

# PTT 八卦版爬蟲實作報告

資工四 B0743022 林聖博

## 實作作業-PTT 八卦版爬蟲

1. 設定每天早上凌晨1:00執行爬蟲程式(連續三天以上爬取)
2. 抓取**前一天**PTT上的**推噓文(蓋樓)**最多的**前3篇八卦版文章**
3. 抓取每篇文章中的相關資訊 (包括**作者,標題,時間,po文IP,內文**)
4. 欄位須加上**爬蟲爬取之時間**
5. 以**csv**方式儲存資料

繳交時間: 2021/11/28(日)之前

報告demo時間: 11/30

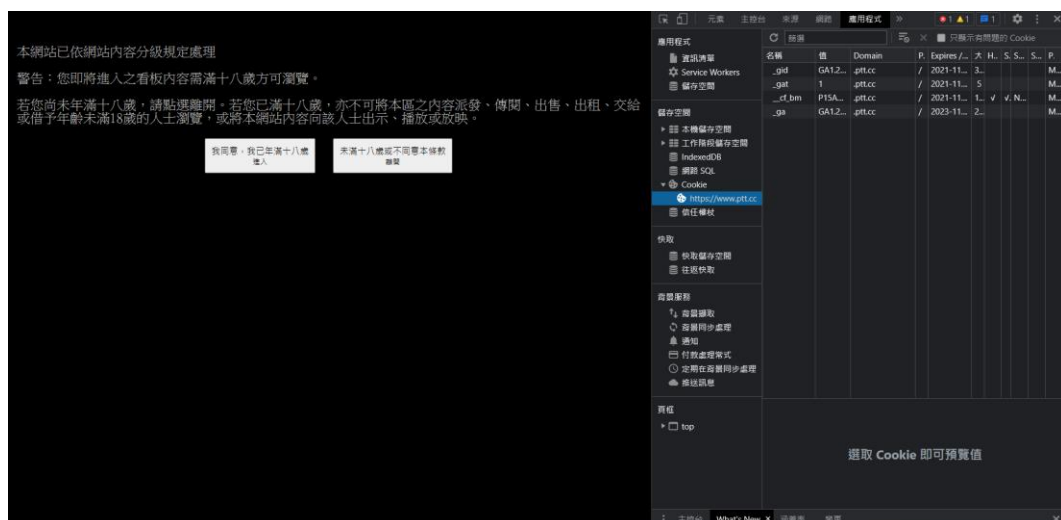
### 一、前置作業

- 引入必備套件：

```
1 # -*- coding: utf-8 -*-
2 import requests
3 import bs4
4 import re
5 from datetime import datetime, timezone, timedelta
6 import csv
```

- 「18 歲驗證」：

尚未驗證 18 歲時，利用 Chrome 檢查工具能發現 cookie 中無相關已驗證訊息。



當完成 18 歲驗證後，cookie 中將新增 over18 的鍵值對。



➤ `get_PTTweb_page(url)`：

建立「18 歲驗證」函式，利用 url 參數傳入欲爬蟲網址，並使用 get 方法進行網頁請求，請求同時帶入 cookie 值，即{'over18': '1'}，表示(模擬)已完成是否 18 歲的按鈕的驗證。

若成功請求，即 Http 狀態碼回傳 200，則回傳網頁請求結果，反之則否。

```
9 def get_PTTweb_page(url):
10     resp = requests.get(url=url, cookies={'over18': '1'}) # ptt18歲的認證
11     if resp.status_code != 200: # 回傳200代表正常
12         print('Invalid url:', resp.url)
13         return None
14     else:
15         return resp.text
```

