**組員：趙睿 呂慧瑄**

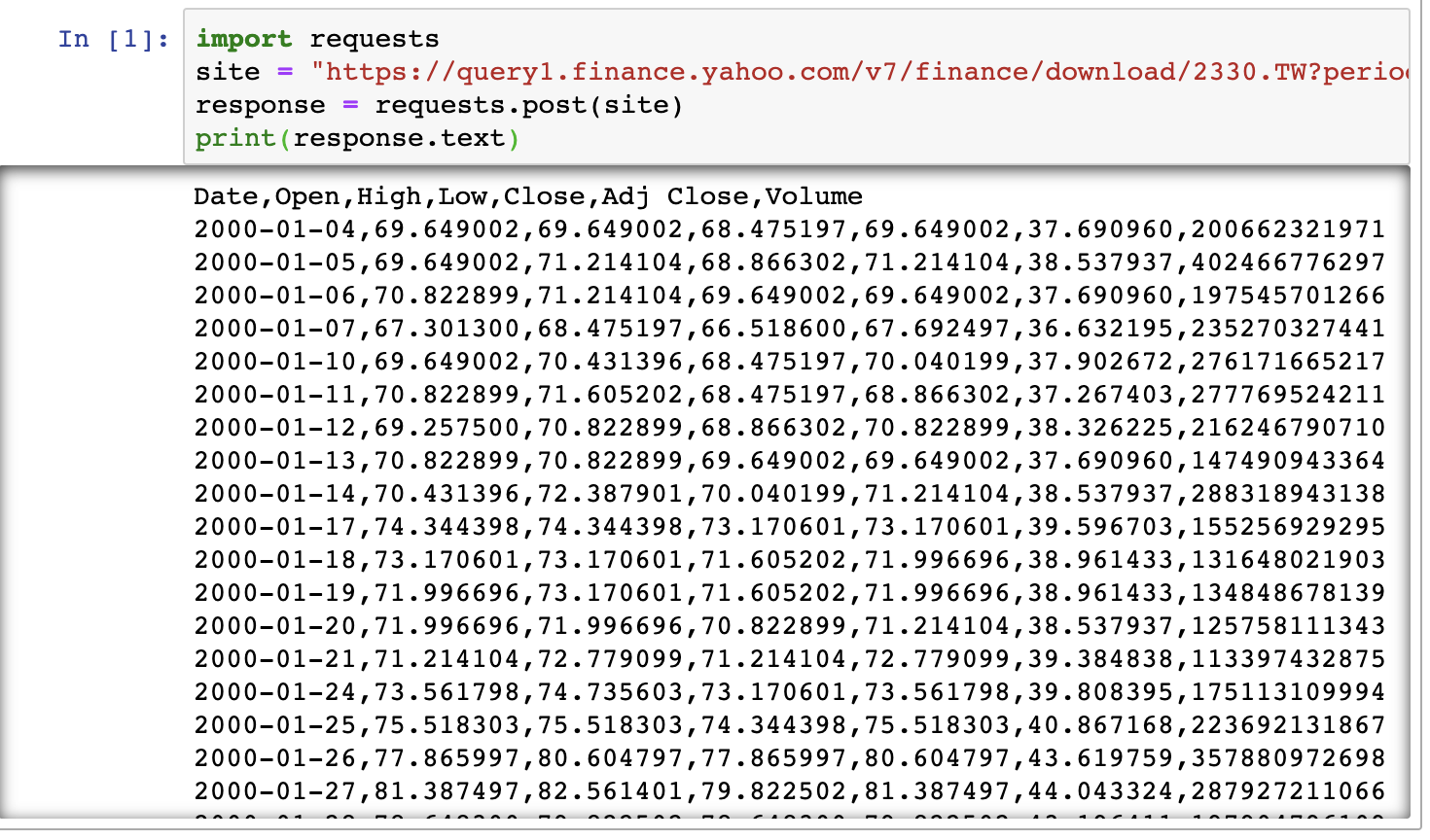
**夏普指數策略**

利用Sharpe ratio來進行台股的模擬買賣，假裝我們這20年來，都使用sharpe ratio的策略，可以得到多少獲利呢？

**爬取歷史股價**

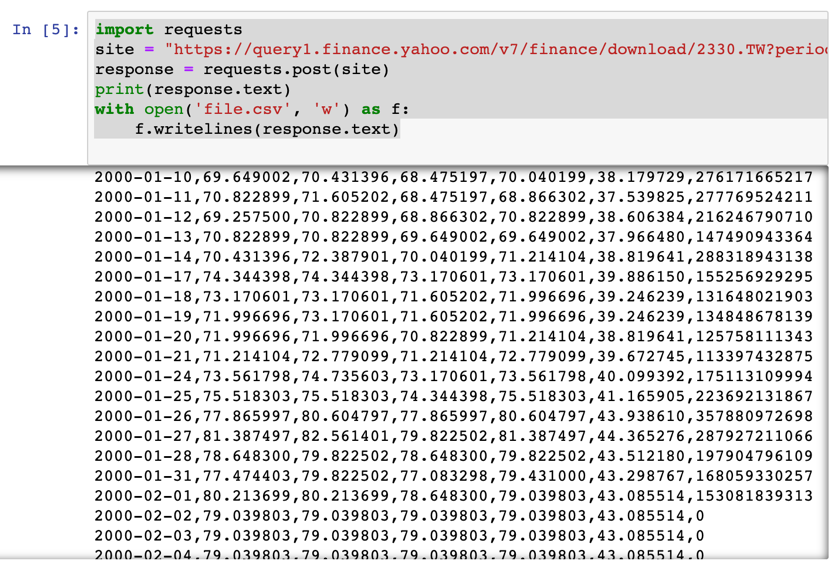
首先要先獲取股價資料，我們會用常用的package：requests  
它可以將網路上的資料給下載下來，給程式使用：

import requests



將資料存檔會發現此文字檔非常大…，所以我們希望能夠先將它存起來  
下方的程式，就是將response.text的文字存入file.csv這個檔案之中，我們會先用with來確保open('file.csv', 'w')有被正確的執行：

with open('file.csv', 'w') as f:  
f.writelines(response.text)



**import requests**

**site = "https://query1.finance.yahoo.com/v7/finance/download/2330.TW?period1=0&period2=1549258857&interval=1d&events=history&crumb=hP2rOschxO0"**

