Thread 程式作業　　109502561 顏紹同

前言:

這幾個月剛入股市，賠越多錢就越發現自己得到的資訊實在是少得過分，要同時關心技術面和基本面，但是時間太少沒辦法自己慢慢過濾資料，有在關注的幾檔股票又沒什麼起伏，所以如果可以有一個方式使自己能快速獲得資訊那會方便許多，以前就想過試著爬蟲看看，這次剛好有機會就來實做看看。
這次做的是用 multi thread 爬蟲爬股票和公司的歷史紀錄，使用者可以輸入年份和月份，程式會輸出股票的資料，我使用了兩種方式呈現。
第一種是輸入個股編號，用 multithread 爬蟲後輸出這段時間區間的資料
第二種是直接輸出當月每一檔股票該公司的營收資料，用 multithread 爬蟲後，將檔案以試算表方式輸出
想做這兩種是因為我覺得這兩種資料對於股市整體的分析(技術面和基本面)都各有好處，說不定以後我會再加進一些功能充實它

內容:

第一種(getSingleStock_multithreading):
使用者輸入個股編號和日期後，生成相對的網址，再透過這個網址去取得資料。
https://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date={Date}&stockNo={Symbol}
Date 和 Symbol 分別是日期和個股編號，以台積電(2330)2020 年 3 月為例，就是
https://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date=20200301&stockNo=2330
進入網頁之後會看到這個樣子

{"stat":"OK","date":"20200301","title":"109年03月 2330 台積電　　各日成交資訊","fields":["日期","成交股數","成交金額","開盤價","最高價","最低價","收盤價","漲跌價差","成交筆數"],"data":
[["109/03/02","86,372,942","26,864,484,400","308.00","317.00","308.00","311.00","-5.00","35,320"],["109/03/03","55,169,411","17,534,766,696","318.50","320.00","316.00","317.50","+6.50","20,713"],
["109/03/04","44,745,146","14,304,795,666","322.00","322.00","317.00","320.50","+3.00","16,118"],["109/03/05","38,223,525","12,392,618,100","325.00","326.00","323.00","323.00","+2.50","15,107"],
["109/03/06","52,807,760","16,733,485,610","320.00","320.50","315.00","315.00","-8.00","24,175"],["109/03/09","88,633,277","27,236,002,054","307.50","310.00","305.50","305.50","-9.50","44,882"],
["109/03/10","74,869,130","22,727,941,511","301.50","309.00","301.00","307.00","+1.50","30,268"],["109/03/11","64,923,710","19,913,151,529","309.00","310.50","302.00","302.00","-5.00","27,176"],
["109/03/12","114,173,351","33,544,278,206","299.00","299.00","287.00","294.00","-8.00","56,989"],
["109/03/13","151,268,148","42,448,997,546","275.00","294.00","272.50","290.00","-4.00","71,990"],
["109/03/16","103,933,228","29,341,288,916","285.00","291.00","275.50","276.50","-13.50","51,292"],
["109/03/17","122,409,651","32,936,712,540","265.00","276.50","265.00","268.00","-8.50","57,123"],
["109/03/18","118,620,643","31,555,231,792","269.50","272.50","260.00","260.00","-8.00","53,985"],
["109/03/19","160,811,697","39,559,889,553","252.00","253.00","235.50","248.00","X0.00","77,302"],
["109/03/20","159,179,726","42,110,816,042","258.50","270.00","256.00","270.00","+22.00","69,829"],
["109/03/23","76,963,709","19,851,467,203","257.00","262.50","252.00","255.00","-15.00","41,621"],
["109/03/24","82,377,933","22,287,033,614","268.00","274.00","266.00","267.50","+12.50","41,077"],["109/03/25","80,514,199","22,368,633,590","276.50","280.00","274.00","277.00","+9.50","40,932"],
["109/03/26","54,189,633","15,108,299,935","279.50","280.00","275.50","280.00","+3.00","27,648"],["109/03/27","69,320,306","19,324,060,590","284.00","286.00","273.00","273.00","-7.00","36,025"],
["109/03/30","53,403,956","14,205,364,184","263.50","269.00","262.50","267.50","-5.50","27,779"],
["109/03/31","53,901,923","14,712,557,472","273.00","274.00","269.50","274.00","+6.50","20,343"]],"notes":["符號說明:+/-/X表示漲/跌/不比價","當日統計資訊含一般、零股、盤後定價、鉅額交易，不含拍賣、標購。","ETF證券代號第六碼為K、M、S、C者，表示該ETF以外幣交易。"]}