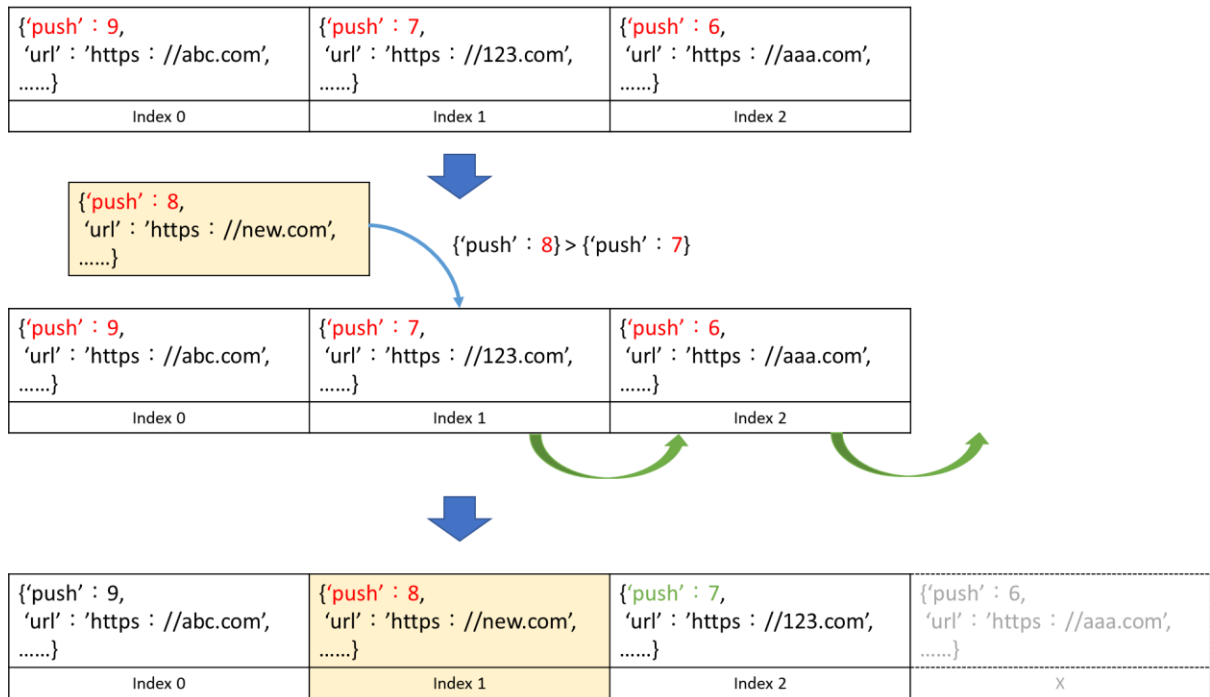
## 二、前三筆最多推噓文(蓋樓)文章處理方式

### ● 演算法說明：

為了節省空間的使用量，因此建立一項簡單的排序演算法。

其關鍵原理為使用已知大小陣列(本此因需要前三筆最多蓋樓，因此使用一長度為 3 的 list)，在每次爬取文章時計算該文章蓋樓數，並在這已知陣列上由左向右判斷是否大於陣列元素。若大於，則插入該文章的相關訊息，並陣列中由該處的所有右邊元素將向右移動一位。

如此，每次進行爬文章時，就可動態更新前三筆文章陣列，並保持該陣列元素皆為前三筆最多推噓文(蓋樓)文章資訊。此作法目的是避免，需要存取大量元素的陣列，最後才進行排序並在進行前三筆切片，若資料量大，排序的效率會變差。



● 函式說明：

➤ find\_top(one\_article, top)：

傳入目前爬取文章資訊(one\_article)，如蓋樓數，並可自訂前幾筆最多推文數參數(top)，目前使用前三筆。

※註：使用字典 dict 中的方法 get()，可避免當字典無該鍵時會出現錯誤訊息導致程式無法執行。

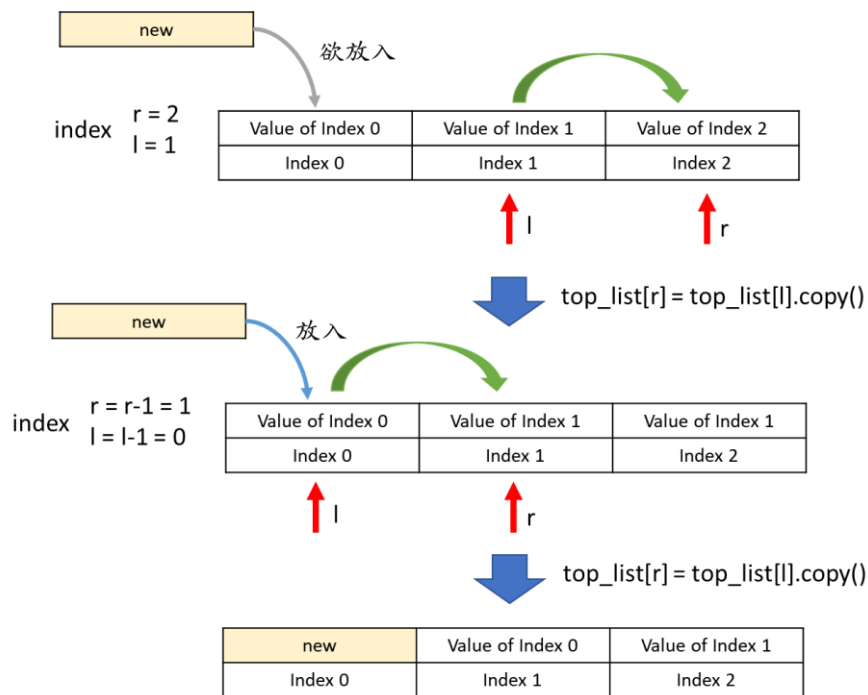
```
28 def find_top(one_article, top):
29     global top_list
30     for i in range(top):
31         if top_list[i].get('push') < one_article['push']: # 用get 能避免沒有key時 不會都出錯誤
32             push_right(i, top-2, top-1)
33             top_list[i] = one_article
34             break
```

➤ push\_right(i, l, r)：

使用遞迴方式進行陣列元素移動，由最右邊兩元素開始移動，並由右向左運作。

※註：因為陣列內存放字典物件，因此在指派時，要使用 copy()進行複製，否則會因為物件指派是傳址，導致當物件變化時，陣列元素也將跟著變化。

```
20 def push_right(i, l, r):
21     global top_list
22     if r == i:
23         return
24     top_list[r] = top_list[l].copy()
25     push_right(i, l-1, r-1)
```



➤ top\_list :