**response = requests.post(site)**

**print(response.text)**

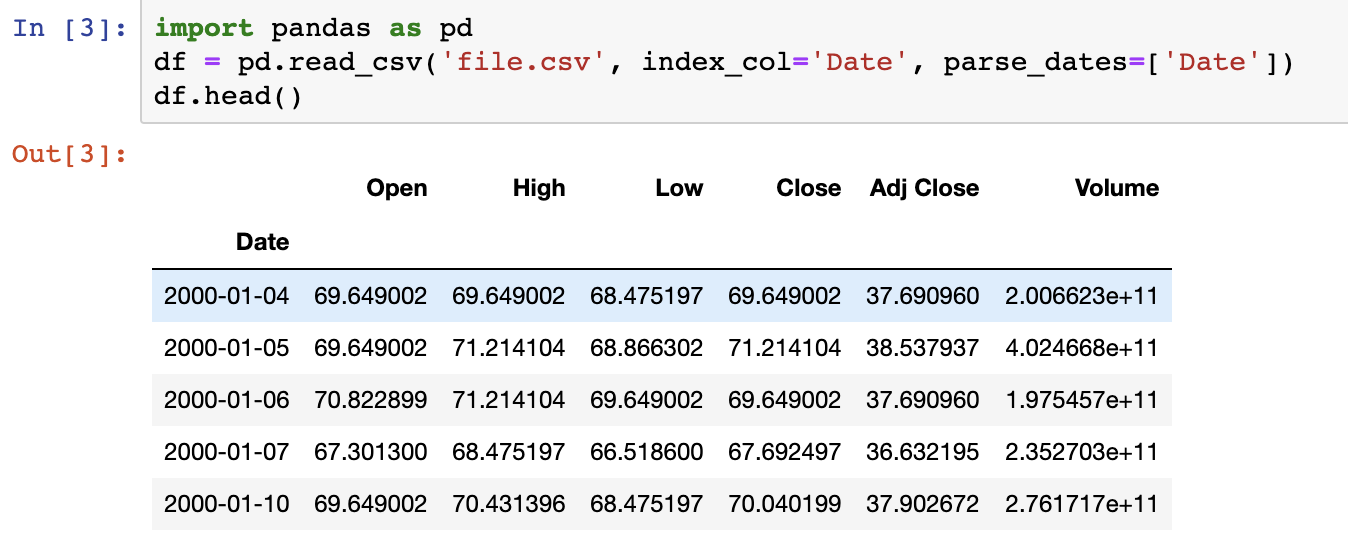
**with open('file.csv', 'w') as f:**

**f.writelines(response.text)**

**將資料讀進來**

接下來我們重新把這個歷史資料讀進來，可以使用pandas這個套件，  
pandas有點像是python版的excel，可以快速的對行列做運算，我們利用pandas的read\_csv將csv檔給讀進來；

並用df.head()印出前面五行：



**import pandas as pd**

**df=pd.read\_csv('file.csv',index\_col='Date',parse\_dates=['Date'])**

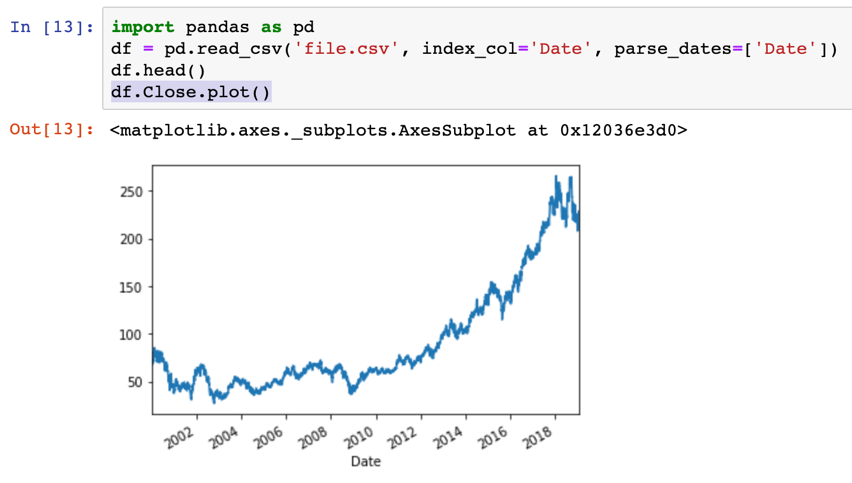
**df.head()**

其中index\_col就是將Date這條column當作是index，而parse\_dates可以將Date轉換成程式瞭解的日期格式，而非單純的字串。

**繪圖**

得到股價後，我們就可以把收盤價的歷史紀錄給畫出來：

**df.Close.plot()**



df = pd.read\_csv('file.csv', index\_col='Date', parse\_dates=['Date'])  
df.Close.plot()

**全球指數一次抓**

我們已經知道怎麼抓取任意一檔股票歷史價格了，但有時候我們一次需要很多檔股票，怎麼辦呢？難道每個股票我們都要寫這麼多行，才能抓下來嗎？今天我們教你如何將上一次的功能包裝成 function，然後一次下載全球指數！

**上次的程式碼爬取台積電**

首先，我們上次已經教了怎麼樣爬取一檔股票：「台積電2330」，這次我們重新複習一下，下方的程式，假如有任何一個部分忘記了

**將上述程式碼打包成function**

但是以上的程式碼只能爬「台積電」的股價，我想要爬其他的指數怎麼辦呢？

我們可以用function將上面的程式碼打包起來，方便我們多次使用，變成下面這樣，其實跟之前長的非常像，可以先比較一下：

首先，第一行多了def crawl\_price(stock\_id):，這行的意思就是宣告一個可呼叫的程式區塊，叫做crawl\_price(stock\_id)，

再來，原本的程式碼前方都被加了4個空白格，代表這些功能是同一個function中被執行的代碼。

上面的代碼中，有個特別可以注意的變數：stock\_id，當今天stock\_id被傳進function中的時候，url就會隨著stock\_id的不同，而出現不同的url所以我們就可以爬到不同的股票歷史股價！

既然我們已經寫好了此function，接下來就可以呼叫它，得到不同股票的歷史股價

**import io**

**import requests**

**import datetime**

**import pandas as pd**

**def crawl\_price(stock\_id):**

**now = int(datetime.datetime.now().timestamp())+86400**

**url = "https://query1.finance.yahoo.com/v7/finance/download/" + stock\_id + "?period1=0&period2=" + str(now) + "&interval=1d&events=history&crumb=hP2rOschxO0"**

**response = requests.post(url)**

**f = io.StringIO(response.text)**

**df = pd.read\_csv(f, index\_col='Date', parse\_dates=['Date'] )**

**return df**

**df = crawl\_price("2354.TW")**

**df.Close.plot()**

**國際重要指數清單**

首先我們必須要爬取到國際重要指數清單，可以到以下網址來獲取：  
<https://finance.yahoo.com/world-indices/>

url = "https://finance.yahoo.com/world-indices/"

response = requests.get(url)

import io

f = io.StringIO(response.text)

dfs = pd.read\_html(f)

world\_index = dfs[0]

dfs[0]

