投稿類別：資訊類

篇名：

Python 雲端台股查詢程式

作者：

涂允澤。桃園市立永豐高級中學。高三七班

指導老師:

曹齡齡老師

Python 雲端台股查詢程式

壹、前言

一、研究動機

科技日興月異，網頁的發展極為快速，現今已有許多架設網站的方法，例如 google 的協作平台，或是寫網頁的三大元素：HTML、CSS、JAVASCRIPT，自從我有記憶以來，就有各式各樣的網頁出現在我眼前，這讓我愈加的好奇一個網站是如何被架設出來的，很剛好的我在暑假時期接觸了 python 這個程式語言，也發現了可以用 flask 架設網站，激起了我的興趣，去研究如何架設一個網站。

二、研究目的

網路上有許多網站，各個個有特色，因此使得我也想架設一個，利用所學到的 python 程式來去撰寫程式，並且寫出查詢台股資料的網頁與 line 機器人，一方面在當中學到更多的程式撰寫，也可更加學習如何使用模組和網頁架設，另一方面也可多多了解雲服務，進一步讓程式寫得更好。

三、研究方法

(一) 實作架設網站：

架設網站，並且將架設好的網站上傳到 Google Cloud Platform(GCP)。

(二) 實作 line 機器人：

架設 line 機器人，用來傳送台股資訊，也可查詢每日股票資訊。

(三) 創造 firebase 帳號：

創造 firebase 資料庫用來儲存每日台股資料。

貳、正文

一、何謂雲服務

過去從大企業到中小企業，開發資訊系統時，都需要將其部署在自家的伺服器，而機房與伺服器等資訊系統的基礎建設、需要面對硬體折舊、網路成本、機房用電管控等問題，需要花費大量的人力與金錢資源，因此一些大公司就出現了把機房租出去給中小企業來去上傳自家的程式，上傳之後就由大公司跑動和維護硬體設備等等，連普通人也可

以上傳自己的程式上去，就可實現 24 小時程式都在跑的願望了。

二、需要用到的模組

(一) FLASK：

Flask 是一個使用 Python 編寫的輕量級 Web 應用框架，由於其輕量特性，也稱為 micro-framework（微框架），Flask 最顯著的特點是它是一個“微”框架，輕便靈活，但同時又易於擴展，也正是這種靈活的特性使許多人喜歡使用它，在 python 中只要下載並導入模組後即可使用，與 Django 比起來也簡單許多。

(一) BS4：

BS4 是一個可以從 HTML 檔案中提取資料並且進行抓取資料與分析的動作，優點是操作簡單沒有複雜的結構要求，通過提供面向物件的操作方式將文件物件中的各種節點、標籤、屬性、內容等等都封裝成了 python 中物件的屬性，對 html 檔案的處理效率高。

(二) Requests：

使用 Python 來下載網頁上的資料，最基本的作法就是以 requests 模組建立適當的 HTTP 請求，透過 HTTP 請求從網頁伺服器下載指定的資料，可說是網頁爬蟲中相當重要的一個模組。

(三) firebase_admin：

是連線到 firebase 資料庫的方法，如果是在自己測試時會需要使用自己資料庫的金鑰來去做連線，但在上傳 GCP 時會自動連線，就不用金鑰了，前提是在創建資料庫時需要選取在 GCP 上的專案。

(四) linebot：

Line 提供給 Python 控制 Line 聊天機器的的模組，可以控制機器人去發送文字、圖片或是連結等等。

三、python 程式撰寫

(一) 網頁爬蟲

```python
def yahoo_stock_crawler(stock_id):
    doc = requests.get(f"https://tw.stock.yahoo.com/q/q?s={stock_id}")
    html = BeautifulSoup(doc.text, 'html.parser')
    table = html.findAll("table", {"border": 2})[0]
    data_row = table.select("tr")[1].select("td")
    return {
        "open": float(data_row[8].text),
        "high": float(data_row[9].text),
        "low": float(data_row[10].text),
        "close": float(data_row[2].text),
        "lastClose": float(data_row[7].text),
        "dailyReturn": float(data_row[2].text)/float(data_row[7].text)-1
    }
def dayfind(stock_number):
    doc = requests.get(f"https://tw.stock.yahoo.com/q/q?s={stock_number}")
    html = BeautifulSoup(doc.text, 'html.parser')
    tables = html.findAll("table" ,{"cellpadding": "1"})
    table = tables[0]
    td = table.findAll("td")
    day = td[1].text
    day=day.split(' ')
    return day[1]
```