此變數為前三筆最多蓋樓陣列，存放文章相關資訊，使用字典物件作為元素。下圖為設定初始元素。

```
60 top_list = [{'push': 0}]*top
```

### 三、 相關基本變數設定

```
38 # timestamp 取得現在時間、指定時區
39 tz = timezone(timedelta(hours=+8)) # 設定為 +8 時區
40 dt = datetime.now(tz).replace(microsecond=0)
41
42 top = 3
43 article_date = ''
44 stop = False
45 first_crawl = True
46 article_count = 0
47 page_count = 0
48 |
49 PTT_ROOT_URL = 'https://www.ptt.cc'
50 payload = {
51     'timestamp': '',
52     'push': 0,
53     'author': '',
54     'title': '',
55     'href': '',
56     'date': '',
57     'ip': '',
58     'contents': ''
59 }
60 top_list = [{'push': 0}]*top
61 compile_ip = re.compile("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}")
```

## 四、爬蟲主程式

對 PTT 八卦版 URL 進行 GET 請求，此處用到上述第一點的函式進行網頁請求，並獲取請求後的回應(response)。接著針對 response 利用 BeautifulSoup 解析 HTML 的 [DOM](#)。

```
69 response = get_PTTweb_page('https://www.ptt.cc/bbs/Gossiping/index.html')
70 soup = bs4.BeautifulSoup(response, "html.parser")
```

而一個頁面(看板)下有許多文章，因此使用 for 迴圈逐步爬取當前頁面中所有的文。因 PTT 每一篇文章都利用一 div 包起來，因此找尋該 div 屬於的 class 名稱，即 r-ent，並獲取該文章連結，藉著進入文章獲取每篇文章內的相關資訊，如：蓋樓數、作者、標題、連結、發文時間、IP、內容。

```
75 while(True):
76     for article_div in soup.find_all('div', 'r-ent'):
77         push_count = 0
78         article = article_div.find('div', 'title')
79         article_date = article_div.find('div', 'date').text
```

爬取文章時將進行日期判斷，判斷是否是前一天文章，若否，則忽略不進行爬取。而判斷非前一天文章，也將會巢狀判斷是否為前前天文章並同時符合非第一次頁面爬取(因為第一次頁面 index.html 可能含有其他天文章或公告等)，若兩項皆符合條件，則表示當前頁面是爬蟲結束點，當進行完此頁爬蟲後，將停止爬取。

```
81         # 判斷前一天
82         if not(article_date == (dt.timedelta(days=1)).strftime("%m/%d")):
83             if (article_date == (dt.timedelta(days=2)).strftime("%m/%d")) and not(first_crawl):
84                 stop = True
85                 continue
```

div.r-ent 1151 × 61.98		
2	【問卦】 公投換成哪四大議題比較有效？ linbasohigh	11/27 ...
爆	【協尋】 代轉貼：萬華跪求肇事車禍現場車記錄器 Gossiping	11/18 ...
爆	【協尋】 11/20求行車記錄器台中臺灣大道 petersnape	11/20 ...
	【公告】 八卦板板規(2021.11.21) ubcs	! 11/21 ...
本網站已依台灣網站內容分級規定處理。此區域為限制級，未滿十八歲者不得瀏覽。		

開始進入每一篇文章內，針對所需資訊進行擷取。

```
87         try:
88             response_in = get_PTTweb_page(PTT_ROOT_URL+article.a['href'])
89             soup_in = bs4.BeautifulSoup(response_in, "html.parser")
90
91             header = soup_in.find_all('span', 'article-meta-value')
92             author = header[0].text
93             title = header[2].text
94             date = header[3].text
95
96             main_container = soup_in.find(id='main-container')
97             all_text = main_container.text
98             ip = re.search(compile_ip, all_text, flags=0).group()
99
100            pre_text = all_text.split('--')[0]
101            texts = pre_text.split('\n')
102            contents = texts[2:]
103            content = '\n'.join(contents)
104
105            # timestamp
106            dt = datetime.now(tz).replace(microsecond=0)
107            timestamp = dt.strftime("%Y-%m-%d %H:%M:%S")
```

進入文章方式使用根 URL 加上文章 URL 位置(根據 div title 的資超連結)並同樣使用 get\_PTTweb\_page() 進入該文章。

```
▼<div class="title"> == $0
  <a href="/bbs/Gossiping/M.1638002806.A.122.html">[問卦] 公投換成哪四大議題比較有效?</a>
</div>
```

接著，找出文章 header 資訊，其 HTML tag 為{'span', 'article-meta-value'}。

```
header = soup_in.find_all('span', 'article-meta-value')
author = header[0].text
title = header[2].text
date = header[3].text
```



```

▼<div id="main-content" class="bbs-screen bbs-content">
  ▼<div class="article-metaline">
    <span class="article-meta-tag">作者</span>
    <span class="article-meta-value">nanachi (上善若水)</span> == $0
  </div>
  ▼<div class="article-metaline-right">
    <span class="article-meta-tag">看板</span>
    <span class="article-meta-value">Gossiping</span>
  </div>
  ▼<div class="article-metaline">
    <span class="article-meta-tag">標題</span>
    <span class="article-meta-value">[問卦] 為什麼panpiano沒有被搜出本人身分?</span>
  </div>
  ▼<div class="article-metaline">
    <span class="article-meta-tag">時間</span>
    <span class="article-meta-value">Sat Nov 27 16:47:27 2021</span>
  </div>