非常適合用來爬蟲呢
Multi thread 的核心程式碼是這幾句:
with concurrent.futures.ThreadPoolExecutor(max_workers=12) as executor:

```
    try:
        executor.map(Get_StockPrice,symbols,dates)
    except:
        print("Something is wrong")
```

max_workers 是代表有幾條 threads
因為我是把要執行的編號和日期都存在陣列(symbols,dates)裡面，所以用 executor.map 使每一條 thread 分配要用編號和日期的陣列第幾個元素來執行 Get_StockPrice()

在 Get_StockPrice()使用 pd.DataFrame 處理資料，網頁資料輸入為 Stock_data，根據每個 column，存入陣列 StockPrice，到此為止就可以做很多事情了!但是我還沒想好分析方式，所以我就直接把它 print 出來，證明確實有爬蟲到。

```
請輸入年份： 2020
請輸入股票編號： 2330
請輸入起始月份： 9
請輸入截止月份： 11
multithreading...
             Open   High    Low  Close     Volume
Date
2020-09-01  430.0  435.0  428.0  435.0  50129.577
2020-09-02  441.0  441.0  430.5  433.0  42013.407
2020-09-03  439.5  439.5  433.5  436.0  40683.617
2020-09-04  427.0  432.5  427.0  429.0  51349.911
2020-09-07  428.0  432.5  425.0  426.0  39854.584
2020-09-08  428.0  433.0  427.5  431.0  23714.670
2020-09-09  425.0  428.0  423.0  427.0  40727.362
2020-09-10  432.5  435.0  430.5  435.0  35281.921
2020-09-11  435.5  436.5  432.5  436.5  34893.469
2020-09-14  436.0  442.0  435.5  441.0  39325.833
2020-09-15  440.5  447.0  439.5  445.0  39661.449
2020-09-16  460.0  462.0  455.5  458.0  70608.238
2020-09-17  453.0  455.0  446.5  448.5  44420.506
2020-09-18  447.0  449.5  443.0  444.0  52999.922
2020-09-21  443.5  450.0  440.0  440.0  40193.920
2020-09-22  440.0  441.0  436.0  437.0  36842.639
2020-09-23  436.0  438.5  432.0  433.5  47426.963
```

```
2020-11-13  459.0  462.0  456.5  462.0   31483.881
2020-11-16  470.0  484.5  469.0  484.0   83380.911
2020-11-17  502.0  506.0  485.5  485.5   75763.870
2020-11-18  490.0  497.0  486.5  497.0   56607.501
2020-11-19  499.0  499.0  490.0  490.0   47345.316
2020-11-20  486.0  490.0  486.0  488.0   35242.579
2020-11-23  494.5  498.5  492.0  496.5   43708.352
2020-11-24  499.5  500.0  491.5  492.0   38630.079
2020-11-25  495.0  495.5  487.0  487.0   47179.640
2020-11-26  489.0  493.5  488.0  489.0   31844.322
2020-11-27  487.5  492.0  486.5  489.0   35196.829
2020-11-30  493.0  493.5  480.5  480.5  149311.778

18.358386516571045 秒爬取
```

