项目链接：[Lancercxy/automatic_transaction: 分析上证指数，并建立买卖规则，实现自动化交易 (github.com)](https://github.com/Lancercxy/automatic_transaction)

准备工作：

1、导入相关模块

```
import akshare as ak
import matplotlib.pyplot as plt
from pylab import mpl
import pandas as pd
import requests
from bs4 import BeautifulSoup
import re
from jqdatasdk import *
import time
import datetime
```

2、使用之前的项目收集上证 A 股的所有个股信息（这里为了实验只获取部分数据）

项目链接：[Lancercxy/Finance_crawler: 财经个股爬虫 (github.com)](https://github.com/Lancercxy/Finance_crawler)

效果图：

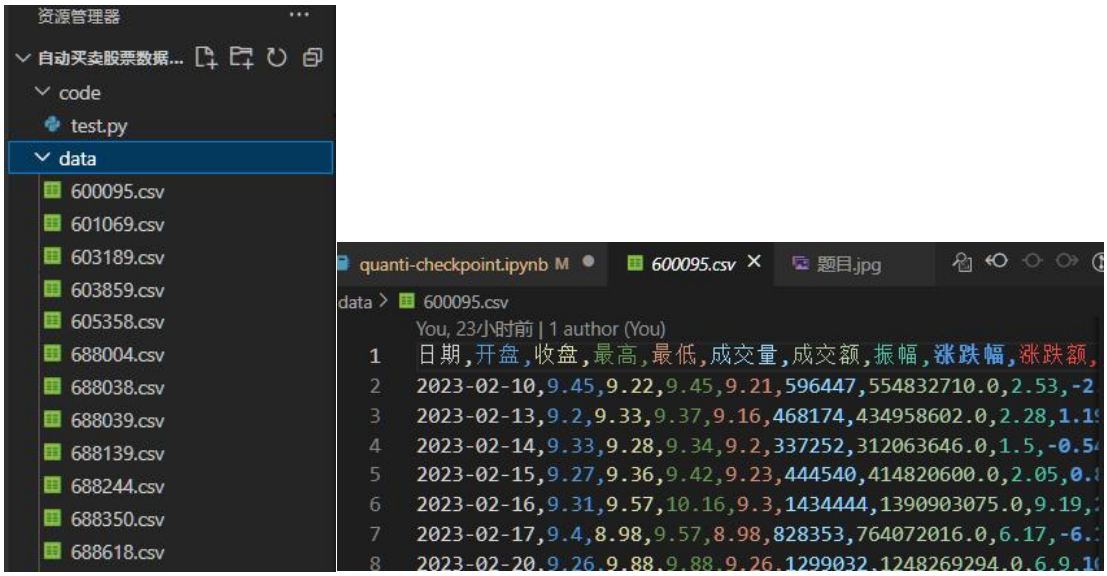
	代码	名称	最新价	涨跌幅	涨跌额	成交量	成交额	振幅	最高	最低	今开	昨收	量比	换手率	市盈率(动态)	市净率
0	"603190"	N亚通	41.89	44.0	12.8	25672	107368398.0	23.99	41.89	34.91	34.91	29.09	"-"	8.56	33.34	2.63
1	"832149"	N利尔达	6.98	39.6	1.98	290851	208265047.53	37.0	8.35	6.5	6.5	5.0	36.73	26.3	23.37	3.61
2	"300314"	戴维医疗	17.5	20.03	2.92	188008	315384096.74	16.53	17.5	15.09	15.09	14.58	6.46	9.93	49.74	4.86
3	"688466"	金科环境	22.74	20.0	3.79	78421	171758457.0	18.52	22.74	19.23	19.23	18.95	6.19	17.0	28.7	2.16
4	"301211"	亨迪药业	33.69	16.78	4.84	347868	1088681681.43	21.73	34.33	28.06	28.94	28.85	5.83	59.75	77.2	3.63
5	"301301"	川宁生物	10.36	14.6	1.32	927424	913687341.06	15.15	10.39	9.02	9.02	9.04	3.93	44.33	51.98	3.76
6	"300922"	天秦装备	17.05	13.67	2.05	170364	283421959.6	20.47	17.99	14.92	15.03	15.0	4.45	19.87	85.89	3.1
7	"832735"	德源药业	32.7	13.27	3.83	20257	63480773.27	18.01	33.6	28.4	28.46	28.87	1.97	4.51	22.8	2.79
8	"300039"	上海凯宝	9.03	13.02	1.04	2032532	1749389537.32	19.02	9.43	7.91	7.99	7.99	2.3	22.18	59.72	2.56