圖(一)網頁爬蟲

這是一個網頁爬蟲的函式，會將所傳入的股票代號去到 yahoo 股票網頁中，因為 yahoo 股票網頁的網址裡的股票代號是由一個變數 s 去做存取，所以將網址中要寫入股票代碼的地方變成這個函式中傳入的 stock_id 後發送請求到網址去，並且蒐集特定的股票資料，和資料的日期。

(二) 每日蒐集台股資料

```python
def yahoo_stock_crawler(stock_id):
    doc = requests.get(f"https://tw.stock.yahoo.com/q/q?s={stock_id}")
    html = BeautifulSoup(doc.text, 'html.parser')
    table = html.findAll(text="個股資料")[0].parent.parent.parent
    data_row = table.select("tr")[1].select("td")

    return {
        "open": float(data_row[8].text),
        "high": float(data_row[9].text),
        "low": float(data_row[10].text),
        "close": float(data_row[2].text),
        "lastClose": float(data_row[7].text),
        "dailyReturn": float(data_row[2].text)/float(data_row[7].text)-1
    }

firebase_admin.initialize_app()
db = firestore.client()

def get_tw_stock_data(request):
    stock_ids = db.collection("watch_list").document("stocks").get().to_dict()["watch_list"]

    doc_id = time.strftime("%Y%m%d")

    for sid in stock_ids:
        print(f"開始截取 {sid} 的資料")
        data = yahoo_stock_crawler(sid)
        db.document(f"{sid}_daily_data/{doc_id}").set(data)

    return "台股資料更新完畢！"
```

圖(二)定時蒐集台股資料

這是使用 GCP 中的 Cloud Functions 來去做的，Cloud Functions 就是一個可以直接在 GCP 上寫程式，不需要寫完之後再上傳，適合小型程式，也可與 Cloud Scheduler 搭配使用，Cloud Scheduler 就是一個可以把程式排程，讓程式一到時間自動跑動，使用這兩個東西就可做出定時抓取台股資料程式了。

(三) 網頁部分

```python
@app.route("/callback", methods=['POST'])
def callback():
    signature = request.headers['X-Line-Signature']
    body = request.get_data(as_text=True)
    print(body)
    try:
        handler.handle(body, signature)
    except InvalidSignatureError:
        abort(400)
    return 'OK'

@app.route("/", methods=["GET"])
def loby():
    return render_template("loby.html")

@app.route("/index.html", methods=["GET"])
def greet():
    number = request.args.get("number")
    test = db.collection(f"{number}_daily_data").document(f"{time.strftime('%Y%m%d')}").get()
    if test.to_dict() == None:
        data = yahoo_stock_crawler(number)
        day = dayfind(number)
    else:
        data = test.to_dict()
        day = time.strftime('%Y/%m/%d')
    data['dailyReturn']=float(round(data['dailyReturn']*100, 2))
    return render_template("index.html", number=number,data=data,day=day)
```

圖(三)flask 網頁程式

1、第一個 route 中是為了 line 的機器人而寫的，因為 line 機器人運作方式是 line 那邊的伺服器會先接收使用者傳送的訊息，並且將訊息傳到自己架設的後端 app 中，來去偵測要回覆什麼，在每次傳入訊息時也會發送 callback 這個請求來去檢查這個後端程式是不是合法並且安全的。

2、接下來兩個 route 都是為了前端網頁而寫的，中間這個就是簡單的做一個主頁並呈現出來，主頁中可以打想要查詢的股票代碼，之後會將輸入的代碼傳到最後一個 route 中來去做查詢今日資料，會先到 firebase 資料庫中找有沒有今天的資料，如果沒有就會將代碼傳到爬蟲程式中，來去及時擷取最新資料，並且呈現在網頁中。