最後輸出花多久時間，這就是精隨了，隨後我試用了不用 multithread 的版本，發現在相同資料範圍內，使用 multithread 速度相差非常多。

無 multithread 版本:

```
請輸入年份：  2020
請輸入股票編號：  2330
請輸入起始月份：  9
請輸入截止月份：  11
crawling...
             Open   High    Low  Close     Volume
Date
2020-09-01  430.0  435.0  428.0  435.0  50129.577
2020-09-02  441.0  441.0  430.5  433.0  42013.407
2020-09-03  439.5  439.5  433.5  436.0  40683.617
2020-09-04  427.0  432.5  427.0  429.0  51349.911
2020-09-07  428.0  432.5  425.0  426.0  39854.584
2020-09-08  428.0  433.0  427.5  431.0  23714.670
2020-09-09  425.0  428.0  423.0  427.0  40727.362
2020-09-10  432.5  435.0  430.5  435.0  35281.921
2020-09-11  435.5  436.5  432.5  436.5  34893.469
2020-09-14  436.0  442.0  435.5  441.0  39325.833
2020-09-15  440.5  447.0  439.5  445.0  39661.449
2020-09-16  460.0  462.0  455.5  458.0  70608.238
2020-09-17  453.0  455.0  446.5  448.5  44420.506
2020-09-18  447.0  449.5  443.0  444.0  52999.922
2020-09-21  443.5  450.0  440.0  440.0  40193.920
2020-09-22  440.0  441.0  436.0  437.0  36842.639
2020-09-23  436.0  438.5  432.0  433.5  47426.963
2020-09-24  425.5  429.0  423.0  423.0  82510.381
2020-09-25  427.0  428.0  421.0  424.0  41263.269
2020-09-28  427.0  431.5  424.5  431.5  34427.479
2020-09-29  432.5  435.0  428.0  431.0  42730.502
2020-09-30  430.5  435.0  430.5  433.0  28859.204
2020-10-29  436.5  439.5  435.5  437.0  43075.387
2020-10-30  437.0  437.0  432.0  432.0  49770.771
             Open   High    Low  Close      Volume
Date
2020-11-02  433.0  435.5  428.0  435.5   34539.576
2020-11-03  439.5  443.0  438.0  441.0   31461.727
2020-11-04  444.5  451.5  443.0  450.0   37929.652
2020-11-05  451.5  451.5  445.5  451.0   36539.446
2020-11-06  455.0  455.5  450.0  452.5   32644.190
2020-11-09  458.0  460.0  454.0  458.5   43415.670
2020-11-10  452.0  454.5  448.5  451.0   36786.882
2020-11-11  448.5  457.0  448.5  457.0   40355.804
2020-11-12  463.0  463.5  457.5  458.0   57993.774
2020-11-13  459.0  462.0  456.5  462.0   31483.881
2020-11-16  470.0  484.5  469.0  484.0   83380.911
2020-11-17  502.0  506.0  485.5  485.5   75763.870
2020-11-18  490.0  497.0  486.5  497.0   56607.501
2020-11-19  499.0  499.0  490.0  490.0   47345.316
2020-11-20  486.0  490.0  486.0  488.0   35242.579
2020-11-23  494.5  498.5  492.0  496.5   43708.352
2020-11-24  499.5  500.0  491.5  492.0   38630.079
2020-11-25  495.0  495.5  487.0  487.0   47179.640
2020-11-26  489.0  493.5  488.0  489.0   31844.322
2020-11-27  487.5  492.0  486.5  489.0   35196.829
2020-11-30  493.0  493.5  480.5  480.5  149311.778
53.48954486846924 秒爬取
```

因為只有爬 3 個月份，在有 12 個 threads 的情況下，時間快 3 倍，如果爬一整年的資料，可以快 12 倍!代表有沒有 multithread 對整個程式的執行速度影響非常大。

第二種(getAllStock):

使用者輸入日期後，生成相對的網址，再透過這個網址去取得當月台股營收資料。

這裡我用不同的網址測試~

url = https://mops.twse.com.tw/nas/t21/sii/t21sc03"_{year}_{month}_0.html

year 和 month 分別是民國年和月份，以 109 年 3 月份為例就是

https://mops.twse.com.tw/nas/t21/sii/t21sc03_109_3_0.html

大概長這樣

另存CSV (檔案內容包含國內及國外公司)

上市櫃公司、興櫃公司、及金管會主管之金融業自民國102年起適用IFRSs；非上市上櫃興櫃之公發公司(含金控子公司為公發公司之非金融業、投控子公司為公發公司者)，自民國104年起適用IFRSs，配合IFRSs實施，營業收入係以合併營業收入申報；無子公司者，則以個別營業收入申報；自願提前適用IFRSs者亦同。
上市櫃、(興)櫃公司自96年起電子工業細分為「半導體業、電腦及週邊設備業、光電業、通信網路業、電子零組件業、電子通路業、資訊服務業及其他電子業」等8大類，另獨立化學工業及生技醫療業，為免資訊重複，刪除「電子工業」及「化學生技醫療業」彙總資訊。

出表日期：111/04/28

產業別：水泥工業　　單位：千元

公司代號	公司名稱	營業收入 當月營收	上月營收	去年當月營收	上月比較增減(%)	去年同月增減(%)	累計營業收入 當月累計營收	去年累計營收	前期比較增減(%)	備註
1101	台泥	9,473,250	5,000,692	10,876,929	89.43	-12.90	21,976,083	25,356,331	-13.33	-
1102	亞泥	5,077,457	2,851,105	8,318,969	78.08	-38.96	13,138,881	19,390,605	-32.24	-
1103	嘉泥	179,493	150,813	166,883	19.01	7.55	495,334	452,391	9.49	-
1104	環泥	479,717	414,075	437,140	15.85	9.73	1,253,758	1,187,296	5.59	-
1108	幸福	418,037	390,288	313,093	7.10	33.51	1,104,720	834,347	32.40	-
1109	信大	488,590	199,776	541,582	144.56	-9.78	1,165,081	1,360,434	-14.35	-
1110	東泥	146,412	115,270	147,819	27.01	-0.95	387,944	417,039	-6.97	-
合計		16,262,956	9,122,019	20,802,415	78.28	-21.82	39,521,801	48,998,443	-19.34	