```

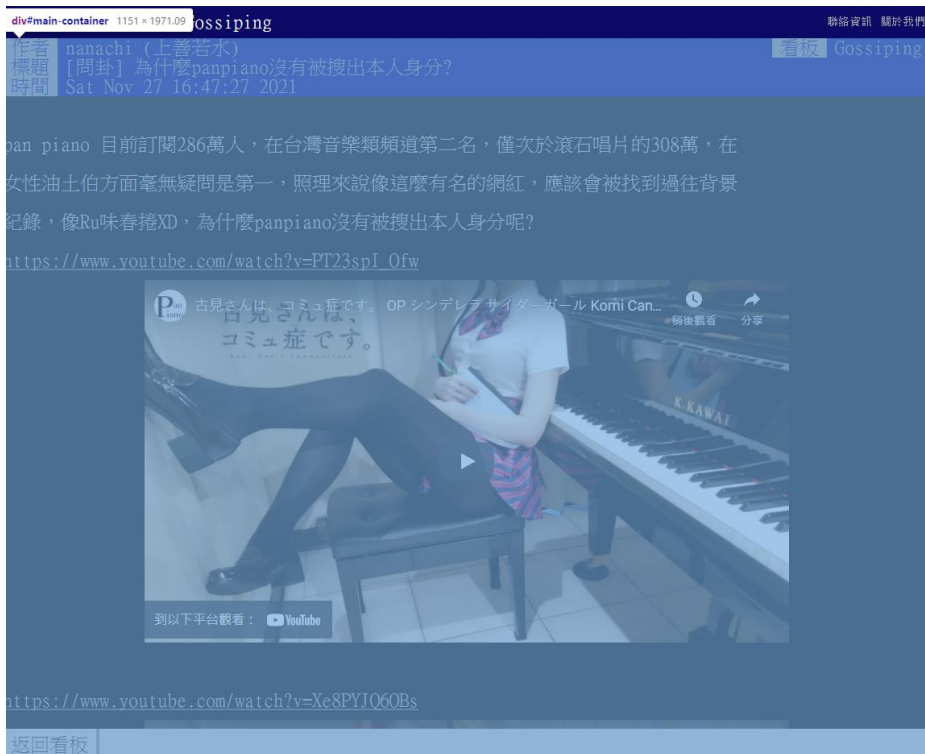
而後，對內容進行擷取，利用 PTT 文章版面特性，找在內容 div id='main-container' 中的 text 找出 "--" 符號進行分隔。利用 split('--') 方法切個文字形成陣列，此元素 index 0 為文章內容，index 1 為來源 ip 位置資訊。

```

main_container = soup_in.find(id='main-container')
all_text = main_container.text

pre_text = all_text.split('--')[0]
texts = pre_text.split('\n')
contents = texts[2:]
content = '\n'.join(contents)

```





```

" pan piano 目前訂閱286萬人，在台灣音樂類頻道第二名，僅次於滾石唱片的308萬，在 女性油
土伯方面毫無疑問是第一，照理來說像這麼有名的網紅，應該會被找到過往背景 紀錄，像Ru味春捲
XD，為什麼panpiano沒有被搜出本人身分呢？ "
<a href="https://www.youtube.com/watch?v=PT23spI_0fw" target="_blank" rel="nore
ferrer noopener nofollow">https://www.youtube.com/watch?v=PT23spI_0fw</a>
▶<div class="richcontent">...</div>
<a href="https://www.youtube.com/watch?v=Xe8PYJQ60Bs" target="_blank" rel="nore
ferrer noopener nofollow">https://www.youtube.com/watch?v=Xe8PYJQ60Bs</a>
▶<div class="richcontent">...</div>
" -- "
<span class="f2">※ 發信站：批踢踢實業坊(ptt.cc)，來自：223.140.143.211 (臺灣)

```

而為了獲取 ip 位置，使用正規表達式(Regular Expression, regex)來進行搜尋，因已獲取 main-container 內所有文字(all\_text)，利用 [re.search\(\)](#) 搜尋符合 ip address 的格式字串，並獲取該發文者 ip 位置。

※註：相關參考網站

1. <https://www.runoob.com/python/python-reg-expressions.html>
2. <https://ithelp.ithome.com.tw/articles/10232174>

```
compile_ip = re.compile("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}")
```

```
ip = re.search(compile_ip, all_text, flags=0).group()
```

紀錄爬取文章時的時間點 timestamp。

獲得時間方法可以直接使用 datetime 函式結合 format string 的方法使用

↳ %Y(大寫是選擇世紀年份小寫則是[00,99]) 對應年。

↳ %m、%d 對應分別為月、日。

↳ %H、%M、%S 分別為小時(24)、分鐘、秒數。

```

# timestamp
dt = datetime.now(tz).replace(microsecond=0)
timestamp = dt.strftime("%Y-%m-%d %H:%M:%S")