上述的程式碼，剛開始很簡單，就是用requests.get來獲取此網頁的資料，網頁中的資料都存在response.text中，跟之前一模一樣。

不過第5行有點不一樣，我們使用io.StringIO(response.text)將資料存成檔案f  
第6行，我們將此文件利用pd.read\_html(f)來分析網頁f中的表格，將所有的表格存成 a list of dataframe  
第7行，我們將第一張dataframe給拿出來。

**延續上個單元，我們應該已經爬取了world\_index這張表格：**

這個單元我們要利用 for 迴圈將所有的歷史數據爬下來，並且繪製所有的歷史股價。我們也會介紹一些 for 迴圈的進階技巧，並且將這些技巧結合起來，完成這次的功能，想要學習python，就慢慢跟著我們一步一步來吧！

假如你有認真研究前三點功能，相信這些代碼應該難不倒你！其中沒有教到的是，我們利用time.sleep(5)，來休息一下，不要抓的太頻繁，不然會被封鎖，除此之外，上述代碼非常簡單，寫一個for迴圈，將所有的指數都抓下來，存在world\_index\_history中。

**import time**

**world\_index\_history = {}**

**for symbol, name in zip(world\_index['Symbol'], world\_index['Name']):**

**print(name)**

**world\_index\_history[name] = crawl\_price(symbol)**

**time.sleep(5)**

**繪製指數**

**import matplotlib.pyplot as plt**

**plt.rcParams['figure.figsize']=(16,9)**

**plt.style.use('ggplot')**

**for name,history in world\_index\_history.items():**

**history.Close.plot()**

為什麼美股大跌，在台灣的投資人要很緊張？原因就是相關性！想知道台股跟哪國指數最同步嗎？不是S&P 500美國指數，也不是深圳指數，那究竟是哪國呢？這篇文章我們就來研究一下吧！

投資就是預測未來現象與當下現象的相關性！  
就像你看到天空烏雲密佈，就能預測接下來會下雨，  
我們也可以將相關性係數用於全球指數，為什麼美股大跌，在台灣的投資人要很緊張？原因就是相關性！  
台股跟美股有正相關。

我們可以用Python將相關性畫出來，

**print(world\_index\_history['S&P 500'].head())**

還記得world\_index\_history嗎？

它是一個dictionary，根據不同的字串，例如S&P 500，可以叫出相對應的表格（dataframe）

有了 world\_index\_history 我們就可以將國際指數的相關性給畫出來，但首先，我們要先來瞭解

Adj Close跟Close的差異？

close 就是收盤價，每天股市休盤瞬間的最後數字。那Adj Close呢？  
我們用圖會比較好解說，先將 S&P500 指數的 Adj Close 跟 Close 畫出來

**sp500 = world\_index\_history['S&P 500']**

**sp500['Close'].plot(color='red')**

**sp500['Adj Close'].plot(color='blue')**

上圖中紅色的為Close而藍色的為Adj Close，兩者幾乎重疊，因為Adj Close其實跟Close很像，但是有點不同，台灣叫做「調整後收盤價」，你有時候也會看到「復權收盤價」。這個價格的好處是，它移除了除權息和增減資的價格波動，更準確的反映了「股票價值」的變化。

當除權息、增減資時，一般股價會有大波動，以除權息為例，股票價格瞬間下跌

然而帳戶裡的股票張數會變多（例如領股利）  
或是帳戶資金量變多（領股息）  
總體來講是不賺不賠的

所以當除權息時，  
收盤價 Close 會突然下跌很多，  
而調整後收盤價 Adj Close 不會因為除權息有巨幅下跌  
更能夠精確的反映股票價值，跟投資的獲利狀況！  
所以我們在做相關性分析，可以用 Adj Close！

**Python 終於開始了**

首先我們先將所有指數的 Adj Close 先集合起來，變成一張表格(dataframe)

**adjclose = {}**

**for name, price in world\_index\_history.items():**

**if price.index.is\_unique:**

**adjclose[name] = price['Adj Close']**

**adjclose = pd.DataFrame(adjclose)**

**adjclose**

**接下來就可以來觀察每檔指數的 Adj Close 相關性，**

**corr = adjclose.corr()**

**print(corr)**

上圖可以看到一個二維的表，就是每個指數跟每個指數之間的相關性，  
上圖中可以發現，指數自己跟自己的相關性一定是1，代表完全正相關！  
而這張表會是symmetric matrix，也就是做對角線會是對稱的。

接下來我們可以用更視覺化的方式來表示，假如沒有sns的package，要記得先去安裝pip install sns即可，想瞭解如何安裝，可以到[此安裝教學](https://www.finlab.tw/Python%E8%82%A1%E7%A5%A8%E6%8A%95%E8%B3%87/)的下半部分，有簡單的教學喔！

**import seaborn as sns**

**sns.heatmap(corr, square=True ,vmax=1.0, linecolor='white', annot=True)**

字太小

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**plt.rcParams['figure.figsize'] = (18, 18)**

# original code

import seaborn as sns

sns.heatmap(corr, square=True ,vmax=1.0, linecolor='white', annot=True)

台灣股票跟誰最連動

我們可以人眼觀察，發現台灣指數TSEC weighted index跟英國前一百大Cboe UK 100超像！相關係數高達0.9，代表非常非常像

我們可以將這兩個指數畫出來檢查一下：

adjclose['Cboe UK 100 Price Return']['2015':].plot()

adjclose['TSEC weighted index']['2015':].plot()