產業別：食品工業　　單位：千元

Multi thread 的核心程式碼是這幾句:

```
with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
    executor.map(monthly_report,urls)
```

和上面第一種的手法十分相似，一樣把要爬蟲的網址存在 urls 陣列，在分配給每一個 threads 去分別以拿到的參數執行 monthly_report()

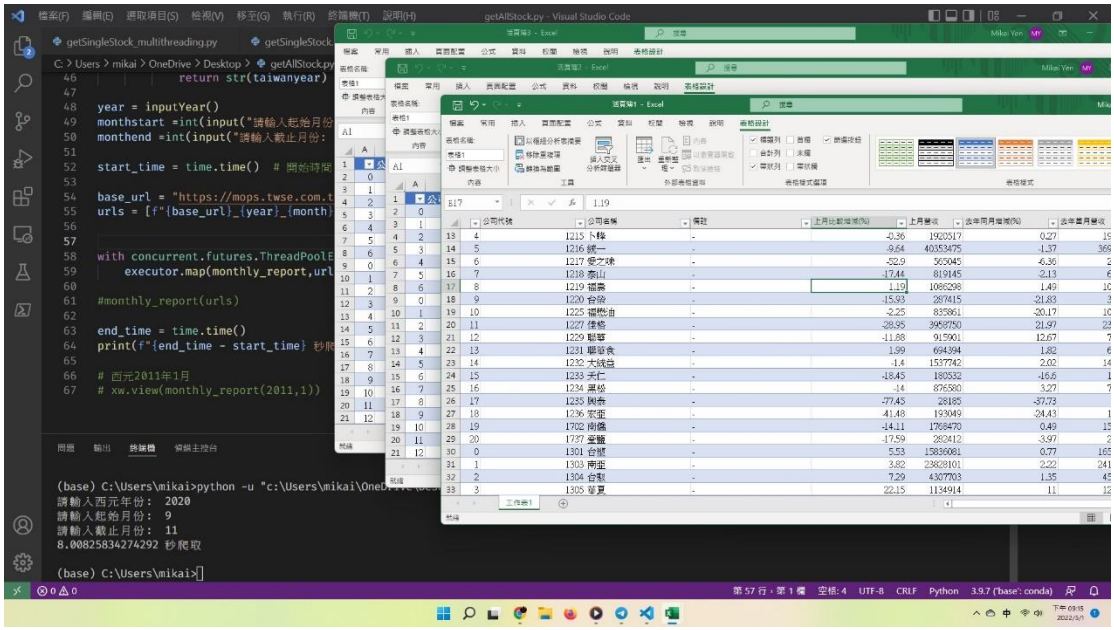
接著是 monthly_report()的部分，因為這個網站和前一個網站差很多，不能直接把文字搬過來使用，所以必須先把網頁文字下載下來:

```
r = requests.get(url, headers=headers)
```

並用 pandas 轉換成 dataframe

後面的部分基本上一樣，最後使用 xw.view()就成功開啟試算表了!這個開啟試算表的功能基本上也可以直接加在第一種的程式碼上，算是做出區隔，而且也不一定要直接放進試算表，所以這麼做是為了讓程式看起來更靈活。

結果就會是這樣



出現三個月份的營收表後，輸出總執行時間

當然這裡我也試過，用 multithread 比不用還要快很多。

總結

爬蟲如果使用 multithread，使多個 thread 同時進行爬蟲，會比一個一個慢慢爬還要快。

使用陣列形式儲存爬蟲資料在資料修改和排序上會比較方便。

如果輸入第一次沒反應，再輸入一次就可以正常運作，不知道是不是我的電腦問題。

參考資料:

Multithread:

https://www.learncodewithmike.com/2020/11/multithreading-with-python-web-scraping.html

爬蟲:

https://www.learncodewithmike.com/2020/11/python-pandas-dataframe-tutorial.html

後記

一開始是使用 multithread 實現 Bogosort，因為這種排序法是先將數列隨機排列，再檢查有沒有排序好，以一般傳統演算法來說這樣根本是在搞笑，但是用 multithread 意外的適合，速度上真的快很多，但是寫完之後莫名來了空虛感，有一種不知道自己在幹嘛的感覺，所以改成做這個。