3、编写 get_stock()函数该函数接收个股代码并通过 akshare 模块获取历史行情

```
def get_stock(code):

    #获取当前日期与 30 天的历史行情
    start_date = (datetime.datetime.now() - datetime.timedelta(days = 30))
    start_date = start_date.strftime('%Y%m%d')
    end_date = time.strftime('%Y%m%d')

    df = ak.stock_zh_a_hist(symbol=code,start_date=start_date, end_date=end_date,
adjust="")

    #个股历史行情存储路径，以股票代码命名
    path = f'D:\office\Github\爬虫\自动买卖股票数据采集\data\{code}.csv'
```

```
    #以 CSV 格式写入信息
    df.to_csv(path,index=False)
```

效果图：



4、编写 get_sz()函数该函数使用网络爬虫获取上证指数的历史行情

```
def get_sz():

    #1、请求链接
    url =
'https://q.stock.sohu.com/hisHq?code=zs_000001&stat=1&order=D&period=d&callback
=historySearchHandler&rt=jsonp&0.4530586399394587'

    #2、请求头
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
Edg/109.0.1518.70',
               'Cookie':'gidinf=x099980109ee1586bb342f0390007640b6db6df607be;
reqtype=pc;
BIZ_MyLBS=cn_688039%2C%u5F53%u8679%u79D1%u6280%7Ccn_000002%2C%u4E07%u79D1%uFF21
%7Ccn_601121%2C%u5B9D%u5730%u77FF%u4E1A; t=1678499703304'
               }

    #3、发送 get 请求
    response = requests.get(url, headers=headers)

    #4、获取响应中的字符
    sz = response.text
```

```
    #5、通过三步清洗获取 T-1 日涨跌幅
    sz01 = sz.split('[')[4].split(']')[0]
    sz02 = sz01.split(',')[4]
    sz03 = sz02.replace('%','').replace('"','')

    #6、判断涨跌幅是否大于等于 0.5,满足条件返回 True 否则返回 False
    if sz03 >= 0.5:
        return True
    else:
        return False
```

上证指数历史行情：


```
historySearchHandler([{"status":0,"hq":[["2023-03-10","3255.51","3230.08","-46.02","-1.40%","3229.50
09","3285.94","3276.09","-7.15","-0.22%","3260.00","3289.06","264021856","32882570.00","-"],["2023-0
07","3320.21","3285.10","-36.93","-1.11%","3284.41","3342.86","389957952","43008116.00","-"],["2023-
03","3314.77","3328.39","17.74","0.54%","3302.62","3330.60","343996896","38886000.00","-"],["2023-03
01","3279.14","3312.35","32.74","1.00%","3272.04","3315.16","318398720","38438740.00","-"],["2023-02
27","3257.00","3258.03","-9.13","-0.28%","3251.72","3276.58","242855344","32460508.00","-"],["2023-0
23","3293.52","3287.48","-3.67","-0.11%","3275.36","3307.44","262324304","33436454.00","-"],["2023-0
21","3291.63","3306.52","16.19","0.49%","3282.44","3308.79","324409440","39069124.00","-"],["2023-02
17","3244.73","3224.02","-25.01","-0.77%","3223.26","3262.47","278451232","35562584.00","-"],["2023-
15","3294.01","3280.49","-12.79","-0.39%","3274.55","3296.20","269939104","35762184.00","-"],["2023-
13","3256.99","3284.16","23.49","0.72%","3252.63","3285.09","297213184","39701140.00","-"],["2023-02
09","3227.73","3270.38","38.28","1.18%","3225.77","3270.38","254754528","35176084.00","-"],["2023-02
07","3245.23","3248.09","9.40","0.29%","3233.83","3250.03","250946464","32125764.00","-"],["2023-02-
03","3275.66","3263.41","-22.26","-0.68%","3235.35","3275.66","293342816","37864852.00","-"],["2023-
01","3262.20","3284.92","29.25","0.90%","3245.41","3284.92","339509408","41339520.00","-"],["2023-01
30","3308.87","3269.32","4.50","0.14%","3266.76","3310.49","353325824","45317700.00","-"],["2023-01-
19","3221.52","3240.28","15.87","0.49%","3210.38","3240.28","226465040","29681556.00","-"],["2023-01
17","3229.44","3224.24","-3.35","-0.10%","3211.76","3231.26","226592240","30261180.00","-"],["2023-0
```