**如何衡量風險和報酬？**

今天我們來看一下市場上常見的風險與報酬指標：夏普比率（sharpe ratio），這是一個風險跟報酬的比例，有了夏普比率，你就知道為什麼要投資美股而不是台股了！

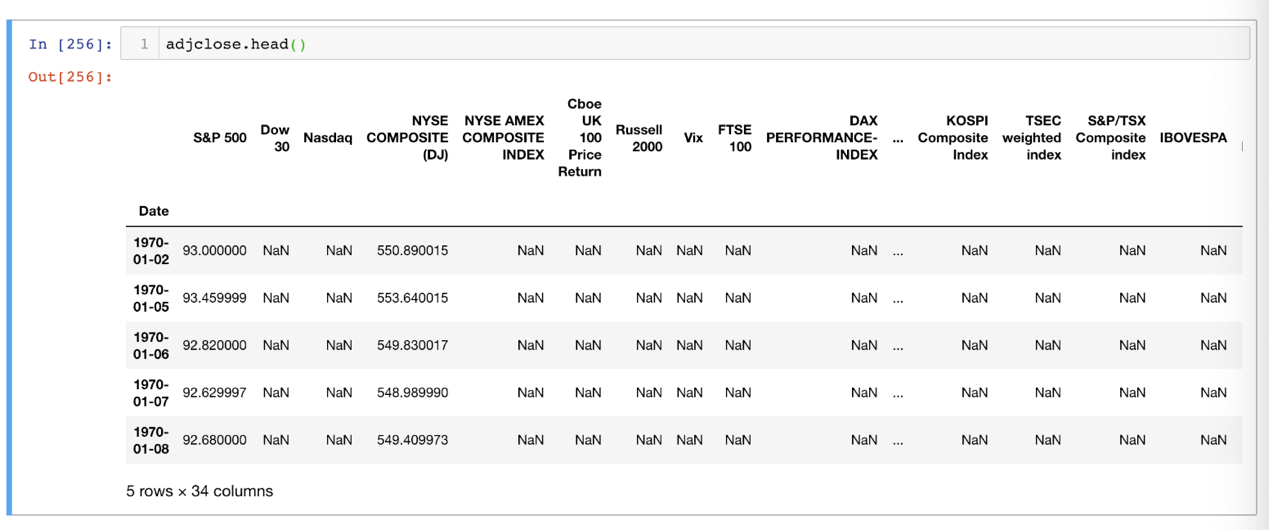
**sharp ratio 簡單講，就是「報酬 / 風險」！**

以這著比率，可以想像，sharp ratio 越高，代表獲利大於風險，  
而sharp ratio 越低，代表風險大於獲利，那就會有點危險了！  
所以找一個sharp ratio 越高的指數，就等於找出了「獲利大且風險相對小」的指數喔！

**如何定義獲利？**

獲利可以用每天平均的漲跌來代表，也就是今天漲1％，明天跌1％，平均獲利就是0％，  
接下來我們就用python來計算每天平均獲利吧

首先，記得回去前一個單元，找出上次的adjclose：



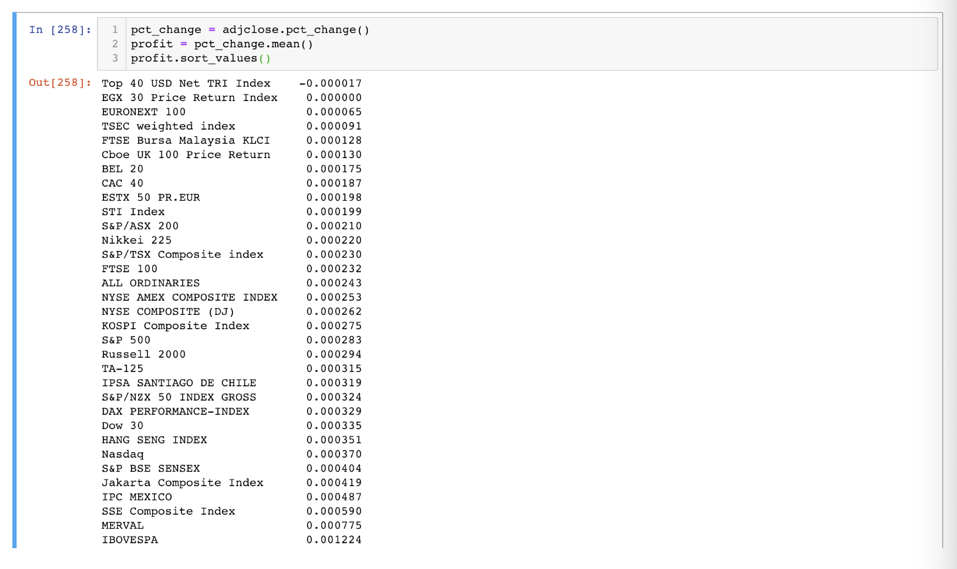
接下來可以計算獲利：

**pct\_change = adjclose.pct\_change()**

**profit = pct\_change.mean()**

**profit.sort\_values()**

上方的程式碼，  
第一行，有好用的功能，叫做adjclose.pct\_change()，這個函示會計算今天漲了x％，並且以x/100來表示，將整個table中的每一條時間序列都進行計算喔！  
第二行，針對每一檔指數，將每一天的漲跌都平均起來，  
第三行，進行由小到大的排序



### 計算sharpe ratio

**risk=pct\_change.std()**

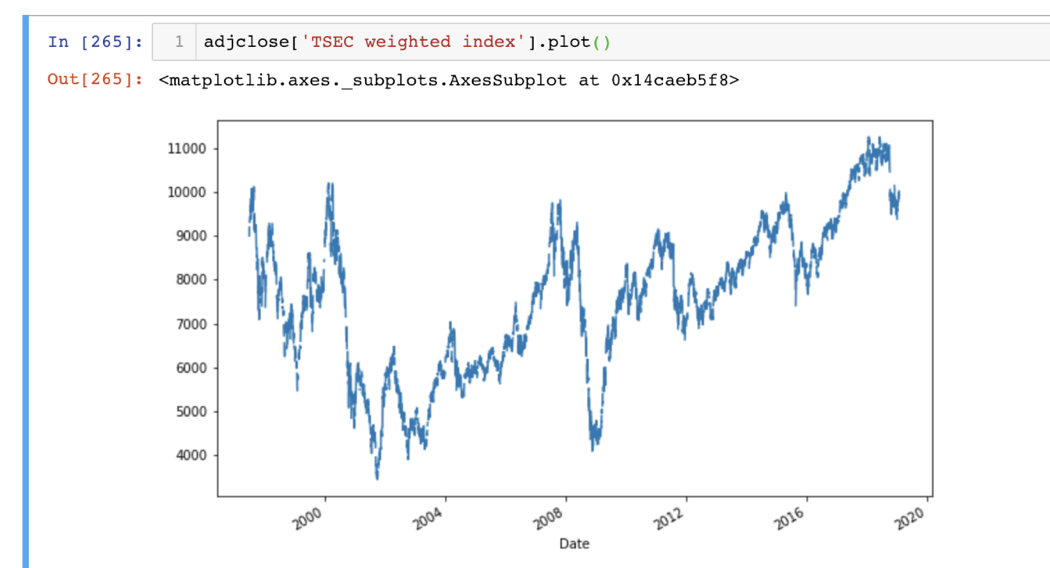
**sharpe = profit / risk \* (252 \*\* 0.5)**

**sharpe.sort\_values()**

可以看到上述程式，我們額外乘了一個「252 \*\* 0.5」  
因為我們希望算年化 annual sharpe ratio，  
其中的252是一年大約的交易天數，  
而「\*\*」是「次方」的意思。  
為什麼要乘這個常數？最主要是因為大家幫自己的歷史回測計算sharpe ratio時候，都有乘上這個數字，要乘了才有辦法跟別人比較XD，  
當然這背後可能有更深層的數學邏輯，但我傾向於這樣理解（懶）。