```

因看板的文章旁數字為推數-噓數之結果，必非完整蓋樓數，引此進入文章時，將會計算該文章內的蓋樓數，獲取文章內 HTML tag 的所有{'span', 'push-tag'}並進行該文章蓋樓數計數。

※缺點：會花費大量時間進行 for 迴圈遍歷。

欲解決方法：利用看板頁面中具有「爆」的文章進行文章爬蟲，其優點 1. 可避免冗餘文章的爬取(不用每一篇文章爬取)、2. 標示為「爆」的文章為較多推噓文的文章。



```

110         # find push
111         for push_tag in soup_in.find_all('span', 'push-tag'):
112             # print(push_tag)
113             if ('推' in push_tag.text) or ('噓' in push_tag.text) or ('→' in push_tag.text):
114                 push_count += 1

```

將文章處理後的所需資訊結果存入字典中。

```

116         payload['timestamp'] = timestamp
117         payload['push'] = push_count
118         payload['author'] = author
119         payload['title'] = title
120         payload['href'] = PTT_ROOT_URL+article.a['href']
121         payload['date'] = date
122         payload['ip'] = ip
123         payload['contents'] = content

```

利用第二點所敘述的前三筆蓋樓數排序函式 find\_top(one\_article, top)進行比對，判斷是否為前三筆的文章。而後，完成一篇文章的爬取，進行計數。

```

125         find_top(payload.copy(), top)
126         article_count += 1

```

值得注意的是，因為文章可能被刪除，因此當爬蟲程式要進獲取文章連結(div title 無超連結<a href="..." />)時會有錯誤，需要利用 try except 進行例外處理。

```

128         except Exception as e:
129             # print(e)
130             print('\033[93m' + article.text.strip() + '\033[0m')
131             pass

```



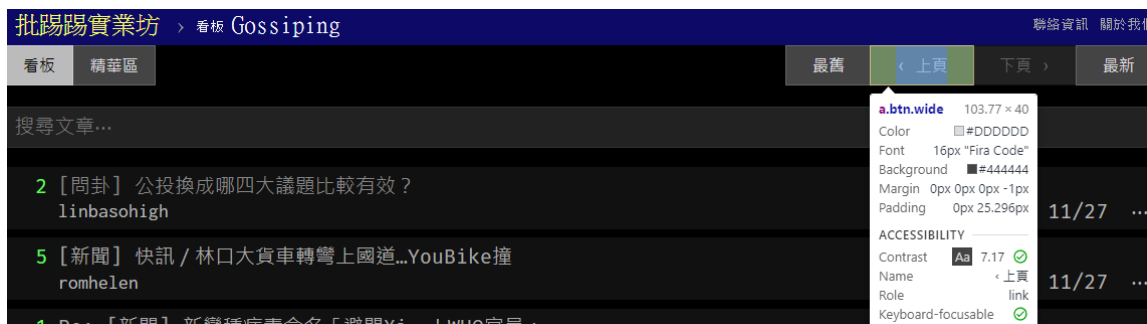
```

▼<div class="r-ent"> == $0
  <div class="nrec"></div>
  <div class="title"> (本文已被刪除) [victorly] </div>
  ▶<div class="meta">...</div>
</div>

```

## 五、換頁處理

為了動態處理換頁動作，可藉由看板頁面中上頁的 button 進行換頁。



完整爬取一看板頁的文章後，找尋該頁中換頁的 HTML tag {'a', 'btn wide'}，並判斷為「上頁」的按鈕時，獲取該上一頁的看板連結，接著重複上述爬蟲程式(最外層利用 while 迴圈)，直到頁面有前前天的文章為止。

```
137     #換頁處理
138     for link in soup.find_all('a', 'btn wide'): # 用來抓取上一頁
139         if link.text == '< 上頁':
140             response = get_PTTweb_page(PTT_ROOT_URL+link['href'])
141             soup = bs4.BeautifulSoup(response, "html.parser")
142             break
```

## 六、後續處理(Bonus)

為了解爬蟲程式處理時間，因此利用時間戳記對程式開始時到結束時間中間處理時間間隔進行計算。

```
72     # process time
73     start = datetime.now()
74
75 > while(True): ...
144
145     end = datetime.now()
```

最後將相關結果 print 出來，可用於後續顯示爬蟲處理資訊應用。

```
147     datetime_dt = datetime.today() # 獲得當地時間
148     dt = datetime_dt.date() # 最小日期顯示到日
149     new_date = dt - timedelta(days=1)
150
151     print("批踢踢實業坊>看板 Gossiping 爬蟲結果")
152     print("文章日期: {}年 {}月 {}日".format(new_date.year, new_date.month, new_date.day))
153     print("文章數:", article_count, "篇")
154     print("完成時間:", datetime.now().strftime("%Y/%m/%d %H:%M:%S"))
155     print("執行時間:", end - start)
156     print('Done ... ')
157     print('*****Result*****')
```