(四) Line 機器人

1、當 line 收到用戶發來的訊息時就會傳入這個函式中，所以只要在這個函式中寫入想要看到的關鍵字，如果有讀到，就去做甚麼，如果想要新增更多關鍵字，就在這函式中用 if、elif 寫下去就行了，例如當用戶輸入“查詢 2330”時，偵測到了查詢，就會把 2330 傳入 createReplyMessge 這個函式中。

```python
@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    if "查詢" in event.message.text:
        sid = event.message.text.split()[1]
        line_bot_api.reply_message(
            event.reply_token,
            TextMessage(text=createReplyMessge(sid), type="text")
        )
```

圖(四)line 查詢指令

createReplyMessge 與上述第三個 flask 第三個 route 非常類似，也是將股票代號傳入函式，並先查詢 firebase 中有無資料，沒有的話一樣傳到爬蟲程式中，最後回傳抓取到的資料，並叫機器人傳送給用戶。

```python
def createReplyMessge(sid):
    doc = db.collection(f"{sid}_daily_data").document(f"{time.strftime('%Y%m%d')}").get()
    if doc.to_dict() == None:
        data =  yahoo_stock_crawler(sid)
    else:
        data = doc.to_dict()

    replyCheckMessage = ("PYGCP BOT\n\n"
                         f"開盤價：{data['open']} 元\n"
                         f"最高價：{data['high']} 元\n"
                         f"最低價：{data['low']} 元\n"
                         f"收盤價：{data['close']} 元\n"
                         f"漲幅：{ round(data['dailyReturn'] * 100, 2) }\n")

    return replyCheckMessage
```

圖(五) createReplyMessge 函式

2、當有偵測到“教學”或“help”在用戶輸入訊息中時就會傳出 help_txt 來教使用者如何使用指令。

```python
elif "help" in event.message.text or "教學" in event.message.text:
    line_bot_api.reply_message(
        event.reply_token,
        TextMessage(text=help_txt, type="text")
    )
```

圖(六)教學、help 指令

3、用戶輸入“新增 2330”，當有偵測到“新增”，就會將 firebase 中的 watch_list 裡的 stocks 讀取出來，並檢查裡面有無 2330 沒有的話就新增，並回傳“新增完成”，如果有的話，就不新增，並回傳“已在每日蒐集中”，刪除也是同道理只是回傳不同而已，為甚麼會有 watch_list 在 firebase 中呢? 因為上述提到過的每日蒐集台股資料中，就是依照 watch_list 中的股票代號去做每日抓取的，因為如果把 watch_list 用程一個陣列儲存在程式碼中的話很難去做新增與刪除，才將它存到 firebase 中。

```
elif "新增" in event.message.text:
    a = event.message.text.split()[1]
    test = db.collection("watch_list").document("stocks").get()
    test = test.to_dict()
    if a not in test["watch_list"]:
        test["watch_list"].append(a)
        db.collection("watch_list").document("stocks").set(test)
        line_bot_api.reply_message(
            event.reply_token,
            TextMessage(text="新增完成", type="text")
        )
    else:
        line_bot_api.reply_message(
            event.reply_token,
            TextMessage(text="已在每日蒐集中", type="text")
        )
```

圖(六)新增指令

```
elif "刪除" in event.message.text:
    a = event.message.text.split()[1]
    test = db.collection("watch_list").document("stocks").get()
    test = test.to_dict()
    if a in test["watch_list"]:
        test["watch_list"].remove(a)
        db.collection("watch_list").document("stocks").set(test)
        line_bot_api.reply_message(
            event.reply_token,
            TextMessage(text="刪除成功", type="text")
        )
    else:
        line_bot_api.reply_message(
            event.reply_token,
            TextMessage(text="此代碼不在每日蒐集中喔!", type="text")
        )
```

圖(七)刪除指令

4、用戶輸入“每日蒐集股票代碼”回傳在每日蒐集中的股票代碼股票代碼列表。

```
elif "每日蒐集股票代碼" in event.message.text:
    test = db.collection("watch_list").document("stocks").get()
    test = test.to_dict()
    for i in test["watch_list"]:
        line_bot_api.push_message(user_id, TextSendMessage(text=i))
```