**可以發現台股真的好慘，慘不忍睹，為什麼會這樣呢，我們可以將歷史圖表畫出來：**

**adjclose['TSEC weighted index'].plot()**



可以發現因為我們是從1998年開始計算的，那個時候剛好也上萬點，跟現在的萬點是同一個萬點，反觀我們來看美股：

可以看到從1998年開始，直到現在漲了將近3倍！  
可見台股的獲利不理想是導致於sharpe ratio比較低的很大的因素。

我們目前計算的sharpe ratio，是所有歷史資料的平均值，  
然而我們知道，雖然台股總平均來說很爛，但是應該有時也有可圈可點之處，  
我們希望用時間移動窗格，每日都計算252天以前的sharpe值，來找出台股表現比較好的時段

**移動窗口**

# before  
profit = pct\_change.mean()  
risk = pct\_change.std()  
sharpe = profit / risk \* (252 \*\* 0.5)  
  
# after  
**profit = pct\_change.rolling(252).mean()  
risk = pct\_change.rolling(252).std()  
sharpe = profit / risk \* (252 \*\* 0.5)**

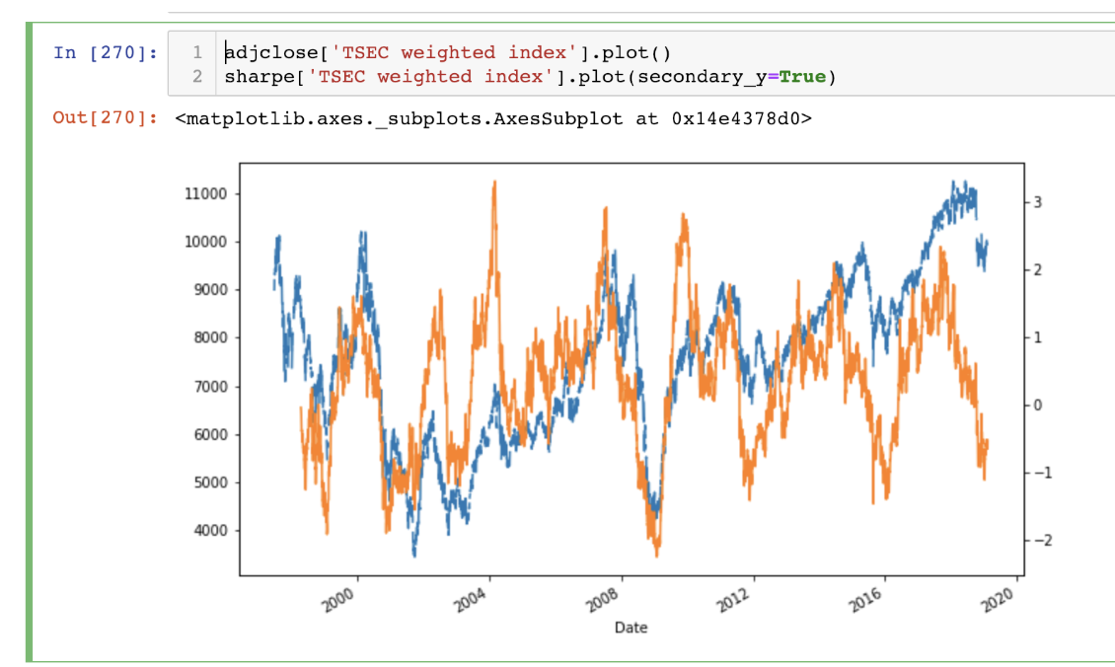
幾乎長的一模一樣對吧？唯一不一樣的是rolling(252)這個功能，  
這是移動窗格252天的意思。  
額外要注意的是，之前的寫法中，sharpe是一個series，index為指數名稱，而在現在的寫法中，sharpe變成了一個dataframe（table），其index代表日期，而columns代表每檔指數，其中的數值是 252 天的 sharpe ratio，神奇吧！

這就是python跟R最強大的資料處理功能！有了這個移動窗格版的sharpe ratio，我們做圖後，就可以來看一些端倪：

**adjclose['TSEC weighted index'].plot()**

**sharpe['TSEC weighted index'].plot(secondary\_y=True)**

以上的代碼可以繪出下面這張圖，其中藍色的為加權指數，而黃色的為sharpe ratio，由於這兩個時間序列的數值差非常多，台股可能是在4000～10000左右，而sharpe可能是在-2～2左右，所以上方第二行程式中，我們用secondary\_y=True這個參數，來將兩個數值的座標分開，所以下方的圖中，可以看到sharpe ratio的大小標示在右邊。

****

可以看到，sharpe ratio （黃）在台股加權指數（藍）高點時，會比較大，而台股低點時，會比較低  
圖中還可以看出，在大盤高點時，sharpe ratio會領先大盤往下落，接下來我們就可以利用這個特點，來模擬一些買賣的實驗。我們就來創造一個夏普策略吧！

上面我們介紹了Sharpe ratio，可以用來衡量風險跟報酬的指標（也就是報酬 / 風險），這集我們就利用Sharpe ratio來進行台股的模擬買賣，假裝我們這20年來，都使用sharpe ratio的策略，可以得到多少獲利呢？

**首先，我們得準備台股的歷史紀錄，還有台股的夏普指標**

**adjclose['TSEC weighted index'].plot()**

**sharpe['TSEC weighted index'].plot(secondary\_y=True)**

為了使用python寫出上述的策略，我們要先將夏普值平滑一下，不然雜訊太多了：

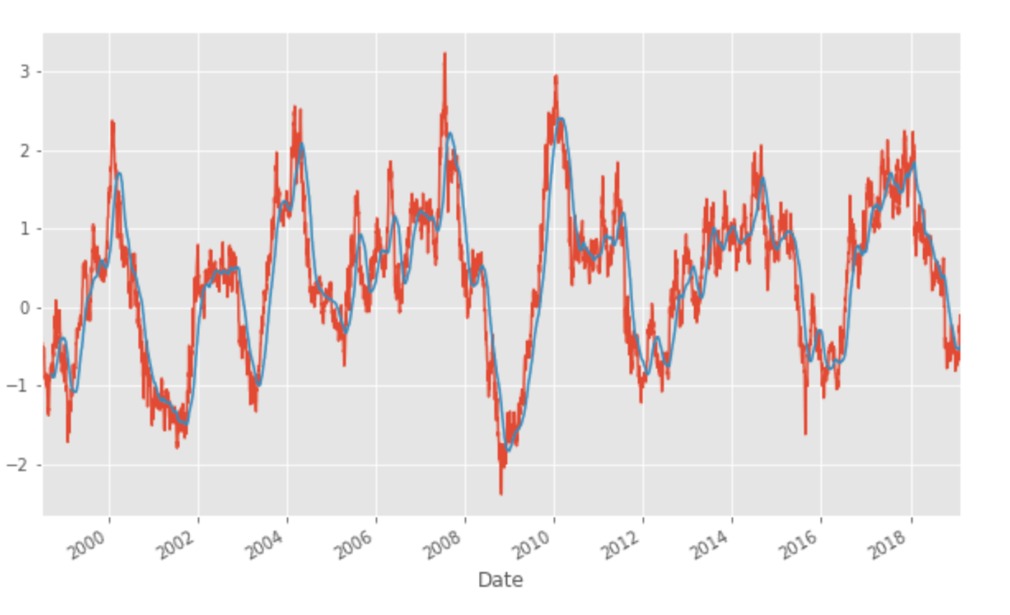
**sr = sharpe['TSEC weighted index'].dropna()**

