，返回 dataframe 表格。步骤主要包括：获取总页数和表头、抓取 records 存入 data 表格，并保存为 **data_new/{code}_new.xlsx** 表格文件。

在爬取的时候需要注意：**per 最大为 49**，也就是一页最多爬 49 个记录，如果设置为 50，只会显示 20 条，从而影响爬虫。利用该函数保存总简表的源代码如下。

```
if "抓取一个基金的历史价格 --------------------------------":
    def crawl_one_fund_price(code, per=49, sdate='', edate=''):
        if "1、获取总页数和表头 ---------------":
            url = 'http://fund.eastmoney.com/f10/F10DataApi.aspx'
            params = {'type': 'lsjz', 'code': code, 'page': 1, 'per': per, 'sdate': sdate, 'edate': edate}
            rsp = requests.get(url, params)
            rsp.raise_for_status()
            html = rsp.text

            # 使用正则表达式读取总页数 pages
            pattern = re.compile(r'pages:(.*),')
            pages = int(re.search(pattern, html).group(1))
            heads = []
            soup = BeautifulSoup(html, 'html.parser')
            for head in soup.findAll("th"):
                heads.append(head.contents[0])
        if "2、开始抓取 ------------------":
            records = []
            page = 1
            while page <= pages:
                params = {'type': 'lsjz', 'code': code, 'page': page, 'per': per, 'sdate': sdate, 'edate':
edate}
                rsp = requests.get(url, params=params)
                rsp.raise_for_status()
                html = rsp.text
                soup = BeautifulSoup(html, 'html.parser')
                for row in soup.findAll("tbody")[0].findAll("tr"):
                    row_records = []
                    for record in row.findAll('td'):
```

```
                            val = record.contents
                            # 处理空值
                            if not val:
                                row_records.append(np.nan)
                            else:
                                row_records.append(val[0])
                        records.append(row_records)
                    page = page + 1
                if len(records) == 0:
                    records = [[np.nan, np.nan, np.nan, np.nan, np.nan, "---", "---", np.nan]]
            if "3、得到数据表 ----------":
                np_records = np.array(records)
                data = pd.DataFrame()
                for col, col_name in enumerate(heads):
                    data[col_name] = np_records[:, col]
                data['单位净值'] = data['单位净值'].astype(float)
                data['累计净值'] = data['累计净值'].astype(float)
                data['日增长率'] = data['日增长率'].str.strip('%').astype(float)
                data = data.sort_values(by='净值日期', axis=0,
ascending=True).reset_index(drop=True)
            if "4、保存到文件 ":
                file = u'data/new_data/{}.xlsx'.format(code)
                data.to_excel(file, index=False, encoding='gbk')
    if 1:
        file = u"data/wholeInfo.xlsx"
        df = pd.read_excel(file, usecols=['代码'], dtype={'代码': str})
        total = len(df)
        count = 0
        start = datetime.datetime.now()
        wrong_list = []

        # 读取各个基金信息，同时对每条信息计算所消耗的时间，进行格式化输出
        for i in range(100):
            count += 1
            if not os.path.exists("data/new_data/" + df.iloc[i, 0] + '.xlsx'):
                this_start = datetime.datetime.now()
                try:
                    crawl_one_fund_price(df.iloc[i, 0])
                    this_end = datetime.datetime.now()
```

```
                print("Done [{}]. ({:5} / {:5}) items finished.   ({} / {}) items used time.".
                      format(df.iloc[i, 0], count, total, this_end - this_start, this_end - start))
            # 防止因为出错而中断了程序
            except requests.exceptions.Timeout or requests.exceptions.ConnectionError:
                print("wrong [{}]. test again later.".format(df.iloc[i, 0]))
                wrong_list.append(df.iloc[i, 0])
                sleep(10)
                continue
        else:
            print("exists [{}]. ({:5} / {:5}) items finished.".format(df.iloc[i - 1, 0], count, total))
    print(wrong_list, "should crawl again.")
```

由于只使用一个程序爬虫的速度过慢，查询速度很缓慢，还容易时不时被网站检测到异常，经常主动关闭连接，因此只能实现一部分内容的爬取，爬取界面如下所示。为保证后面题目的数据和老师所给数据集一致，后面仍然使用老师所给 fund_data 的截止到 **2021 年 11 月 26 日**的数据集，这里只是进行所给代码的一个复现，所爬取的 100 个截止到 **2022 年 2 月 16 日**基金的数据放置于 data_new 文件夹下。

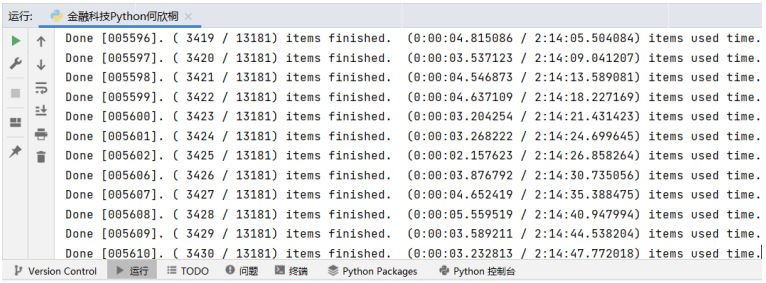

图 3 爬取各个基金交易日数据的程序截图

1.3 treeview_dataframe_general(df, table_info="")

