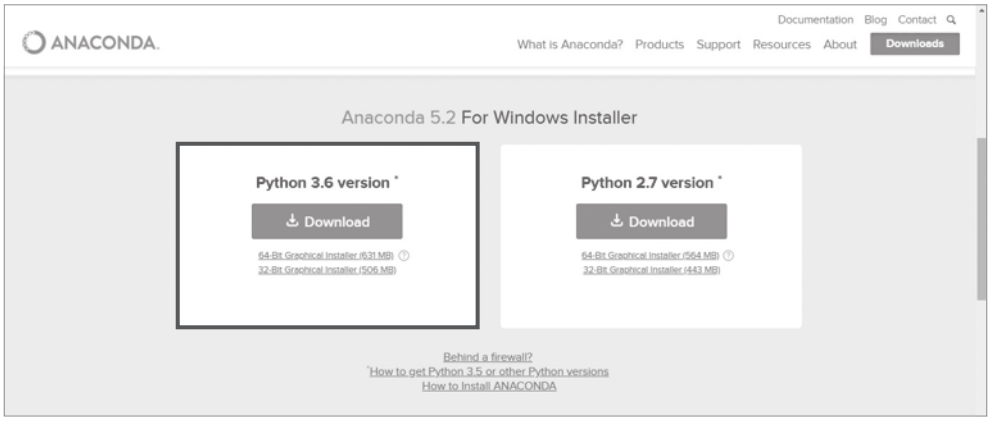


安裝完成後，只要在「搜尋」輸入 Anaconda，可找到剛剛安裝成功的 Anaconda 軟體，雙擊兩下即可打開軟體。打開後，上方已經安裝好開發中可能會使用到的軟體，只要雙擊兩下即可執行，本書中會使用到的開發環境只有 Spyder，其他的開發環境可以依個人的需求做更改。

程式碼

```
float_var = 10.51162319884
float_var
```

執行結果

```
10.51162319884
```

前面字串有講到，可以使用 str()，將任何型態轉換成字串，那是不是也可以將浮點數強制轉換成整數呢？ 答案是可以的，int() 可以將浮點數直接轉換成整數，但小數點後方的數字會被刪除，這可能會造成計算上的誤差值，在程式設計上，這點必須要考慮進去。

程式碼

```
float_var = 10.51162319884
int(float_var)
```

執行結果

```
10
```

Dataframe

Dataframe 是一種專門拿來處理二維資料的格式（三維以上也可以，但不方便），是資料分析時許多人的首選。Dataframe 的底層是用 C 語言構成，在執行上非常快速，並且會幫你節省記憶體，以防電腦當機。舉例來說，如果要打開一份 140 萬筆資料的 CSV 檔案，若是用一般的文書編輯器打開，就算打得開也要開個 5 分鐘，但 Dataframe 只要幾秒鐘就可以完成了，為何它可以如此快速呢？由下圖可知，當你打開 Dataframe 變數的時候，只會顯示頭尾，中間的部分都省略了，基本上如果欄位過多，就算全部顯示出來也很難閱讀，因此 Dataframe 就利用這樣的設計方式，來節省記憶體空間。

	日期	成交股數	成交金額	...	漲跌價差	成交筆數	Unnamed: 9
0	107/10/01	18,382,123	468,595,122	...	0.00	6,838	NaN
1	107/10/02	17,393,413	438,733,554	...	-0.30	5,830	NaN
2	107/10/03	18,858,066	459,120,261	...	0.25	6,872	NaN
3	107/10/04	14,457,266	365,236,442	...	-0.05	4,843	NaN
4	107/10/05	20,297,998	508,520,882	...	-0.30	7,338	NaN
5	107/10/08	14,358,056	362,741,167	...	0.15	5,798	NaN
6	107/10/09	27,285,894	692,297,484	...	0.10	8,978	NaN
7	107/10/11	62,854,866	1,513,774,465	...	-1.15	20,842	NaN
8	107/10/12	27,918,321	677,383,390	...	0.30	12,270	NaN
9	107/10/15	22,322,907	536,652,872	...	-0.45	6,888	NaN
10	107/10/16	22,881,096	553,344,384	...	0.40	8,896	NaN
11	107/10/17	12,249,140	297,227,687	...	-0.30	6,297	NaN
12	107/10/18	11,432,212	277,444,913	...	0.10	3,759	NaN
13	107/10/19	14,498,046	352,403,350	...	0.05	3,229	NaN
14	107/10/22	9,885,865	217,555,917	...	-0.15	3,240	NaN