圖(八)每日蒐集股票代碼指令

四、網頁程式撰寫

(一) 首頁

首先使用網址來去設定背景，使用這個方法的缺點是，如果在網路上的圖片被移

除，就沒辦法使用了，就要換新背景，並且使用 form 標籤來去製作輸入與按鈕，並在使用者輸入後將網址導到 index.html 這個 route。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>stock</title>
</head>
<body background="https://img.wantgoo.com/WantGooFiles/UpdateFiles/98845/9f65905a-af35-4858-9012-66c3ce031620.jpg"
style=" background-repeat:no-repeat ;
background-size:100% 100%;
background-attachment: fixed;">
<h1>台股查詢</h1>
<form action="index.html" method="GET">
    <div>
        <label for="name">股票代碼:</label>
        <input name=number>
    </div>
    <br />
    <br />
    <button type="submit">submit</button>
</form>
</body>
</html>
```

圖(九)首頁程式

(二) 資料網頁

將資料傳到這個網頁後，以表格的形式呈現，並且在最後加上一個按鍵，讓使用者可以回到首頁繼續查詢想要的資料。

```
<body background="https://img.wantgoo.com/WantGooFiles/UpdateFiles/98845/9f65905a-af35-4858-9012-66c3ce031620.jpg"
style=" background-repeat:no-repeat ;
background-size:100% 100%;
background-attachment: fixed;">
    <h1>台股查詢</h1>
        <h1>Your stock is: {{number}}</h1>