**d = 60**

**srsma = sr.rolling(d).mean()**

**sr.plot()**

**srsma.plot()**

****

來色的線是我們就將sharpe ratio做移動窗格的平均，可以發現平均之後，時間序列比較平滑，這樣子我們找轉折點比較方便，所謂的轉折點，就是斜率由正到負，或由負到正的瞬間，所以我們要先找出夏普曲線的斜率。

**夏普曲線的斜率**

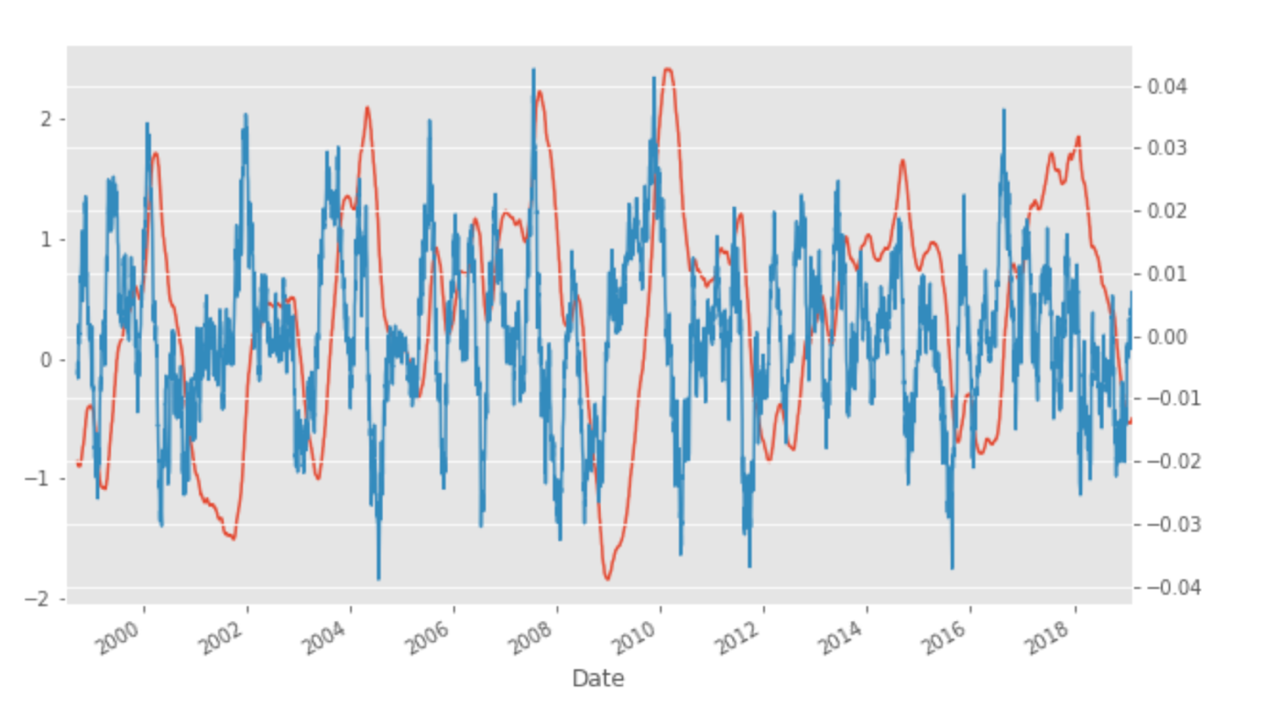
斜率非常簡單，可以使用diff這個功能：

**srsma = sr.rolling(d).mean()**

**srsmadiff = srsma.diff()**

**srsma.plot()**

**srsmadiff.plot(secondary\_y=True)**

****

可以發現上圖中，橘色的為sharpe ratio，藍色的為斜率，當橘色線由上而下轉折時，藍色的線會快速向下穿越0，有了這個特性，我們就可以來找轉折點了！

**找到轉折點**

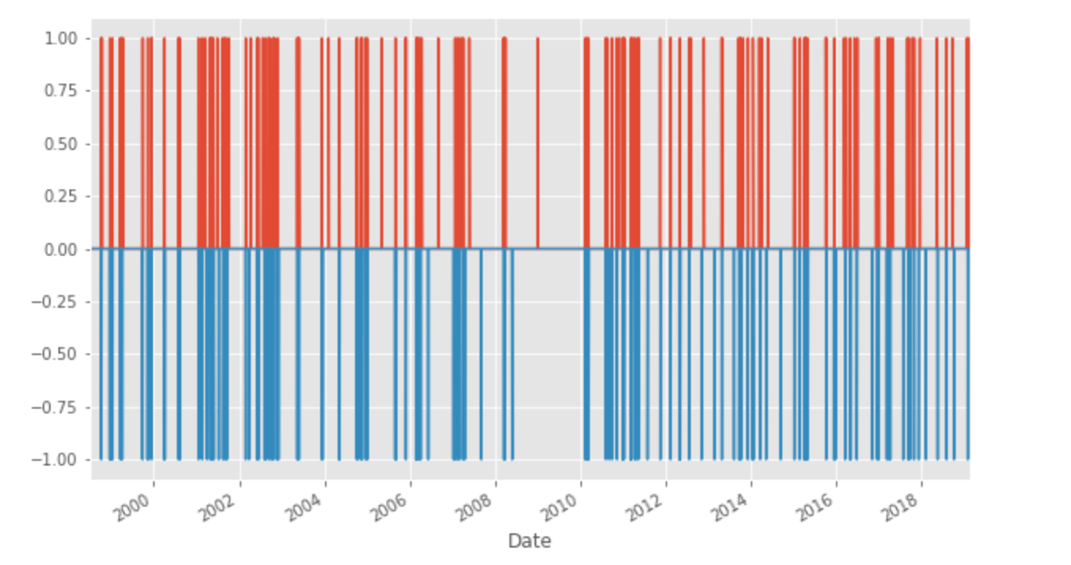
接下來我們可以來找轉折點了，就是斜率由正到負，或由負到正的瞬間。