當拿到一筆 CSV 資料時，首先要了解該資料大概是什麼內容。Dataframe 提供了幾個方法，能迅速了解該筆資料。利用 head() 方法列出前五筆資料，能大致查看資料長相。

程式碼

```
string=['你','要發','大才']
for i in string:
    print(i)
```

執行結果

```
你
要發
大才
```

巢狀迴圈是很常使用的迴圈，專門用來解析多為度的資料，在網路爬蟲中特別常用。

程式碼

```
for i in range(0,2): #第一層
    for j in range(0,3): #第二層
        print('現在I是： '+ str(i) + '  J是： ' + str(j))
```

執行結果

```
現在I是： 0  J是： 0
現在I是： 0  J是： 1
現在I是： 0  J是： 2
現在I是： 1  J是： 0
現在I是： 1  J是： 1
現在I是： 1  J是： 2
```

3.2 Beautifulsoup

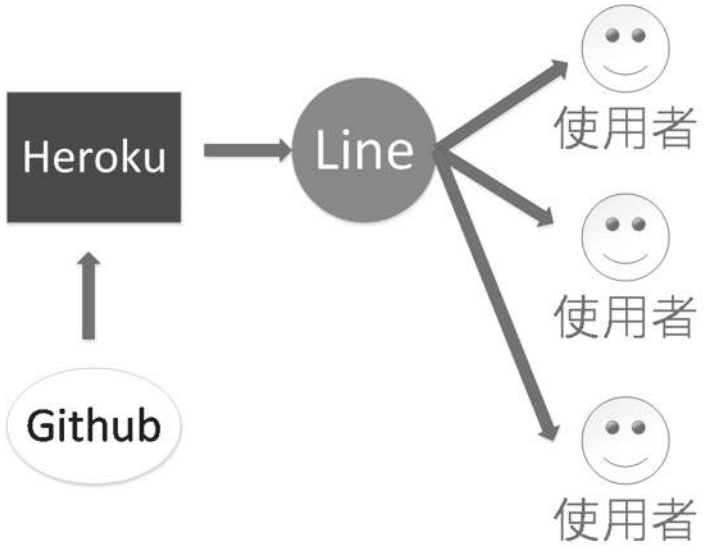
爬蟲的主要程式碼非常少，只需要三行就可以爬下來了，利用 BeautifulSoup 這個套件，就能將網頁中的 HTML 文字抓取下來，最費工的是要如何從裡頭篩選出想要的資訊。

本節教學要爬下來的文字如下圖框框所示，是該股票的收盤價格。

若對該數字點選右鍵選擇檢查，即可看到右方跑出 HTML 程式碼，且當滑鼠指到上方時，瀏覽器會自動幫你反白，讓你知道現在指的標籤位於程式碼的哪個位置，這對於網路爬蟲的工作來說，非常方便，可以迅速找到想要的資料位於什麼標籤裡頭，由圖片可知，我們需要的文字在 <b> 這個標籤裡頭，因此先試著直接爬下 <b> 標籤看看。

4.1 基礎架構介紹

整個 LINEBot 的建置流程如下，將我們寫好的程式碼先上傳到 Github，藉由 Github 連動到 Heroku 雲端服務，等於是在 Heroku 上架設屬於自己的主機，而後向 LINE 通訊軟體申請一個 API 服務，讓 Heroku 主機可以與 LINE 達成連接，最後由 LINE 做視覺化的方式，將想要推播的訊息，在手機上顯示。