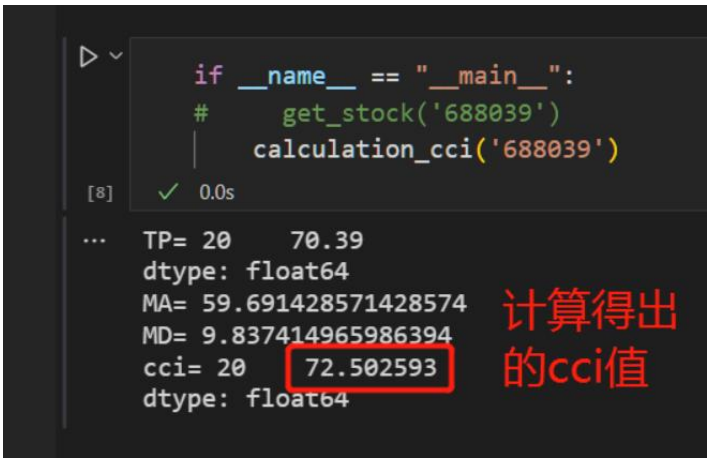

5、编写 calculation_cci()函数该函数接收个股代码列表，设计计算 CCI 值算法筛选满足条件的股票，然后返回

```
def calculation_cci(stock_list):

    #1、创建 buy_code_list 列表用于存储满足条件的个股代码
    buy_code_list = []

    #2、遍历代码
    for code in stock_list:

        #3、根据传入的股票代码读取数据
        df = pd.read_csv(f'D:\office\Github\爬虫\自动买卖股票数据采集
\data\{code}.csv',sep=',')

        #4、判断数据是否有 14 条以上，获取 T-1 日涨跌幅
        if len(df) > 14:
            df_Rise_fall = df[-2:].head(1)['涨跌幅']
            df_Rise_fall = float(df_Rise_fall)
```