输入 dataframe 表格和表格的标题(可选,默认没有)可以利用窗口查看 dataframe 数据，因为如果直接 print(df)的话不能显示出所有项，也不够美观，故设立该函数。

```
if "数据表的窗口可视化------------------------":
    def treeview_dataframe_general(df, table_info=""):
        if "窗口基本属性 -------------":
            min_num = 29 if len(df) > 29 else len(df)
            win_width = len(df.columns) * 110 + 100
            win_width = 1300 if win_width > 1300 else win_width
            win_high = len(df) * 20 + 100
```

```
            win_high = 600 if win_high > 600 else win_high
            win = tk.Tk()
            win.geometry(str(win_width + 100) + "x" + str(win_high + 100))
            win.title(table_info)
            win.resizable(width=True, height=True)
            tk.Label(win, text=table_info + "  共" + str(len(df)) + "行", font=('微软雅黑', 14),
width=60,
                     height=2).pack()
        if "创建表格窗体 ----------":
            tree = ttk.Treeview(win, height=min_num, show="tree headings")# "增加滚动条":
            vsb = ttk.Scrollbar(win, orient="vertical", command=tree.yview)
            vsb.pack(side='right', fill='y')
            tree.configure(yscrollcommand=vsb.set)
        if "表格行列添加 ----------":
            # 列设置
            tree["columns"] = tuple(["index"] + list(df.columns))
            tree.column("#0", width=0, anchor="center")
            tree.column("index", width=50, anchor="center")
            column_width = int((win_width - 50) / len(df.columns))
            for col in df.columns:  # 增加列
                if col == '代码':
                    tree.column(col, width=50, anchor="center")
                else:
                    tree.column(col, width=column_width, anchor="center")
                tree.heading(col, text=col, anchor="center")
            # 行设置
            for x in range(len(df)):  # 增加行
                item = [list(df.index)[x]]
                for col in df.columns:
                    item.append(df.iloc[x][col])
                tree.insert("", x, text=str(x + 1), values=item)
            tree.pack()
            win.mainloop()
```

1.4 read_filenames_from_folder(folder)

输入文件夹名称 folder，walk 方法可以递归地读取该文件夹下文件，返回文件名列表 files，对文件名进行判断，其子目录和孙子目录等等内部所有名字长度为 6 的 xlsx 后缀的 excel 文件都将被判断为基金文件，最后将返回一个基金代码列表。

```
if "读取 data 文件夹基金价格文件----------":
    def read_filenames_from_folder(folder):
        filename_list = []
        for root, dirs, files in os.walk(folder):
            for f in files:
                portion = os.path.splitext(f)
                if portion[1] == ".xlsx" and len(portion[0]) == 6:
                    filename_list.append(portion[0])
        return filename_list
```

1.5 figure_fund_price_history(codes, start_day=, end_day=):

输入一个基金代码列表和以及起止日期，可绘制这些基金在该时间段内的价格走势。

第一部分是读取数据。首先建立 codes_dict 字典，codes_dict[code] = df 是指，键是代码，值是该代码在起止日期内的数据。对 codes 中的每一个 code 进行遍历，先读取一个代码的 excel 文件为 dataframe 表格，再计算起止日期，如果表格中的第一天比请求的开始日期要晚，那么开始日期应该变更为表格中的第一天，结束日期的处理同理，iloc 可以定位到表格的每个单元格。df[np.array(df['净值日期'] >= start 是取表格中所有净值日期比开始日期大的项目，df[np.array(df['净值日期'] <= end 是取表格中所有净值日期比结束日期小的项目，取一个与操作就是要同时满足两种情况，返回真值，df = (df[np.array(df['净值日期'] >= start) & np.array(df['净值日期'] <= end)]).copy()则是将 df 保存为当前基金在该时间段内的价格。

第二部分是进行绘图。random.shuffle(line_style)的作用是对于一系列的 line_style 进行随机打乱，对于后面每一个代码赋予一种线的形态，plt.plot_date 函数是专门制作时序图的一种方法。最后将 price_trend.jpg 保存在 src 文件夹下。

```
if "基金历史价格可视化-----------------------":
    def figure_fund_price_history(codes, start_day="1000-01-01", end_day="3000-01-01"):
        if "读取基金价格--------------------------------":
            codes_dict = {}
            for code in codes:
                start = start_day
                end = end_day
```

```
                df = pd.read_excel(u"data/" + code + '.xlsx')
                df = df.reindex(columns=["净值日期", "单位净值", "日增长率", "累计净值"])
                if df.iloc[0, 0] > start_day:
                    start = df.iloc[0, 0]
                if df.iloc[len(df) - 1, 0] < end_day:
                    end = df.iloc[len(df) - 1, 0]
                df = (df[np.array(df['净值日期'] >= start) & np.array(df['净值日期'] <=
end)]).copy()
                df['净值日期'] = pd.to_datetime(df['净值日期'], format='%Y-%m-%d')
                codes_dict[code] = df
        if "画图----------------------------------------":
            plt.figure(figsize=(16, 8), dpi=150)
            plt.rcParams['font.family'] = 'Microsoft YaHei'
            plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
            plt.title('基金：' + ' '.join(codes) + '  价格历史行情')
            plt.xlabel('时间')
            plt.ylabel('价格')
            plt.grid(True)
            color = ["m", "g", "b", "c", "y", "r"]
            style = ["-", "--", ":", "-."]
            line_style = []
            for s in style:
                for c in color:
                    line_style.append(c + s)
            random.shuffle(line_style)  # 随机打乱
            i = 0
            for code in codes:
                plt.plot_date(codes_dict[code]["净值日期"], codes_dict[code]["单位净值"],
line_style[i], label=code)
                i = i + 1
            plt.legend(numpoints=1, fontsize=14)
            plt.legend(loc='upper left')
            plt.savefig('./src/price_trend.jpg')
            plt.show()
```

1.6 fund_rise_days_num(codes, start_day=" ", end_day=" ")

输入代码列表和起止日期可计算该基金在起止日期内价格涨跌日占总时间比率。

首先是读取文件，接着是确定有效的起止日期，这和上一个函数的处理基本一致。

df_rise = df[df["日增长率"] > 0]是为了读取符合日增长率大于 0 的那些项目，all_days_num = df.shape[0]取 df 的行数，作为起止日期内的总交易日日期数量，rise_days_num = df_rise.shape[0]取 df_rise 的行数，作为上涨日期数量，rise_ratio = round(rise_days_num / all_days_num, 4)则是使用上涨日期数量除以总日期数量得到上涨日期比率，这里 round 方法取浮点数小数前四位。ratio 列表的保存形式是[[代码，上涨比率，下跌比率][...]...[...]]。接着把 ratio 转换为 df_ratio 表格并返回。