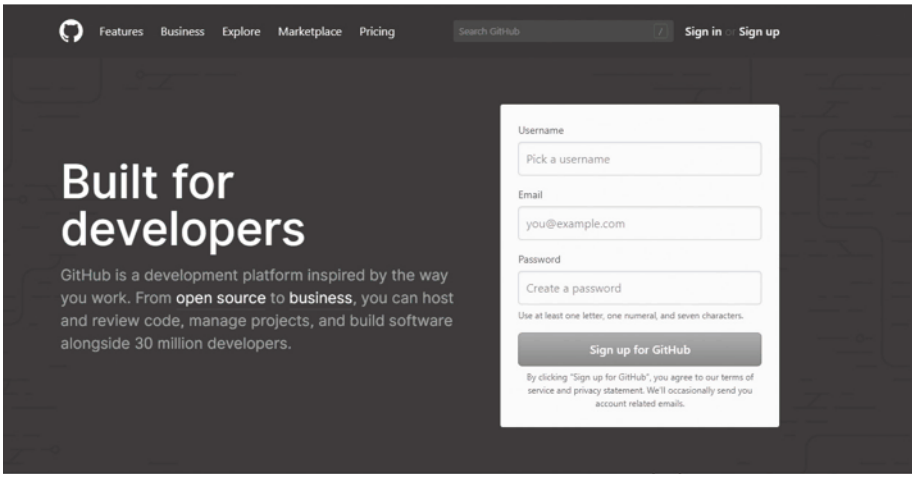
4.2 Github 教學

GitHub 是透過 Git 進行版本控制的軟體原始碼代管服務，簡單來說，我們可以把打好的程式碼放在上面，依照不同的專案做分類。如果你是程式開發團隊，每個人被分配到不同的任務，大家將最後的交稿上傳到這個平台，這個平台還有做「分支（Branch）」與「主幹（Master）」的功能，每位程式設計師都是一個分支，大家一起集合起來的檔案，就可以設定為主幹。當然，團隊很少一次就做出可以上市的產品，因此在產品上市之前，可能會產出過很多次主幹，Github 會幫你記錄每一次更新的版本，達到版本控管的功能，如果發現更新版有問題，還能夠溯源，找到之前的版本。

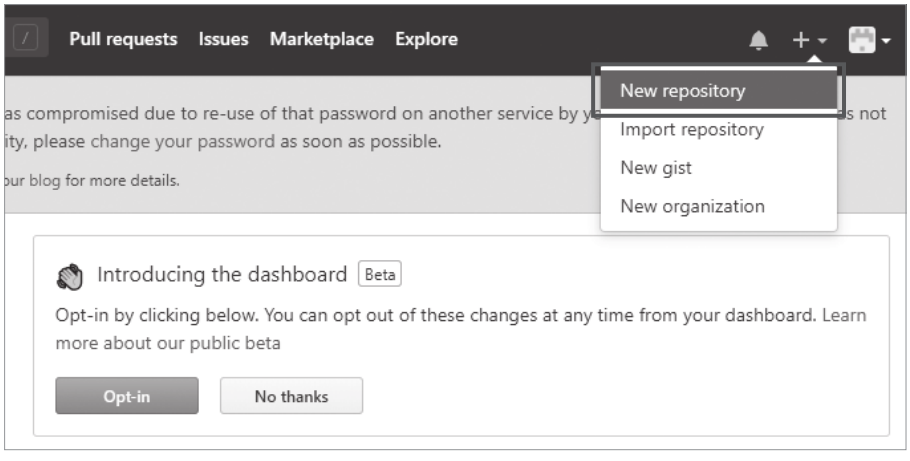
為了推廣開源，公開是免費，私人是要付費的（但在 2019 年起，私人免收費），也就是因為公開是免費的，讓這裡成為工程師的寶庫，許多解法上面都有，有各個頂尖的程式高手會將自己的程式碼作品公開在上面，可以拿來參考，甚至有不少公司聘用軟體工程師時，會查看應徵者的 Github 空間，作為選才的參考。