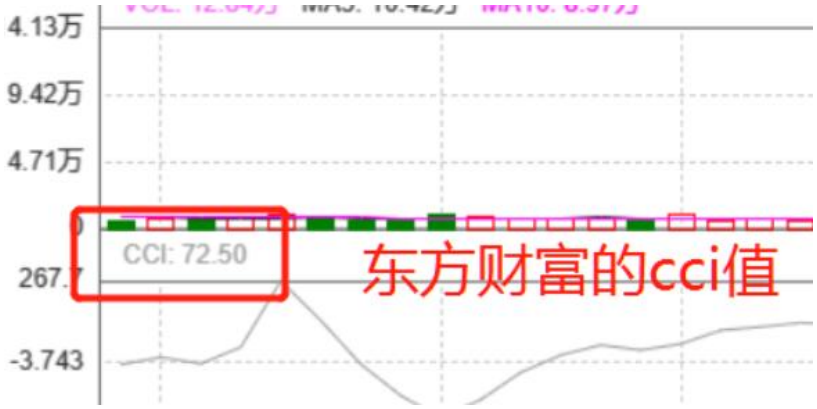
```
            #5、创建 cci_list 列表用于存储 T-1 与 T-2 的 cci 值
            cci_list = []

            for j in range(1,3):
                #6、删去 T 日与 T-1 的数据然后计算 cci
                df.drop(df.tail(1).index,inplace=True)

                #7、获取最后一行数据（当天数据）计算 TP 值
                TP = (df[-1:]['最高']+df[-1:]['最低']+df[-1:]['收盘'])/3
                # print("T-{0}的 TP={1}".format(j,TP))

                #8、获取进 14 日的数据，并计算 MA 值
                df_14 = df.tail(14)
                MA = (df_14[['最高','最低','收盘']].sum(axis=1).sum(axis=0))/42
                # print("T-{0}的 MA=".format(j,MA))

                #9、计算 MD 值
                md = df_14[['最高','最低','收盘']].sum(axis=1)

                #10、参数 add 用于存储 14 天 TP 之和
                add = 0
                for i in md:
                    add += abs(i/3 - MA)
                MD = add/14
                # print("T-{0}的 MD=".format(j,MD))

                #11、计算 cci 数据
                cci = (TP - MA)/MD/0.015
                # print("T-{0}的 cci 值为:{1:.2f}".format(j,float(cci)))

                #12、将 cci 值添加入列表
                cci_list.append(float(cci))
                # print(cci_list)

            #13、根据要求进行判断
            # if cci_list[1] >= -50 and cci_list[1] <= 100 and cci_list[0] >= 100
and cci_list[0] <= 200 and df_Rise_fall >= 10:
            #为了测试降低了买入要求
            if cci_list[1] >= -50 and cci_list[1] <= 100 :
                print('可买股票',code)

                #14、将符合要求的个股添加入 buy_code_list 列表
                buy_code_list.append(code)
                # return code
```

```
            else:
                print('不可买',code)

    #15、返回符合要求的代码列表
    return buy_code_list
```

计算得到的 CCI 值：



实际 CCI 值：



6、编写 high_open()函数该函数用于接收个股代码进行高开判断

```
def high_open(code):
    #1、获取今天日期
    g.today = context.current_dt.strftime('%Y-%m-%d')

    #2、获取上个交易日日期
    g.start = context.current_dt - datetime.timedelta(day = 1)

    #3、以上个交易日为开始时间，今日为结束时间获取个股信息
    grid = get_price(f'{code}.XSHE', start_date=g.start, end_date=g.today,
fields=['open', 'high', 'low', 'close', 'high_limit', 'paused'])

    #4、判断是否满足高开在 0-2%之间
```

```
    if len(grid)>1 and grid.open[-1] > grid.close[-2] and grid.open[-1] /
grid.close[-2] >= 1.00 and grid.open[-1] / grid.close[-2] <= 1.02 :
        return True
    else:
        return False
```

7、编写 change_pct()函数该函数用于接收个股代码进行涨跌幅判断

```
def change_pct(code):
    #1、获取今天日期
    g.today = context.current_dt.strftime('%Y-%m-%d')

    #2、获取上个交易日日期
    g.start = context.current_dt - datetime.timedelta(day = 1)

    #3、获取涨跌幅
    money_flow = get_money_flow(f'{code}.XSHG',start_date=g.start,
end_date=g.today, fields="change_pct")

    #4、判断是否满足高开在 0-2%之间
    if len(money_flow)>1 and money_flow.change_pct[-1] < 0.1 and
money_flow.change_pct[-1] > -0.1 :
        return '0'
    elif money_flow.change_pct[-1] <= -0.1:
        return '-1'
    else:
        return '1'
```

8、编写 timeFun()函数该函用于定时执行某函数

```
def timerFun(sell_Timer,fun):

    #标识参数
    flag = 0
    while True:

        #获取当前时间并与传入时间对比
        now = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        if now == sell_Timer.strftime('%m%d%H%M%S'):

            #若到了规定时间则执行函数
            # fun

            #并将 flag 参数设置为 1
            flag = 1
        else:
```

```
            #若 flag 为 1,意味着已经执行完目标函数，可将时间调整为下一天
            if flag == 1:
                sell_Timer = sell_Timer + datetime.timedelta(day=1)

                #将标识重新标为 0
                flag = 0
```