**buy = (srsmadiff > 0) & (srsmadiff.shift() < 0)**

**sell = (srsmadiff < 0) & (srsmadiff.shift() > 0)**

**(buy \* 1).plot()**

**(sell \* -1).plot()**

****

**找出持有時間**

那我們就可以來看一下，假如天都用一樣的方式來產生這些訊號，當 buy訊號為True時，買入，而當sell=True時空手，如此執行20年的持有加權指數的時段：

**import numpy as np**

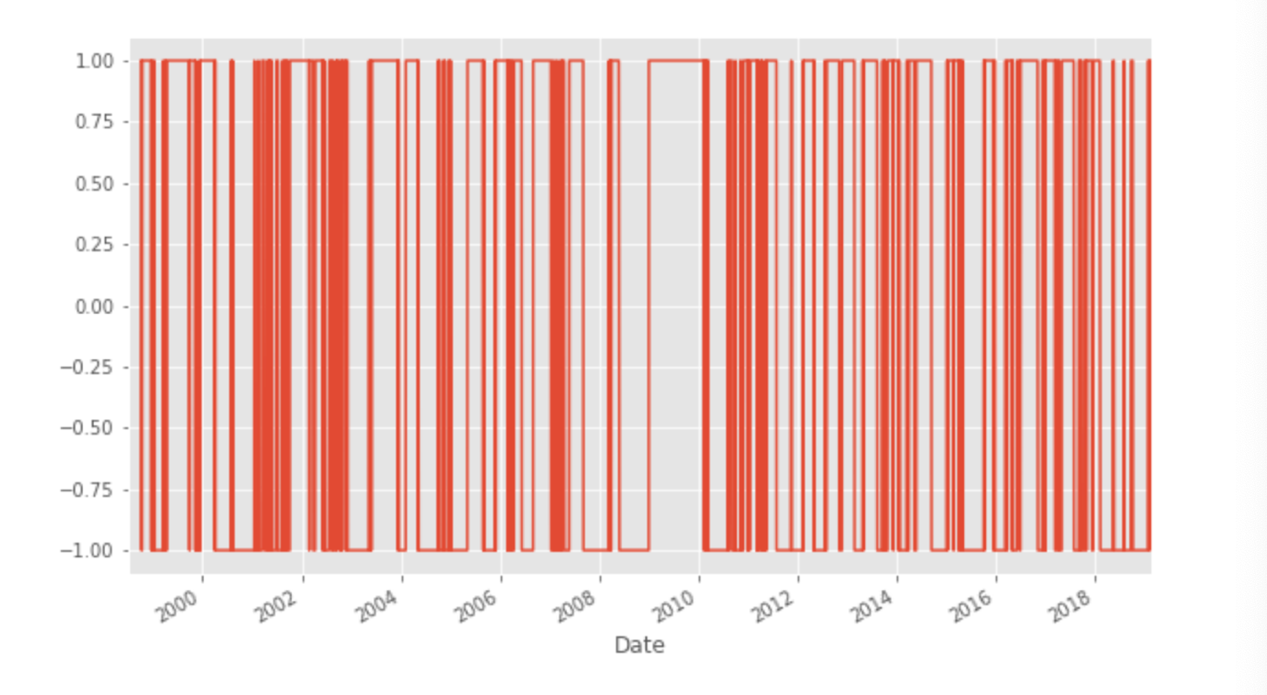
**hold = pd.Series(np.nan, index=buy.index)**

**hold[buy] = 1**

**hold[sell] = -1**

**hold.ffill(inplace=True)**

**hold.plot()**

****

交易頻率似乎有點高，不過沒關係，我們之後還會再做調整  
接來是回測

**回測**

今天我們先簡單算一算，不考慮手續費，但是真實情況是必須考慮的喔！請謹記在心

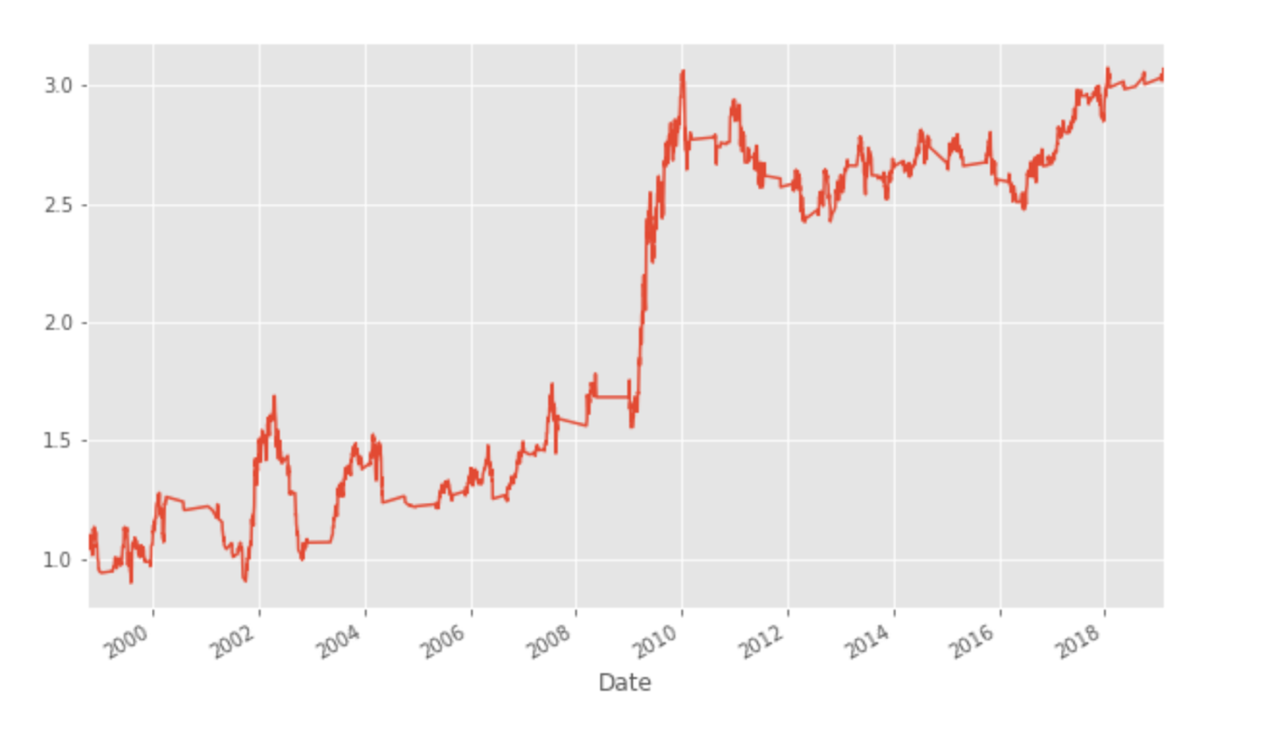
**twii = adjclose['TSEC weighted index'][buy.index]**