## 七、 結果存為 CSV 檔案

利用 with open() as 進行檔案讀寫管理，將 close()方法關閉檔案方法包進 with open() as 裡，能節省程式撰寫行數與提升程式可讀性。

```
160 # 寫入csv
161 try:
162     with open('result.csv', 'a+', newline='') as csvfile:
163         writer = csv.writer(csvfile)
164         for i in range(top):
165             writer.writerow([top_list[i]['timestamp'], top_list[i]['author'], top_list[i]
166                             ['title'], top_list[i]['push'], top_list[i]['date'], top_list[i]['ip'], top_list[i]['contents']])
167 except:
168     with open('result.csv', 'w+', newline='') as csvfile:
169         writer = csv.writer(csvfile)
170         writer.writerow(['Timestamp', 'Author', 'Title', 'Push',
171                         'Date', 'IP_address', 'Contents'])
172
173     for i in range(top):
174         writer.writerow([top_list[i]['timestamp'], top_list[i]['author'], top_list[i]
175                         ['title'], top_list[i]['push'], top_list[i]['date'], top_list[i]['ip'], top_list[i]['contents']])
```

若第一次執行此爬蟲程式，將會以寫入檔案模式，增加一個新的 CSV 檔，並將第一次 PTT 文章爬取結果寫入該檔內，同時，加入 CSV header(第一行)。

```
except:
    with open('result.csv', 'w+', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['Timestamp', 'Author', 'Title', 'Push',
                        'Date', 'IP_address', 'Contents'])

    for i in range(top):
        writer.writerow([top_list[i]['timestamp'], top_list[i]['author'], top_list[i]
                        ['title'], top_list[i]['push'], top_list[i]['date'], top_list[i]['ip'], top_list[i]['contents']])
```

若為第二次以上執行此爬蟲程式，則會先讀取以存在的 CSV 檔案，並利用檔案追加模式，with open('result.csv', 'a+', newline='') as csvfile:，進行新資料的加入。

```
# 寫入csv
try:
    with open('result.csv', 'a+', newline='') as csvfile:
        writer = csv.writer(csvfile)
        for i in range(top):
            writer.writerow([top_list[i]['timestamp'], top_list[i]['author'], top_list[i]
                            ['title'], top_list[i]['push'], top_list[i]['date'], top_list[i]['ip'], top_list[i]['contents']])
```

最終結果如下圖所示：

G2

f<sub>x</sub>Σ▾=

我幫你講清楚啦  
兇嫌 張敦量  
另外兩人叫陳勁豪和李韋霖  
張敦量有多項傷害前科  
養出暴力兇嫌的失格父親名字 張朝棟

	A	B	C	D	E	F	G
1	Timestamp	Author	Title	Push Date	IP Address	Contents	
2						另外兩人叫陳勁豪和李韋霖 張敦量有多項傷害前科 養出暴力兇嫌的失格父親名字 張朝棟 敦量食品名稱 百佳食品 在彰化溪湖 兇嫌是百佳食品小開  為什麼還新聞 什麼張姓是二代 什麼叫某知名食品公司工廠 買大記者你沒釋寫出來就不要幹記者噁gan  許多朋友擔心我會被罰錢 臺北破嗎呀我也買得起 想我寧願不買，這錢拿去被罰 就這這為了臺灣良善人民！ 你相信臺中警察、司法嗎？ 只能靠我們自己  感謝熱烈迴響，加碼兇嫌張敦量玉照 <a href="https://i.imgur.com/IGVCb39.jpg">https://i.imgur.com/IGVCb39.jpg</a>  幫高調咁幹  ※ 引述《Movice (Jake Peavy)》之銘言： ：備註請放最後面 違者新聞文章刪除 1. 媒體來源： ※ 例如蘋果日報、自由時報 (請參考版報下方的核准媒體名單) ETtoday新聞雲 2. 記者署名： ※ 若新聞沒有記者名字或編輯名字，請勿張貼，否則會被水權14天 ※ 外電至少要有來源或編輯 如：法新社 記者葉國忠／綜合報導 3. 完整新聞標題： ※ 標題沒有完整寫出來 ：若這新聞標題的條件都符合然後去山莊買部手機吧 ：我說在座的各位，可憐阿  你在公三小阿？ 這種人隨處可見，到處都是，可不是台中獨有特產，我住新北也是一天到晚看到。  這些8+9的生計不是討債就是開地下錢莊，不然就是去打詐騙電話。 平常在外面就是遊手好閑到處騷擾別人， 目光掃到就瞪你看三小， 去任何商家只會大吼大叫， 以前我在超商打工做大夜班就遇到超多這種低能， 從一開始的害怕到之後只覺得這些人很煩。	
	2021-11-12 01:05:1	LeeANan (就值得了愛，就值得)	Re: [新聞]快訊／馮莎拉希3惡徒「爆打男大生」警押 富商父09:3	1473Thu Nov 11 11:55:16 2021	210.61.193.1		

## 八、排程程序

為了使此爬蟲程式每天早上凌晨 1:00 執行，因此本程式執行於 Linux 上，並使用 crontab 工作排程工具，設定讓系統定時自動執行指定的指令或程式。

```
# _____ 分鐘 (0 - 59)
# | _____ 小時 (0 - 23)
# | | _____ 日 (1 - 31)
# | | | _____ 月 (1 - 12)
# | | | | _____ 星期幾 (0 - 7, 0 是週日, 6 是週六, 7 也是週日)
# | | | | |
# * * * * * /path/to/command
```