同时要注意有考虑**数据的有效性**，如 **000076 就没有有效信息**。处理方法有以下两周。一种是 if isinstance(df.iloc[0, 0], float)，如果有数据，应该是 str 类型的日期，无效的时候是 nan 类型的 float，应该不考虑直接跳过。一种是当 all_days_num 为 0，说明日期无效，应该跳过这个代码，不考虑。

```
if "基金价格涨跌日比率计算----------":
    def fund_rise_days_num(codes, start_day="1000-01-01", end_day="3000-01-01"):
        ratio = []
        count = 0
        total = len(codes)
        for code in codes:
            start = start_day
            end = end_day
            count += 1
            file = u"data/" + code + '.xlsx'   # 由 crawl_fund_price.py 抓取
            if not os.path.exists(file):
                return "not exist"
            df = pd.read_excel(file, usecols=["净值日期", "单位净值", "日增长率"])
            if isinstance(df.iloc[0, 0], float):
                print('\n', code, ' has a wrong start day, so ignore.')
                continue
            if df.iloc[0, 0] > start_day:
                start = df.iloc[0, 0]
            if df.iloc[len(df) - 1, 0] < end_day:
                end = df.iloc[len(df) - 1, 0]
            df = (df[np.array(df['净值日期'] >= start) & np.array(df['净值日期'] <= end)]).copy()
            df['净值日期'] = pd.to_datetime(df['净值日期'], format='%Y-%m-%d')
            df = df[df["单位净值"] != ""]   # 删除没有值的行
```

```
            all_days_num = df.shape[0]
            if all_days_num == 0:
                print('\n', code, ' has no any valid day data, so ignore.')
                continue
            df_rise = df[df["日增长率"] > 0]
            rise_days_num = df_rise.shape[0]
            rise_ratio = round(rise_days_num / all_days_num, 4)
            df_down = df[df["日增长率"] < 0]
            down_days_num = df_down.shape[0]
            down_ratio = round(down_days_num / all_days_num, 4)
            ratio.append([code, rise_ratio, down_ratio])
            print("\rDone [{}]. ({:5} / {:5}) items finished.".format(code, count, total), end='')
        print("")
        np_ratio = np.array(ratio)
        df_ratio = pd.DataFrame()
        df_ratio['code'] = np_ratio[:, 0]
        df_ratio['rise'] = np_ratio[:, 1]
        df_ratio['down'] = np_ratio[:, 2]
        df_ratio['rise'] = df_ratio['rise'].astype(float)
        df_ratio['down'] = df_ratio['down'].astype(float)
        return df_ratio
```

净值日期	单位净值	累计净值	日增长率	申购状态	赎回状态	分红送配
	0	0	0	0	0	0
nan	0	0	0	0	0	0

图 4 000076 的无效交易信息

1.7 recent_price(code)

输入代码可查看一个月以来的价格情况。

```
def recent_price(code):
    file = u"data/" + code + '.xlsx'
    if not os.path.exists(file):
        return "not exist"
    df = pd.read_excel(file)
    df = df[df["单位净值"] != ""]  # 删除没有值的行
    df = df[df["净值日期"] > "2021-01-16"]  # 删除没有值的行
    return df
```

2　A 类题目

2.1 题目 1

要求：输出基金名字中包含有“混合”两字的基金名称及其代码（从总简表中）。

解答：首先已经通过 crawl_whole_info_table 函数获取了总简表 wholeInfo.xlsx，使用 pandas 的 read_excel 方法读取该文件中的代码和名称两个类型的数据，df['名称']是为了找到表格中的“名称“属性这一列数据，df['名称'].str.contains('混合')返回“名称”这一列中的值转换为字符串后，字符串是否包含“混合“两个字的真值列表，名称包含就是真，不包含就是假，df[df['名称'].str.contains('混合')]则是只返回那些条件为真的项，也就是基金名称中含有“混合”的那些基金，最后使用 treeview_dataframe_general 函数对这个筛选出来的 df 表格进行可视化展现即可。

```
if "作业 A.1--------------------------------":
    def homework_A_1():
        file = u"data/wholeInfo.xlsx"
        if not os.path.exists(file):
            return "not exist"
        df = pd.read_excel(file, usecols=['代码', '名称'], dtype={'代码': str})
        return df[df['名称'].str.contains('混合')]
    if 1:
        homework_a1 = homework_A_1()
        treeview_dataframe_general(homework_a1, "作业 A.1")
```

结果截图如下。

图 5 作业 A1 截图

2.2 题目 2

要求：输出所有基金名称的云图（从总简表中）。

解答：首先 imread 方法读取需要稍后使用的图片背景，预留做准备。再读取总简表 wholeInfo，使用 read_excel 方法读取基金名称转换为 df 表格，使用 values.tolist、join 方法把列表转换为字符串。设置 wordcloud 函数的属性，蒙版 mask 设置为自己设置的 background，字体也设置成自己的 font。最后绘制、保存、展示生成的云图。

```
if "作业 A.2--------------------------------":
    def homework_A_2():
        background = plt.imread('src/beijing.jpeg')
        file = u"data/wholeInfo.xlsx"
        if not os.path.exists(file):
            return "not exist"
        df = pd.read_excel(file, usecols=['名称'])
        text = ','.join([x[0] for x in df.values.tolist()])
        wordcloud = WordCloud(
            background_color="white",
            max_words=1000,
            mask=background,
            font_path='src/font.TTF')
        wordcloud.generate(text)
        wordcloud.to_file("src/wordcloud.png")
        plt.figure()
        plt.axis('off')
        plt.imshow(wordcloud)
        plt.show()
    if 1:
        homework_A_2()
```