以上准备工作完毕

****

以下为程序执行顺序

一、开盘前执行 before_market_open()函数该函数用于开盘前对数据进行收集和筛选出符合要求的个股使用 buy_code_list 参数接收返回值

```
#该函数用于开盘前对数据进行收集和筛选出符合要求的个股
def before_market_open():

    #1、读取提前使用网络爬虫收集好的上证 A 股的所有个股信息
    df = pd.read_csv('D:\office\Github\爬虫\财经\东方财富.csv',sep=',')

    #2、创建名为 stock_list 的列表，用于存储个股代码
    stock_list = []

    #3、为了方便测试只拿出 10 支股票进行操作
    for i in df['代码'][30:41]:

        #4、清洗数据中多余的字符，并添加进列表里
        stock_list.append(i.replace('"',''))

    #5、遍历列表
    for j in stock_list:

        #6、将列表中的个股代码逐个传入 get_stock()函数，以获取该股票的历史行情
        get_stock(j)

    #7、判断 T-1 日上证指数涨幅是否大于 0.5%，get_sz()函数用于判断
    # if get_sz():

    #为了方便测试条件设定为 True
    if True:
        #8、满足 T-1 日上证指数涨幅大于 0.5%,使用 calculation_cci()计算 cci 值，进行进一步筛选
        code = calculation_cci(stock_list)
```

```
        #9、判断 code 是否为空，不为空则返回
        if len(code):
            return code
```

二、将列表转化为 DataFrame 类型，再以 CSV 格式保存再本地

```
pf = pd.DataFrame(buy_code_list)
    pf.columns=['code']
    pf.to_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\willbuy_code_list.csv',index=False)
```

满足题目条件的个股代码：

三、开盘时调用 market_open()函数执行买入操作

```
def market_open():
    # log.info('函数运行时间(market_open):'+str(context.current_dt.time()))
    #1、读取将要购入的个股代码
    df = pd.read_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\data\willbuy_code_list.csv')

    #2、创建名为 own_stock 的列表，用于存储以购买的个股代码
    own_stock = []

    #3、取得当前的现金并平均分成 n 份（n 为即将购买的个股数）
    cash = context.portfolio.available_cash/len(df)

    #4、遍历数据
    for code in df['code']:
        # print(code)
        # g.security = f'{code}.XSHE'
        # security = g.security

        #5、调用 high_open()函数判断高开是否在 0-2%之间
        if high_open(f'{code}.XSHE'):
```

```
            #6、若满足要求用 cash 买入股票
            order_value(code, cash)

            #7、将已经买入的股票代码保存入 own_stock 列表
            own_stock.append(code)
        else:
            # 若不满足则打印出来
            print("不买入",code)

    #8、将已经买入的股票代码以 CSV 格式保存入本地文件
    pf = pd.DataFrame(own_stock)
    pf.columns=['code']
    pf.to_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\data\own_stock.csv',index=False)
```

四、14:55 执行卖出操作

```
#获取 T+1 日 14:55 的时间
    tomorrow = (datetime.datetime.now() + datetime.timedelta(days = 1))
    sell_Timer =
datetime.datetime(tomorrow.year,tomorrow.month,tomorrow.day,14,55,0)

    #将规定时间和函数传入 timerFun()函数,该函数能控制在规定时间点运行指定函数
    timerFun(sell_Timer,sell_stock())
```