### 1. 制定 crontab 設定檔

- crontab -e # 編輯 crontab 內容
- 0 1 \* \* \* /usr/bin/python3 /home/ubuntu/ptt\_crawler.py > crawler\_result.log &&  
mpack -s "PTT\_Crawler Result" -d /home/ubuntu/crawler\_result.log  
/home/ubuntu/result.csv b3717700@gmail.com

➤ 設定時間說明(<https://crontab.guru/>)

為了於每日凌晨 1:00 執行，因此設定為=> 0 1 \* \* \*

*	*	*	*	*
分鐘	小時	日期	月份	星期
minute	hour	dayOfMonth	month	dayOfWeek
0 to 59	0 to 23	1 to 31	1 to 12	0 to 7 0 and 7 means Sunday 1 means Monday 2 means Tuesday

➤ 指令說明

🔗 `/usr/bin/python3 /home/ubuntu/ptt_crawler.py > crawler_result.log && mpack -s "PTT_Crawler Result" -d /home/ubuntu/crawler_result.log /home/ubuntu/result.csv b3717700@gmail.com`

- i. `/usr/bin/python3 /home/ubuntu/ptt_crawler.py > crawler_result.log`  
 利用 > 指令符號，將程式的輸出結果導向寫入到指定檔案中。此處將爬蟲結果，如上述第六點的輸出結果寫入 crawler\_result.log 中。

- ii. **&&**：表示前一指令執行成功後，才繼續執行後一指令。此處必須等待爬蟲完成後才可進行下一步指令，否則尚未完成爬蟲就寄送郵件內容會不完整與錯誤。

- iii. **mpack -s "PTT\_Crawler Result" -d /home/ubuntu/crawler\_result.log /home/ubuntu/result.csv [b3717700@gmail.com](mailto:b3717700@gmail.com)**

為了寄送爬蟲結果到指定的信箱內，因此使用 mpack 工具進行附件的郵件寄送。Mpack 是基於 SSMTTP(簡單郵件傳輸協定) 傳輸電子郵件的標準進行郵件發送，因此需先安裝並設定 Linux 上的 SSMTTP。

```

1  # 接收系統郵件的 Email
2  root=b3717700@gmail.com
3
4  # 使用 GMail 的 MTA 送信
5  mailhub=smtp.gmail.com:587
6
7  # 設定 hostname
8  hostname=raspberrypi
9
10 # 允許使用者設定 Email 的 From 欄位
11 FromLineOverride=YES
12
13 # Google 帳號與密碼
14 AuthUser=b3717700@gmail.com
15 AuthPass=PASSWORD
16
17 # 啟用安全加密連線
18 UseSTARTTLS=YES
19 UseTLS=YES
20
21 # 輸出除錯資訊
22 Debug=YES

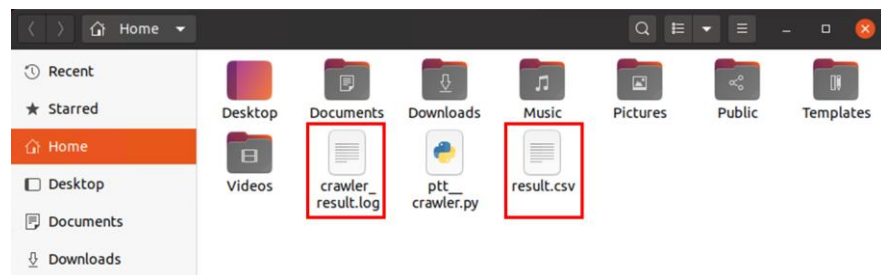
```

接著，利用 Mpack 針對 [MIME](#)(多用途網際網路郵件擴展)檔案，即表示文件、檔案或各式位元組的標準，進行編碼，並將編碼後的郵件檔案被發送至收件人。

- 參數說明

- -s：郵件主旨(Subject)
- -d：郵件內文，在此傳入 crawler\_result.log 的內文
- 檔案(檔案位置)
- 收件人信箱

```
mpack -s "Subject" -d content file_path mail_addr
```