使用的图片蒙版和生成的基金名称云图对比如下。

图 6 作业 A2 原图与结果

2.3 题目 3

要求：输出跌天数比率最大的 10 个基金的代码（从价格文件中）。

解答：首先 read_filenames_from_folder 读取出 data 文件夹下所有的代码文件，使用 fund_rise_days_num 可以获得这些 selected_codes 的涨跌比率表格，sort_values 方法对涨跌比率表格排序，按照 down 的大小进行降序排序（这里 down 原本设计的时候是取的日期比率，所以是百分比是正数），使用 df.values.tolist()[:10]转换为列表，输出前 10 个基金，也就是跌天数比率最大的 10 个基金，最后进行格式化输出展现结果。

```
if "作业 A.3------------------------------":
    def homework_A_3():
        selected_codes = read_filenames_from_folder("data")
        df = fund_rise_days_num(selected_codes)
        df.sort_values(by=['down'], ascending=False, inplace=True)
        return df.values.tolist()[:10]

    if 1:
        homework_a3 = homework_A_3()
        count = 1
        for i in homework_a3:
            print("[{:2}] code: {}, down_ratio = {:.2%}".format(count, i[0], i[2]))
            count += 1
```

运行结果如下所示。

```
Done [000075]. (   54 / 12746) items finished.
 000076  has a wrong start day, so ignore.
Done [870009]. (12565 / 12746) items finished.
 870012  has a wrong start day, so ignore.
Done [980003]. (12746 / 12746) items finished.
[ 1] code: 012623, down_ratio = 96.55%
[ 2] code: 009746, down_ratio = 71.48%
[ 3] code: 873002, down_ratio = 71.43%
[ 4] code: 009745, down_ratio = 71.13%
[ 5] code: 014064, down_ratio = 66.67%
[ 6] code: 014028, down_ratio = 63.64%
[ 7] code: 012144, down_ratio = 63.41%
[ 8] code: 872024, down_ratio = 62.86%
[ 9] code: 872023, down_ratio = 62.86%
[10] code: 872022, down_ratio = 62.86%

进程已结束,退出代码0
```

图 7 作业 A3 结果截图

2.4 题目 4

要求：画出多个基金的价格变化曲线（从价格文件中）。

解答：首先 read_filenames_from_folder 读取出 data 文件夹下所有的代码文件，使用 random.sample 任取十个基金进行绘制，使用 figure_fund_price_history 可以获得这些 selected_codes 在 2021 年 1 月 1 日到 2022 年 1 月 1 日的价格曲线。

```
if "作业 A.4--------------------------------":
    def homework_A_4():
        all_codes = read_filenames_from_folder("data")
        selected_codes = random.sample(all_codes, 10)
        figure_fund_price_history(selected_codes, start_day="2021-01-01", end_day="2022-01-
01")
    if 1:
        homework_A_4()
```

结果如下。

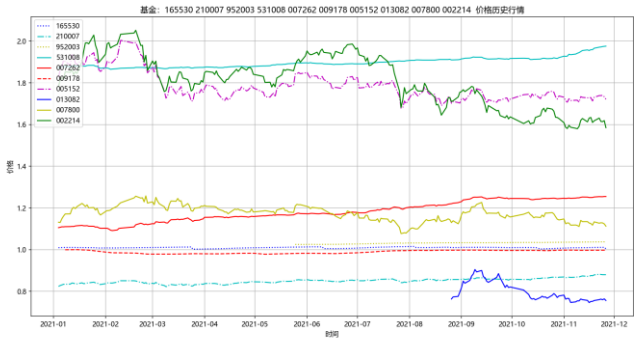

图 8 作业 A4 输出图片截图

2.5 题目 5

要求：输出近 1 个月来跌幅最大的 10 个基金（从总简表中）。

解答：首先读取总简表 wholeInfo，使用 read_excel 方法读取基金名称、近一个月涨跌数据两个列转换为 df 表格，使用 df.sort_values 方法进行排序，条件是对近一个月这个属性进行升序排序，也就是增长由低到高（这里原来所给数据就是有正负的，负的越多就下跌越多），使用 values.tolist 方法将表格转换为列表，取前十个基金的数据表格，然后进行格式化输出。

```
if "作业 A.5------------------------------":
    def homework_A_5():
        file = u"data/wholeInfo.xlsx"
        if not os.path.exists(file):
            return "not exist"
        df = pd.read_excel(file, usecols=['代码', '近 1 月'], dtype={'代码': str, '近 1 月': float})
        df.sort_values(by=['近 1 月'], ascending=True, inplace=True)
        return df.values.tolist()[:10]

    if 1:
        homework_a5 = homework_A_5()
        count = 1
        for i in homework_a5:
            print("[{:2}] code: {}, down_recent_1_month = {:.2f}%".format(count, i[0], i[1]))
            count += 1
```

结果如下。

```
[ 1] code: 004638, down_recent_1_month = -78.47%
[ 2] code: 011815, down_recent_1_month = -18.53%
[ 3] code: 013028, down_recent_1_month = -18.48%
[ 4] code: 001563, down_recent_1_month = -17.08%
[ 5] code: 003956, down_recent_1_month = -15.91%
[ 6] code: 501006, down_recent_1_month = -15.83%
[ 7] code: 501005, down_recent_1_month = -15.80%
[ 8] code: 006080, down_recent_1_month = -15.71%
[ 9] code: 006081, down_recent_1_month = -15.66%
[10] code: 010926, down_recent_1_month = -15.39%

进程已结束,退出代码0
```