**pct\_change = twii.pct\_change()**

**pct\_ratio = (pct\_change.shift(-1)+1) # 今天到明天的價格變化**

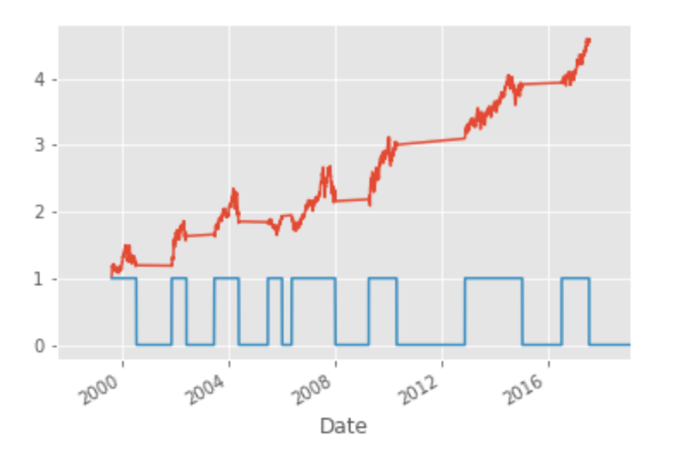
**pct\_ratio.fillna(1)[hold == 1].cumprod().plot()**

這段程式碼，有點複雜，當中的pct\_change是一個每天獲利上下 x％。  
而pct\_ratio代表買入之後每天的變化（不漲不跌是1，大於1則漲，小於1則跌）  
我們希望將「持有」時間段的pct\_ratio全部都乘起來，代表獲利。

****

上一篇文章中，帶大家寫了一個簡單的策略，  
然而，在現實生活中並沒有這麼管用，20年才賺三倍！？

所以這篇文章將帶介紹如何利用修改參數，來調整策略，進而達到更好的績效  
但是人工調整參數很浪費時間，所以我們先使用簡單暴力法，來調整參數試試看。成果如下：



### 先回顧上次的策略

由於這是系列文章，要完成到上次的步驟其實有點煩瑣，  
所以這邊就簡單的前情提要一下  
總共有三個步驟：

1. 下載台股大盤資料
2. 編寫台股的sharpe ratio
3. 利用sharpe ratio製作回測

這邊就不厭其煩的先把上次的code拿來，方便大家直接複製貼上

**1.下載台股大盤資料**

import io

import requests

import pandas as pd

import datetime

import matplotlib.pyplot as plt

%matplotlib inline

plt.style.use('ggplot')

def crawl\_price(stock\_id):

now = int(datetime.datetime.now().timestamp())+86400

url = "https://query1.finance.yahoo.com/v7/finance/download/" + stock\_id + "?period1=0&period2=" + str(now) + "&interval=1d&events=history&crumb=hP2rOschxO0"

response = requests.post(url)

f = io.StringIO(response.text)

df = pd.read\_csv(f, index\_col='Date', parse\_dates=['Date'] )

return df

twii = crawl\_price("^TWII")

twii.head()

**2.編寫台股sharpe ratio**

mean = twii['Adj Close'].pct\_change().rolling(252).mean()

std = twii['Adj Close'].pct\_change().rolling(252).std()

sharpe = mean / std

twii.Close.plot()

sharpe.plot(secondary\_y=True)

**3.編寫台股sharpe ratio策略**

import numpy as np

# sharpe ratio 平滑

sr = sharpe

srsma = sr.rolling(60).mean()

# sharpe ratio 的斜率

srsmadiff = srsma.diff()

# 計算買入賣出點

buy = (srsmadiff > 0) & (srsmadiff.shift() < 0)

sell = (srsmadiff < 0) & (srsmadiff.shift() > 0)

# 計算持有時間

hold = pd.Series(np.nan, index=buy.index)

hold[buy] = 1

hold[sell] = 0

hold.ffill(inplace=True)

hold.plot()

# 持有時候的績效

adj = twii['Adj Close'][buy.index]

(adj.pct\_change().shift(-1)+1).fillna(1)[hold == 1].cumprod().plot()

**4.參數最佳化**

def backtest(a, b, c, d):

sr = sharpe

srsma = sr.rolling(a).mean()

srsmadiff = srsma.diff() \* 100

ub = srsmadiff.quantile(b)

lb = srsmadiff.quantile(c)

buy = ((srsmadiff.shift(d) < lb) & (srsmadiff > ub))

sell = ((srsmadiff.shift(d) > ub) & (srsmadiff < lb))

hold = pd.Series(np.nan, index=buy.index)

hold[buy] = 1

hold[sell] = 0

hold.ffill(inplace=True)

adj = twii['Adj Close'][buy.index]

# eq = (adj.pct\_change().shift(-1)+1).fillna(1)[hold == 1].cumprod().plot()

# hold.plot()

eq = (adj.pct\_change().shift(-1)+1).fillna(1)[hold == 1].cumprod()

if len(eq) > 0:

return eq.iloc[-1]

else:

return 1

backtest(252,0.4,0.6,4)

可以發現，這個function傳入了四個參數「a,b,c,d」，  
而這四個參數是做什麼的呢？是拿來取代原本的數字的，  
可以發現原本的常數部分，都被換成了代數，這樣我們到時候在呼叫時，就可以帶入不同的參數  
而我們最後的回傳值，原本是一張圖片，但此function中被改成了這20年的報酬率  
所以當我們執行「backtest(252,0.4,0.6,4)」的時候，  
這20年的報酬就是9.1%，非常爛  
所以我們才需要做參數優化

**參數枚舉優化**

maxeq = 0

for a in range(100,200,20):

for b in np.arange(0.3, 0.9, 0.03):

for c in np.arange(0.3, 0.6, 0.03):

for d in range(60, 180, 10):

eq = backtest(a,b,c,d)

if maxeq < eq:

maxeq = eq

print(eq, a,b,c,d)

上面第8行，即是我們執行backtest的結果，  
假如我們發現eq，有最高報酬率，  
則將新的最高報酬率print出來，並且print它的參數  
我們就可以看到數字不斷增加的感覺，滿開心的！

不過上述程式要跑滿久的，請耐心等待，

最後成果滿不錯的，算是一個懶人投資策略：

**backtest(100,0.6899999999999996,0.4799999999999998,150)**