這樣的軟體服務平台不只一種，但 GitHub 是最多人使用的，因為有大量的雲端服務都與 GitHub 連動。例如我們接下來會使用到的雲端服務 Heroku，就只有跟 Github 合作。

首先到 GitHub 申請一個帳號後登入，網址：https://github.com/。



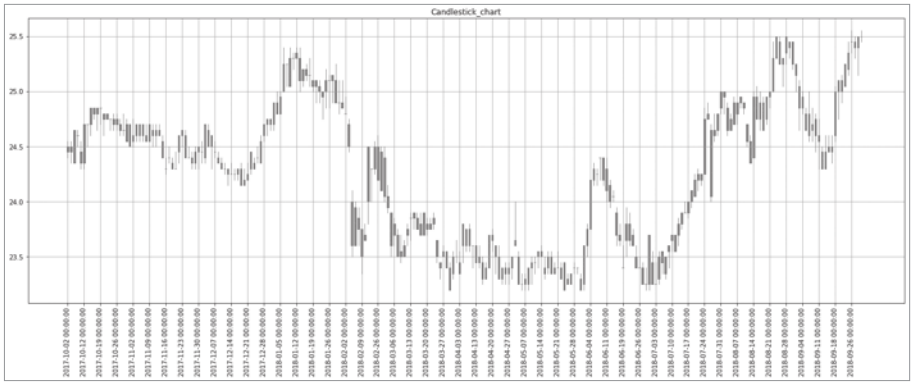
接下來，創建一個新的專案。



鍵入想要設定的專案名稱，並設定程式碼為「私有」。

```
# 股票K線圖
fig = plt.figure(figsize=(24, 8))
ax = fig.add_subplot(1, 1, 1)
ax.set_xticks(range(0, len(stock.index), 5))
ax.set_xticklabels(stock.index[::5])
plt.xticks(fontsize=10,rotation=90)
mpf.candlestick2_ochl(ax, stock['Open'], stock['Close'],
stock['High'], stock['Low'],
                      width=0.5, colorup='r',
colordown='green',
                      alpha=0.6)
plt.title("Candlestick_chart") # 標題設定
plt.grid()
plt.savefig('Candlestick_chart.png') #存檔
plt. close() # 殺掉記憶體中的圖片
```

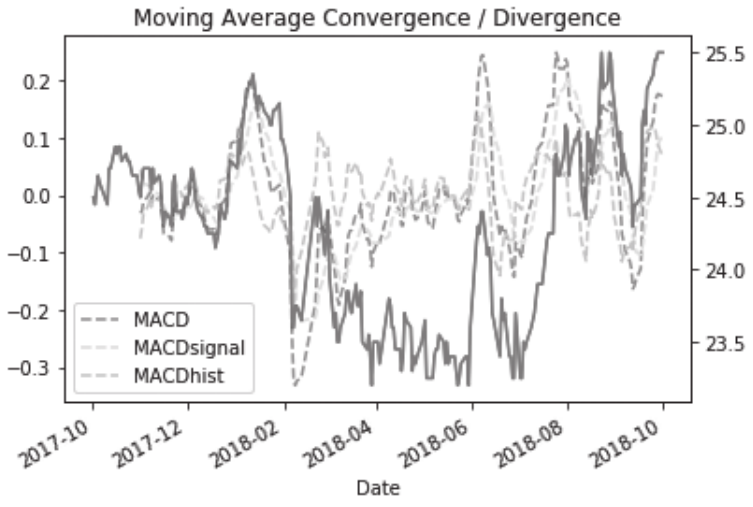
執行結果



移到下一頁

```
### 開始畫圖 ###
ret.plot(color=['#5599FF','#66FF66','#FFBB66'],
LINEstyle='dashed')
stock['Close'].plot(secondary_y=True,color='#FF0000')
plt.title("Moving Average Convergence / Divergence") # 標
題設定
plt.show()
```