实现卖出操作函数：

```
#该函数用于卖出符合要求的个股
def sell_stock():

    #1、读取已拥有的个股代码
    df = pd.read_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\data\own_stock.csv')

    #2、创建 Limit_of_drop 列表用于存储跌停的个股
    Limit_of_drop = []

    #3、判断数据是否为空,不为空则遍历数据
    if len(df) != 0:
        for code in df['code']:
            # print(code)
            # g.security = f'{code}.XSHE'
            # security = g.security
```

```
            #4、调用 change_pct()函数判断其涨跌幅(返回'0'可卖出,返回'-1'为跌停,涨停返回'1')
            state = change_pct(f'{code}.XSHE')
            if state == '0':

                #5、若满足要求卖出股票
                order_target(code, 0)

                #6、将已经卖出的股票代码从 own_stock 文件移除
                df = pd.read_csv('D:\office\Github\爬虫\自动买卖股票数据采集\data\own_stock.csv')
                df01 = df.drop(df[df['code']==code].index)
                df01.to_csv('D:\office\Github\爬虫\自动买卖股票数据采集\data\own_stock.csv',index=False)

            #7、处理跌停的个股
            elif state == '-1':

                #8、将代码添加进列表
                Limit_of_drop.append(code)

                #9、将跌停的股票代码从 own_stock 文件移除
                df = pd.read_csv('D:\office\Github\爬虫\自动买卖股票数据采集\data\own_stock.csv')
                df01 = df.drop(df[df['code']==code].index)
                df01.to_csv('D:\office\Github\爬虫\自动买卖股票数据采集\data\own_stock.csv',index=False)
                # print("无法卖出",code)

        if len(Limit_of_drop) != 0:
            #10、将跌停的股票代码以 CSV 格式保存为 Limit_of_drop_stock 文件
            drop = pd.DataFrame(Limit_of_drop)
            drop.columns=['code']
            drop.to_csv('D:\office\Github\爬虫\自动买卖股票数据采集\data\Limit_of_drop_stock.csv',index=False)
```

五、9:20 执行集合竞价操作

```
#获取 T+2 日 9:20 的时间
    am_tomorrow = (datetime.datetime.now() + datetime.timedelta(days = 2))
    am_sell_Timer = datetime.datetime(am_tomorrow.year,am_tomorrow.month,am_tomorrow.day,9,20,0)

    #在 9:20 执行集合竞价操作
```

```
    timerFun(am_sell_Timer,bidding())
```

实现集合竞价操作：

```
#该函数用于集合竞价操作
def bidding():

    #1、读取跌停的个股代码
    df = pd.read_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\data\Limit_of_drop_stock.csv')

    #2、获取今天日期
    dt=context.current_dt

    #3、得到股票昨日收盘价，每天只需要取一次
    last_df = history(1,'1d','close',code)

    #4、判断数据是否为空,不为空则遍历个股代码
    if len(df) != 0:
        for code in df['code']:

            #5、获取集合竞价买 5 档挂单数据
            d1 = get_call_auction(code, start_date=dt, end_date=dt,fields=['time',
'current', 'b1_p', 'b1_v'])
            money1=float(d1['b1_p']*d1['b1_v'])

            d2 = get_call_auction(code, start_date=dt, end_date=dt,fields=['time',
'current', 'b2_p', 'b2_v'])
            money2=float(d2['b2_p']*d2['b2_v'])

            d3 = get_call_auction(code, start_date=dt, end_date=dt,fields=['time',
'current', 'b3_p', 'b3_v'])
            money3=float(d3['b3_p']*d3['b3_v'])

            d4 = get_call_auction(code, start_date=dt, end_date=dt,fields=['time',
'current', 'b4_p', 'b4_v'])
            money4=float(d4['b4_p']*d4['b4_v'])

            d5 = get_call_auction(code, start_date=dt, end_date=dt,fields=['time',
'current', 'b5_p', 'b5_v'])
            money5=float(d5['b5_p']*d5['b5_v'])

            #6、集合竞价有抢筹动作，有大单挂买档，至少 800 万金额 并且高开不能小于 4%
```

```
            if money1 > g.min_money or money2 > g.min_money or money3>g.min_money
or money4>g.min_money or money5>g.min_money and
d1['current']>(last_df[code][0]*1.04) :

                #7、若满足要求则卖出股票
                order_target(code, 0)

                #8、将已经卖出的股票代码从 Limit_of_drop_stock 文件移除
                df = pd.read_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\data\Limit_of_drop_stock.csv')
                df01 = df.drop(df[df['code']==code].index)
                df01.to_csv('D:\office\Github\爬虫\自动买卖股票数据采集
\data\Limit_of_drop_stock.csv',index=False)
```