图 9 作业 A5 结果截图

2.6 题目 6

要求：指定一些基金，输出近一个月来涨的天数比率最大的 10 个基金的代码。

解答：首先读取出 data 文件夹下所有的代码文件，可以取前 50 个进行下面的操作，由于数据集时间较早，手动设置一个月的数据（注释代码，直接使用 datatime 库读取前 30 天的日期，作为起止日期）。使用 fund_rise_days_num 计算这些代码在起止日期内的涨跌天数比率，保存到 df_ratio 表格。是对 rise 进行降序排序。使用 df_ratio.values.tolist()[:10]转换为列表，取前 10 个基金。最后进行格式化输出如下。

```
if "作业 A.6------------------------------":
    def homework_A_6():
        # yesterday = (datetime.date.today() + datetime.timedelta(-1)).strftime("%Y-%m-%d")
        # one_month_before = (datetime.date.today() + datetime.timedelta(-
31)).strftime("%Y-%m-%d")
        codes = read_filenames_from_folder("data")[:50]
        yesterday = "2021-11-26"
        one_month_before = "2021-10-26"
        df_ratio = fund_rise_days_num(codes, start_day=one_month_before,
end_day=yesterday)
        df_ratio.sort_values(by=['rise'], ascending=False, inplace=True)
        return df_ratio.values.tolist()[:10]
    if 1:
        homework_a6 = homework_A_6()
        count = 1
        for i in homework_a6:
            print("[{:2}] code: {}, rise_rate_recent_1_month = {:.2%}".format(count, i[0], i[1]))
            count += 1
```

```
Done [000071]. (   50 /    50) items finished.
[ 1] code: 000053, rise_rate_recent_1_month = 100.00%
[ 2] code: 000037, rise_rate_recent_1_month = 91.67%
[ 3] code: 000033, rise_rate_recent_1_month = 91.67%
[ 4] code: 000032, rise_rate_recent_1_month = 91.67%
[ 5] code: 000024, rise_rate_recent_1_month = 87.50%
[ 6] code: 000025, rise_rate_recent_1_month = 83.33%
[ 7] code: 000055, rise_rate_recent_1_month = 69.57%
[ 8] code: 000043, rise_rate_recent_1_month = 69.57%
[ 9] code: 000005, rise_rate_recent_1_month = 66.67%
[10] code: 000041, rise_rate_recent_1_month = 65.22%

进程已结束,退出代码0
```

图 10 作业 A6 结果截图

2.7 题目 7

要求：输出近 1 周和近 1 个月跌幅都排名在前 20 的基金（从总简表中）。

解答：首先读取总简表，选取代码、近一月、近一周属性读入 df 表格，df1 存储以近一月属性升序排序的前 20 个基金，df2 存储以近一周属性升序排序的前 20 个基金，pd.merge 函数将 df1 和 df2 进行合并（how 参数默认为'inner'也就是交集），如果合并后的表格长度为 0，表示没有交集，就没有排名都在前 20 的基金，否则直接输出。

```
if "作业 A.7--------------------------------":
    def homework_A_7():
        file = u"data/wholeInfo.xlsx"
        if not os.path.exists(file):
            return "not exist"
        df = pd.read_excel(file, usecols=['代码', '近 1 月', '近 1 周'], dtype={'代码': str, '近 1 月':
float, '近 1 周': float})
        df1 = df.sort_values(by=['近 1 月'], ascending=True)[:20]
        df2 = df.sort_values(by=['近 1 周'], ascending=True)[:20]
        return pd.merge(df1, df2)

    if 1:
        homework_a7 = homework_A_7()
        if len(homework_a7) == 0:
            print("没有近 1 周来和近 1 个月来跌幅都排名在前 20 的基金！")
        else:
            print(homework_a7)
```

前 20 和前 100 的运行结果如下所示。

```
没有近 1 周来和近 1 个月来跌幅都排名在前 20 的基金!

进程已结束,退出代码0
```

图 11 作业 A7 结果截图（前 20）

```
     代码   近1周   近1月
0  011900 -6.06 -12.97
1  011899 -6.05 -12.93
2  168601 -9.20 -12.46
3  006924 -5.23 -12.31
4  006923 -5.23 -12.29

进程已结束,退出代码0
```

图 12 作业 A7 结果截图（前 100）

3 B 类题目

3.1 题目 8

要求：编写函数 rising_days_distribution(code)，可以统计某一个基金 code 的连续涨跌天数，并以直方图的形式画出其连续涨跌天数分布直方图（从价格文件中）。

解答：首先要读取该代码的价格文件，将单位净值属性导入为 df 表格，这里只需要考虑净值的变化即可。

下面介绍如何**计算连续涨跌天数。**首先建立一个 rising_days_list 列表存储连续天数。i 作为循环变量，每次取一个交易日，如果第 i 天的净值大于第 i-1 天，连续计数加一，天数加一，当不大于的时候跳出循环，如果连续计数非零就把计数加入 rising_days_list；如果第 i 天的净值小于第 i-1 天，连续计数加一，天数加一，当不小于的时候跳出循环，如果连续计数非零就把计数的负值（把下跌连续时间当作负数处理）加入 rising_days_list。如果循环到最后，rising_days_list 依旧为空，说明数据有问题。