執行結果

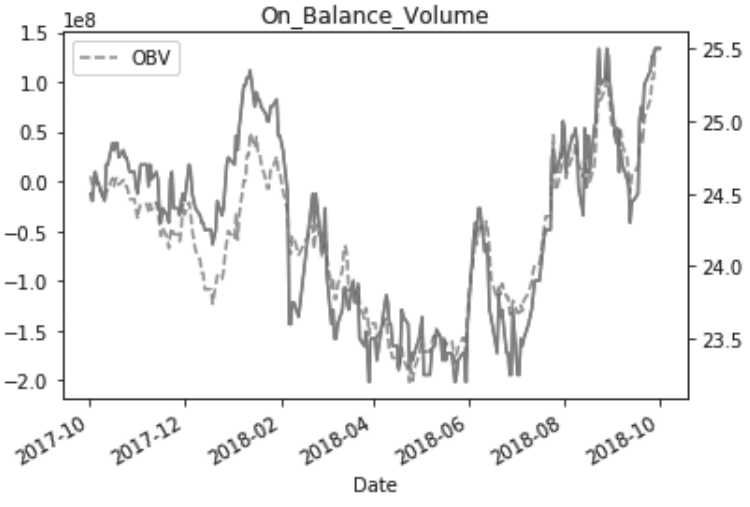


```
# 取得股票資料
stock = data.get_data_yahoo(userstock+'.TW', start, end)

# 股票OBV圖
ret = pd.DataFrame(talib.OBV(stock['Close'].values,
stock['Volume'].values.astype(float)), columns= ['OBV'])
ret = ret.set_index(stock['Close'].index.values)

### 開始畫圖 ###
ret.plot(color=['#5599FF'], LINEstyle='dashed')
stock['Close'].plot(secondary_y=True,color='#FF0000')
plt.title("On_Balance_Volume") # 標題設定
plt.show()
```

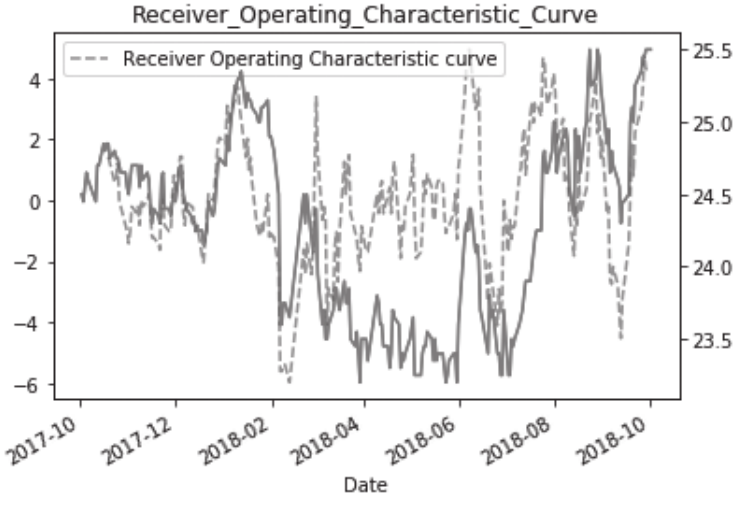
執行結果



```
ret = pd.DataFrame(talib.ROC(stock['Close'].values,
timeperiod=10), columns= ['Receiver Operating
Characteristic curve'])
ret = ret.set_index(stock['Close'].index.values)

### 開始畫圖 ###
ret.plot(color=['#5599FF'], LINEstyle='dashed')
stock['Close'].plot(secondary_y=True,color='#FF0000')
plt.title("Receiver_Operating_Characteristic_Curve") # 標
題設定
plt.show()
```

執行結果



技術面範例