<font>資料日期:{{day}}</font>
<table style="border:3px #cccccc solid;" cellpadding="10" border='1'>
  <tr>
    <th><font size="5">股票代號</font></th>
    <th><font size="5">開盤價</font></th>
    <th><font size="5">最高價</font></th>
    <th><font size="5">最低價</font></th>
    <th><font size="5">收盤價</font></th>
    <th><font size="5">漲幅</font></th>
  </tr>
  <tr>
    <td><font size="5">{{number}}</font></td>
    <td><font size="5">{{data['open']}}</font></td>
    <td><font size="5">{{data['high']}}</font></td>
    <td><font size="5">{{data['low']}}</font></td>
    <td><font size="5">{{data['close']}}</font></td>
    <td><font size="5">{{data['dailyReturn']}}</font></td>
  </tr>
</table>
<br />
<br />
<div id="results"></div>
<a href="/">回查詢</a>
```

圖(十)資料程式

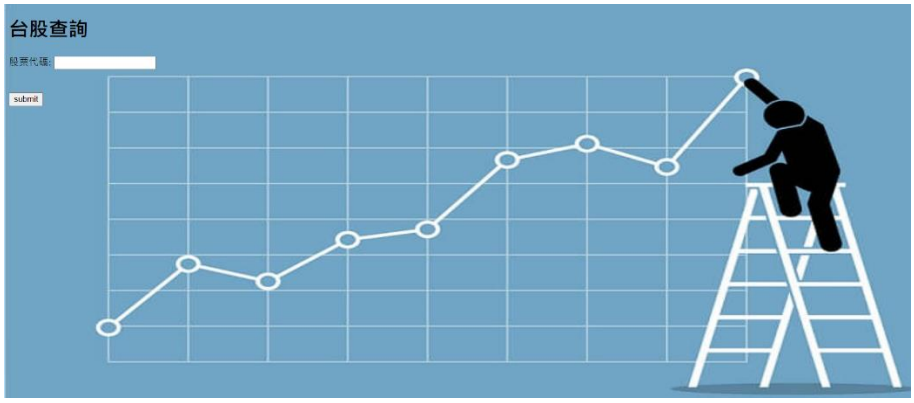
貳、結論

一、成果

(一) 網頁

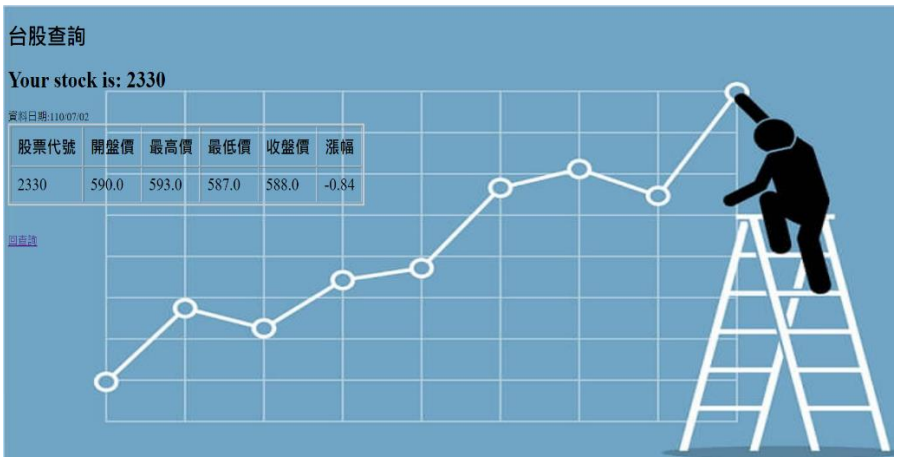
所在網址 https://ntupygcp-340.df.r.appspot.com/。

1、查詢畫面



圖(十一)查詢畫面

2、股票資料



圖(十二)查詢成果

(二) Line 機器人

機器人 id：“ 424cnvit ” 。

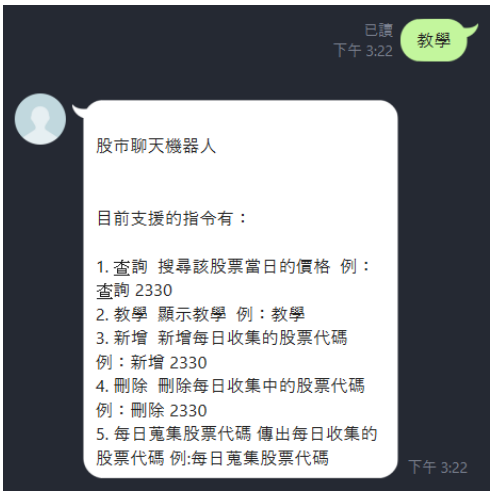
1、查詢指令

輸入“查詢 2330”，並回傳 2330 股票資訊。

圖(十三)查詢結果

2、教學指令

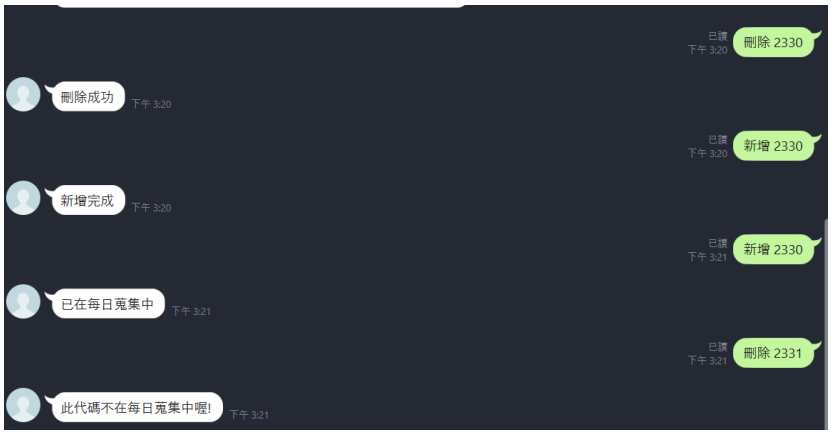
輸入“教學”，並回傳指令教學。



圖(十四)指令教學

3、新增、刪除指令

輸入“新增或刪除 2330”，並改變每日蒐集列表。



圖(十五)新增刪除

4、每日蒐集股票代碼

輸入“每日蒐集股票代碼”，並回傳每日蒐集列表。

二、研究心得

經過這次的研究使的我更加了解了何謂雲服務，和增進寫程式的能力，也了解了 HTML 的基本架構，在過程中也遇到了一些困難，如程式有錯誤，和這個程式本身有的缺點，就是當網站架設者改變網站格式後，爬蟲程式也要跟著去做相對應的改變，等等的一些 BUG，但也正是這些錯誤讓我了解了我在寫程式上的一些不足，也打算在未來繼續改良架設的網站，來去分析和抓取更多資料，讓網頁更加精美、呈現更多資料，也使自己寫程式的能力更加進步。

參、引註資料

Flask 介紹
https://blog.techbridge.cc/2017/06/03/python-web-flask101-tutorial-introduction-and-

environment-setup/

bs4 介紹
https://www.itread01.com/content/1548582670.html

Requests 介紹
https://ithelp.ithome.com.tw/articles/10206215