关于**直方图坐标如何对齐**的方法参考了这篇回答。这种方法首先要确定连续天数最大值和最小值，确定好两边的端点，bins 数量、text 位置、plt.xticks、plt.xlim，均要对应才行，否则就有偏差，直方图就不美观。在这张图中表示的连续天数，负数为连续下跌，整数为连续上涨。

```
if "作业 B.8------------------------------":
    def rising_days_distribution(code):
        file = u"data/" + code + '.xlsx'
        if not os.path.exists(file):
            return "not exist"

        df = pd.read_excel(file, usecols=['单位净值'])

        i = 1
        count = 0
        rising_days_list = []
        while i < len(df):
```

```
            while i < len(df) and df.iloc[i, 0] > df.iloc[i - 1, 0]:
                count += 1
                i += 1
            if count > 0:
                rising_days_list.append(count)
                count = 0

            while i < len(df) and df.iloc[i, 0] < df.iloc[i - 1, 0]:
                count += 1
                i += 1
            if count > 0:
                rising_days_list.append(-count)
                count = 0

            while i < len(df) and df.iloc[i, 0] == df.iloc[i - 1, 0]:
                i += 1

        if len(rising_days_list) == 0:
            print(code, "has no any data please try again.")
            return

        max_num = max(rising_days_list)
        min_num = min(rising_days_list)

        plt.figure(figsize=(10, 5), dpi=150)
        plt.rcParams['font.family'] = 'Microsoft YaHei'
        plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']

        bins = np.arange(min_num, max_num + 2) - 0.5
        nums, bins, patches = plt.hist(rising_days_list, color='steelblue', edgecolor='black',
bins=bins)
        for bin, num in zip(bins, nums):
            plt.text(bin + 0.5, num, int(num), ha='center', va='bottom', fontsize=10)

        plt.title(code + ' 的连续涨跌天数')
        plt.xticks(range(min_num, max_num + 1))
        plt.xlim([min_num - 1, max_num + 1])
        plt.xlabel('连续天数（负数为连续下跌，整数为连续上涨）/天')
        plt.ylabel('频数/次')
```

```
    plt.savefig('./src/rise_down_trend.jpg')
    plt.show()


if 1:
    code = random.choice(read_filenames_from_folder("data"))
    rising_days_distribution(code)
```

输出结果如下。

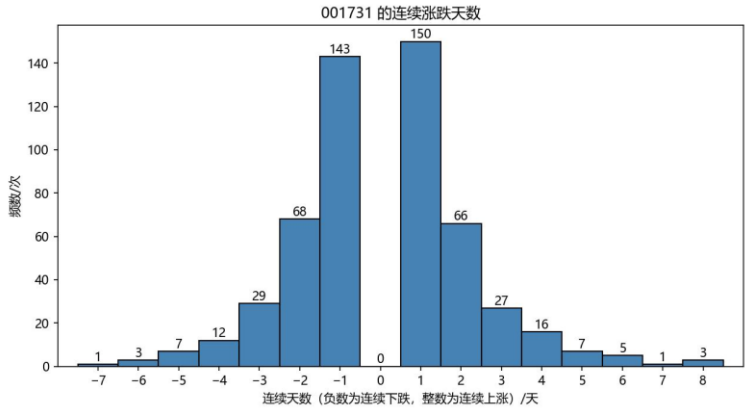

图 13 作业 B8 输出图片结果

3.2 题目 10

要求：编写函数 gain_loss_max(codes, money, from_date, end_date)，计算如果分别都投入资金数额为 money 的资金给代码为 codes 的多个基金（codes 为基金代码序列），投资时间范围是从 from_date 到 end_date，计算投资哪支基金的收益将会最大，输出该基金代码、名称，及相应的收益额（从价格文件中）。

解答：本题较为复杂，且融合了一部分题目 9 的内容，为方便编程与理解，该问题分解了为如下几个函数。

第一个是 get_code_name(code)，其作用是**已知代码求对应名称**，基金代码在运算中需求比较多，只需要对代码传参即可，而且基金代码和基金名称是一对一关系，对基金名称传参太冗余，故建立此函数。

第二个是 get_redemption_rate_and_price(code, from_date=' ', end_date=' '): 该函数求解**一个基金在起止日期内的赎回费率、并返回赎回费率、起止日期的累计净值。**众所周知，基金的赎回费率和持有日期相关，所以要首先确定一下两者的关系，把赎回费率确定下来，这里使用 datetime 库进行计算日期间隔。使用累计净值而不用单位净值是为了更符合基金的收益计算。

第三个是 gain_loss(code, money= , from_date=' ', end_date=' ')，这个函数其实就是问题 9，也就是**计算一个基金在起止日期内的收益。**这里使用的公式参考了[天弘沪深 300 招募说明书]中《(七）申购份额与赎回金额的计算》的计算公式。

具体计算方法如下：

申购费用 = 申购金额 - 申购金额÷（1 + 申购费率)

申购份额 = 净申购金额÷T 日基金份额净值

赎回总额 = 赎回份额× T 日基金份额净值

赎回净额 = 赎回总额×（1-赎回费率 ）

净利润 = 赎回净额 - 申购费用

第四个是 gain_loss_max(codes, money, from_date, end_date)，也就是本程序需要解决的问题，**在多个基金中输出盈利最多的那一个**，首先建立了 gain_loss_dict 字典，键是基金代码，值是利用 gain_loss 函数计算的该基金在起止日期内的盈利。最后输出该字典降序排列后的第一个基金和它的盈利。