## 2. 啟動 crontab 排程服務

- service cron start

## 3. 查看 crontab 服務的狀態

- service cron status



```

ubuntu@ubuntu2004:~$ sudo service cron status
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-11-23 20:38:54 CST; 3 days ago
     Docs: man:cron(8)
    Main PID: 2816 (cron)
      Tasks: 1 (limit: 2299)
     Memory: 4.7M
    CGroup: /system.slice/cron.service
            └─2816 /usr/sbin/cron -f

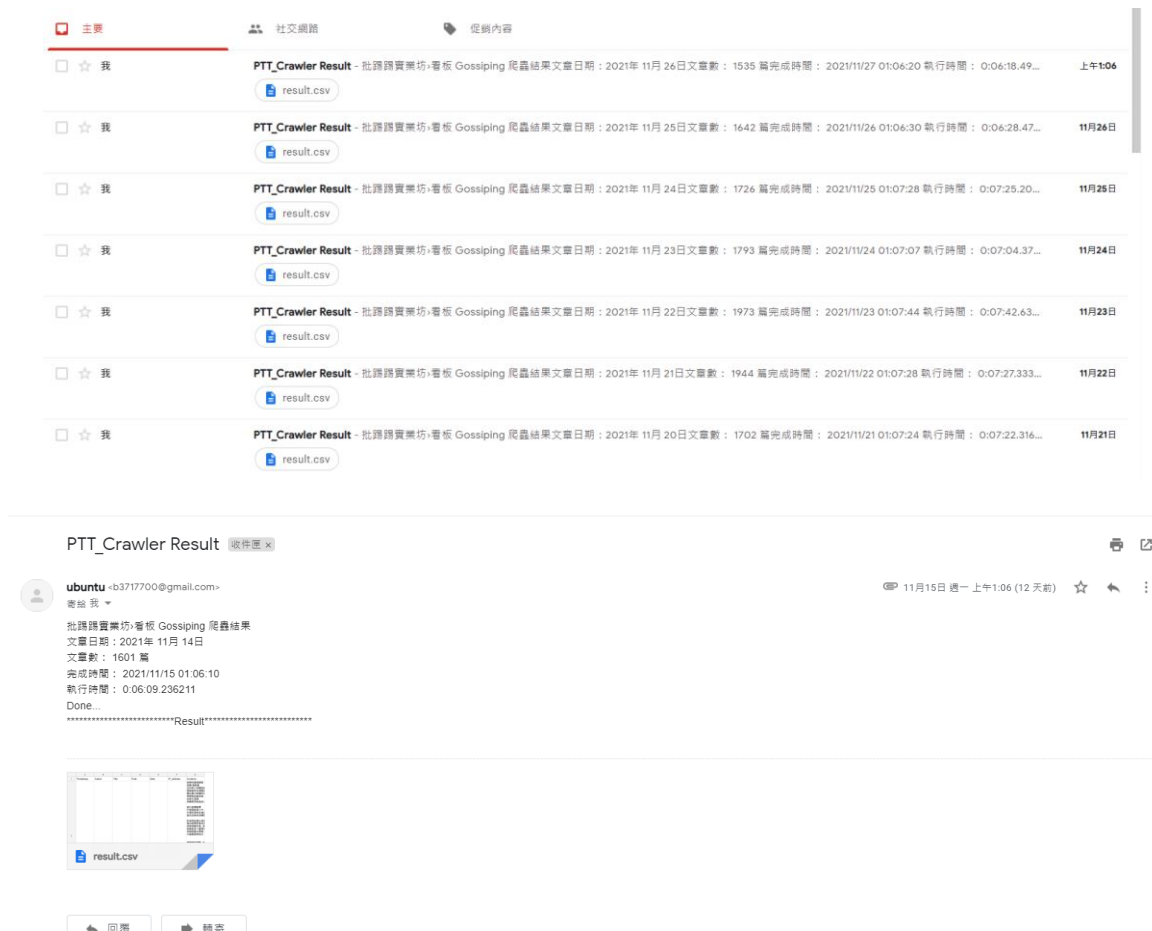
Nov 27 18:17:01 ubuntu2004 CRON[26442]: pam_unix(cron:session): session opened for user root by (uid=0)
Nov 27 18:17:01 ubuntu2004 CRON[26443]: (root) CMD ( cd / && run-parts --report /etc/cron.hourly)
Nov 27 18:17:01 ubuntu2004 CRON[26442]: pam_unix(cron:session): session closed for user root
Nov 27 18:30:01 ubuntu2004 CRON[26466]: pam_unix(cron:session): session opened for user root by (uid=0)
Nov 27 18:30:01 ubuntu2004 CRON[26466]: pam_unix(cron:session): session closed for user root
Nov 27 19:17:01 ubuntu2004 CRON[26755]: pam_unix(cron:session): session opened for user root by (uid=0)
Nov 27 19:17:01 ubuntu2004 CRON[26755]: pam_unix(cron:session): session closed for user root
Nov 27 19:30:01 ubuntu2004 CRON[26826]: pam_unix(cron:session): session opened for user root by (uid=0)
Nov 27 19:30:01 ubuntu2004 CRON[26827]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -d /run/systemd/system
Nov 27 19:30:01 ubuntu2004 CRON[26826]: pam_unix(cron:session): session closed for user root

```

#### 4. 最終郵件寄送結果

每日凌晨 1:00 左右皆會收到爬蟲結果 mail 通知。

值得注意，因為使用 GMail 的 MTA(郵件傳送代理人, Mail Transfer Agent) 送信，所以可發現寄件人為 gmail.com



PTT\_Crawler Result 收件匣 x

ubuntu <b3717700@gmail.com>  
寄給 我

批發路實業坊·看板 Gossiping 爬蟲結果  
文章日期: 2021年 11月 14日  
文章數: 1601 篇  
完成時間: 2021/11/15 01:06:10  
執行時間: 0:06:09.236211  
Done...

\*\*\*\*\*Result\*\*\*\*\*

result.csv

PTT\_Crawler Result

寄件者: ubuntu <b3717700@gmail.com>  
收件者: b3717700@gmail.com  
日期: 2021年11月15日 上午1:06  
主旨: PTT\_Crawler Result  
寄件人: gmail.com