最后是该问题的主函数，read_filenames_from_folder 读取所有基金代码，random.sample 在所有代码中**任意选取多个**进行下面的计算，使用上面已经编写好的 gain_loss_max 函数获得代码和盈利，再用 get_code_name 获得对应的名字即可，最后对计算结果进行格式化输出。

```
if "作业 B.10--------------------------------":
    def get_code_name(code):
        file = u"data/wholeInfo.xlsx"
        if not os.path.exists(file):
            return "not exist"
        df = pd.read_excel(file, usecols=['代码', '名称'], dtype={'代码': str})
        return df[df['代码'] == code].iloc[0, 1]

    def get_redemption_rate_and_price(code, from_date='1900-01-01', end_date='2100-01-01'):
        file = u"data/" + code + '.xlsx'
        if not os.path.exists(file):
            return "not exist"

        if from_date > end_date:
            from_date, end_date = end_date, from_date

        year, month, day = [int(x) for x in from_date.split('-')]
        time_1_struct = datetime.date(year, month, day)
        year, month, day = [int(x) for x in end_date.split('-')]
        time_2_struct = datetime.date(year, month, day)
        delta = (time_2_struct - time_1_struct).days

        if delta < 7:
            redemption_rate = 0.0150
        elif delta < 30:
            redemption_rate = 0.0075
```

```
        elif delta < 365:
            redemption_rate = 0.0050
        elif delta < 365 * 2:
            redemption_rate = 0.0025
        else:
            redemption_rate = 0

        df = pd.read_excel(file, usecols=["净值日期", "累计净值"])
        if df.iloc[0, 0] > from_date:
            from_date = df.iloc[0, 0]
        if df.iloc[len(df) - 1, 0] < end_date:
            end_date = df.iloc[len(df) - 1, 0]
        df = (df[np.array(df['净值日期'] >= from_date) & np.array(df['净值日期'] <=
end_date)]).copy()

        from_price = df.iloc[0, 1]
        end_price = df.iloc[-1, 1]

        return redemption_rate, from_price, end_price


    def gain_loss(code, money=10000, from_date='1900-01-01', end_date='2100-01-01'):
                subscription_rate = 0.015
        redemption_rate, from_price, end_price = get_redemption_rate_and_price(code,
from_date, end_date)

        net_subscription_money = money / (1 + subscription_rate)
        subscription_portion = net_subscription_money / from_price

        redemption_money = subscription_portion * end_price
        net_redemption_money = redemption_money * (1 - redemption_rate)
        return net_redemption_money - money


    def gain_loss_max(codes, money, from_date, end_date):
        gain_loss_dict = {}
        for code in codes:
            result = gain_loss(code, money, from_date, end_date)
            gain_loss_dict[code] = result
```

```
        return sorted(gain_loss_dict.items(), key=lambda x: x[1], reverse=True)[0]


    if 1:
        all_codes = read_filenames_from_folder("data")
        selected_codes = random.sample(all_codes, 10)
        print("you select codes are:")
        count = 0
        for code in selected_codes:
            count += 1
            print("[{:2}] [{}] {}".format(count, code, get_code_name(code)))
        money = 100000
        from_date = "2021-01-01"
        end_date = "2022-01-01"
        max_code, max_money = gain_loss_max(selected_codes, money, from_date, end_date)
        max_name = get_code_name(max_code)
        print("max is {}, code = {}, you can earn ￥{:.2f}.".format(max_name, max_code,
max_money))
```

运行结果如下。

```
you select codes are:
[ 1] [009863] 富国创新趋势股票
[ 2] [900155] 中信证券债券增强六个月持有C
[ 3] [011069] 工银成长精选混合A
[ 4] [007177] 浙商智能行业优选混合A
[ 5] [013720] 新华增怡债券E
[ 6] [003324] 东方永兴18个月定开债A
[ 7] [006360] 财通鸿益中短债债券A
[ 8] [013356] 大摩沪港深精选混合A
[ 9] [011549] 九泰久慧混合C
[10] [003106] 光大永鑫混合C
max is 浙商智能行业优选混合A, code = 007177, you can earn ￥9565.84.

进程已结束,退出代码0
```

图 14 作业 B10 结果截图

4　C 类题目

4.1 题目 11

要求：抓取主题基金：编写程序抓取各个主题或行业的基金代码。

解答：本题目参考了 b 站尚硅谷爬虫课程，使用 **selenium 库**对网站进行爬取，使用 selenium 的原因是对天天基金网站 requests.get()和网页源代码不一致，这是因为网站上面的内容不是直接 html 之类写的，而是使用了 js 等技术进行了加载，所以看源代码根本找不到其中的内容，翻页的话网站的网址也没有更新，而是在同一个页面中进行，在同一个页面中有多个页签，而不是多个页面由浏览器新建了多个页签，这样的动态变化比较麻烦，而且反爬能力要比别的库稍微强一点。

根据观察，**网站具体架构**如下所述。首先是一页行业和概念的网页，这个页面包含所有的行业和概念的文字及其对应的代码。点击文字后，在同一页面生成一个新页签，该页签被选中，进入当前行业或概念，在下方有当前行业或概念的翻页按钮。部分需要的点击或者获取的信息所在位置如下。

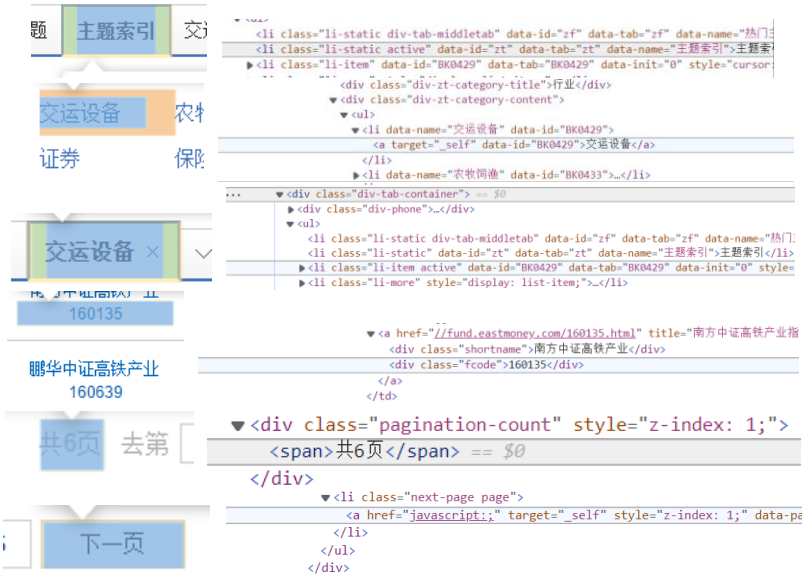

图 15 检索主题基金爬虫问题中需要的关键元素的位置

该问题主要分为五个部分。

第一部分，**配置好 edge 浏览器驱动**，这一部分参照教程完成，代码比较固定，配

置好后可进行自动化测试。

第二部分，**如何进入每个行业或者领域。**要先确定好每个概念或领域的位置，使用 css 选择器遍历不同概念领域，使用 find_elements(By.CSS_SELECTOR, "li[data-type='hy'] a[data-id^='BK']")找到行业元素列表，同理，使用 type='gn'找到概念元素列表。利用 get_attribute('data-id')获取行业的代码，然后使用 click()进入新行业或概念，作为参数传给 get_codes，接着进行第三部分。

第三部分，**如何获取一个行业或者领域的每页信息。**从上面已经得到了行业或领域对应的 BK 代码，使用 css 选择器"div[class='tab-page'][data-tab='" + theme_id + "'] div[data-tab='zt'] .pagination-count"找到当前行业或概念的总页数，使用.fcode 找到当前页面所有基金的六位数代码，加入临时存储列表 codes，翻页需要找到翻页按钮，找到匹配".next-page.page"的对象 click 即可，如此循环即可获得某一行业或领域的所有代码，装在 **codes 列表**中。作为 **get_codes 的返回值。**以行业为例子，主函数的字典 field_dict[field.text] = get_codes(driver, field_id)就是以行业名称为键名，值就是该行业所有基金代码的一个列表。

第四部分，**如何返回主题总页面。**使用 css 选择器找到"li[data-id='zt']"对象，然后 click 即可，后面继续进行进入每个行业和概念的循环，知道行业或者概念循环完毕。

第五部分，**把结果输出到文件。**前面保存了行业和领域的字典，使用循环 write 到文件即可，行业的键值对输出到 fields.txt，概念的键值对输出到 concepts.txt。

```
if "作业 C.11--------------------------------":
    def get_codes(driver, theme_id):
        codes = []
        query_str = "div[class='tab-page'][data-tab='" + theme_id + "'] div[data-
tab='zt'] .pagination-count"
        driver.implicitly_wait(10)
        page_num = driver.find_element(By.CSS_SELECTOR, query_str).text[1:-1]
        # 对当前主题每页进行提取
        for i in range(int(page_num)):
            print('\r[{}] finished {:2}/{:2} page(s).'.format(theme_id, i + 1, page_num), end='')
```

```
            # 获得当前主题的页面所有的 code 加入 codes
            driver.implicitly_wait(10)
            for code in driver.find_elements(By.CSS_SELECTOR, ".fcode"):
                codes.append(code.text)
            # 当前页面代码都保存进 codes 列表，现在跳转到下一页
            driver.implicitly_wait(10)
            driver.find_element(By.CSS_SELECTOR, ".next-page.page").click()
        # 所有代码都保存进 codes 列表，现在返回主题页面
        driver.implicitly_wait(10)
        driver.find_element(By.CSS_SELECTOR, "li[data-id='zt']").click()
        sleep(0.8)
        return codes

    def share_browser():
        field_dict = {}
        concept_dict = {}
        driver = webdriver.Edge('src/msedgedriver.exe')
        driver.implicitly_wait(10)

driver.get(r'http://fund.eastmoney.com/ztjj/#!curr/zt-%E4%B8%BB%E9%A2%98%E7%B4%A2%E
5%BC%95/fs/SON_1N/fst/desc')
        driver.implicitly_wait(10)
        fields = driver.find_elements(By.CSS_SELECTOR, "li[data-type='hy'] a[data-id^='BK']")
        count = 0
        for field in fields:
            count += 1
            field_id = field.get_attribute('data-id')
            field_name = field.text
            field.click()
            sleep(0.8)
            field_dict[field.text] = get_codes(driver, field_id)
            print(" [{:^8}] finished {:2}/{:2}.".format(field_name, count, len(fields)))

        f = open('src/fields.txt', 'w')
        for k, v in field_dict.items():
            print(k, v)
            f.write(k + ': ' + ' '.join(v)+ '\n')
        f.close()
```

```
        driver.implicitly_wait(10)
        concepts = driver.find_elements(By.CSS_SELECTOR, "li[data-type='gn'] a[data-id^='BK']")
        count = 0
        for concept in concepts:
            count += 1
            concept_id = concept.get_attribute('data-id')
            concept_name = concept.text
            concept.click()
            sleep(0.8)
            concept_dict[concept.text] = get_codes(driver, concept_id)
            print(" [{:^8}] finished {:2}/{:2}.".format(concept_name, count, len(fields)))

        f = open('src/concepts.txt', 'w')
        for k, v in concept_dict.items():
            print(k, v)
            f.write(k + ': ' + ' '.join(v) + '\n')
        f.close()
        driver.quit()

    if 1:
        share_browser()
```

运行截图和爬虫结果如下所示。

```
[BK0429] finished  1/1  page(s). [  交运设备  ] finished  1/27.
[BK0433] finished  1/1  page(s). [  农牧饲渔  ] finished  2/27.
[BK0438] finished 16/16 page(s). [  食品饮料  ] finished  3/27.
[BK0447] finished 12/12 page(s). [ 互联网服务  ] finished  4/27.
[BK0448] finished  1/1  page(s). [  通信设备  ] finished  5/27.
[BK0451] finished  3/3  page(s). [ 房地产开发  ] finished  6/27.
[BK0454] finished  1/1  page(s). [  塑料制品  ] finished  7/27.
[BK0456] finished  1/1  page(s). [  家电行业  ] finished  8/27.
[BK0459] finished  2/2  page(s). [  电子元件  ] finished  9/27.
[BK0465] finished  7/17 page(s).
```

图 16 作业 C11 运行截图

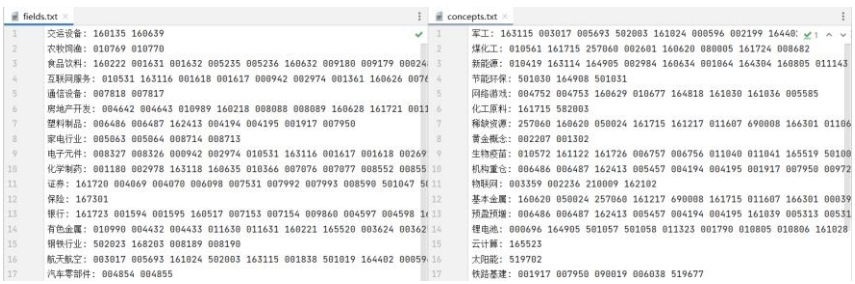

图 17 作业 C11 结果