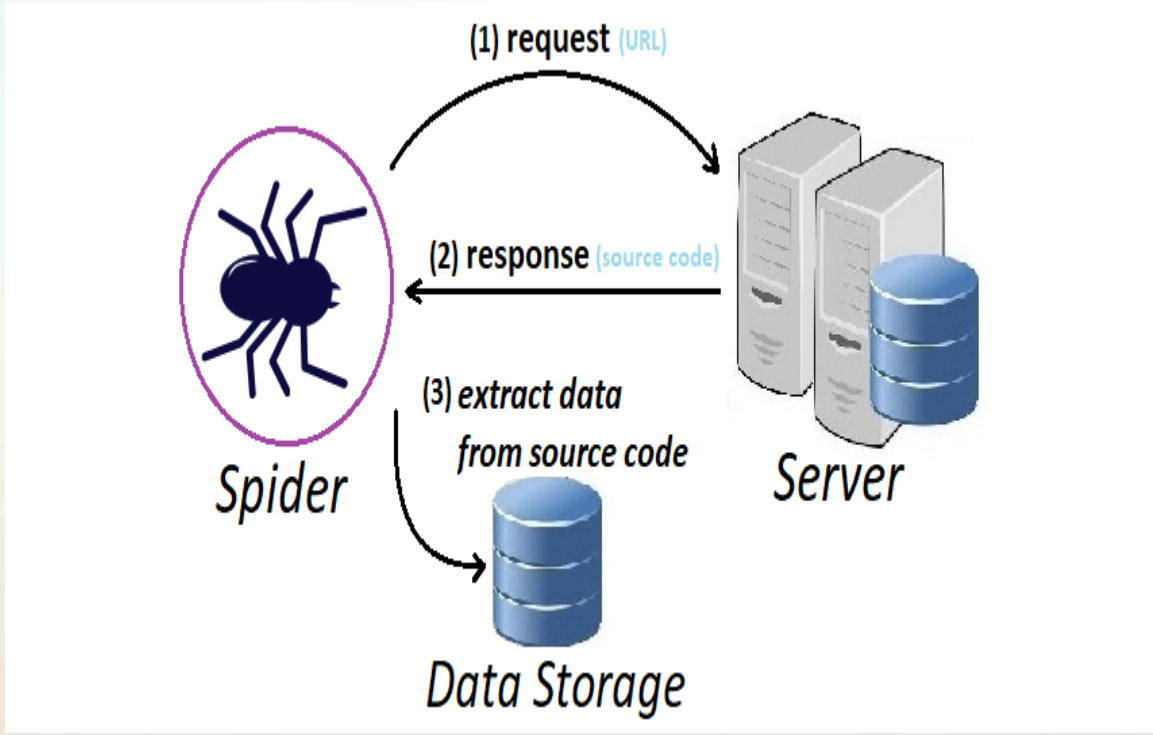
Python爬蟲程式與資料視覺化

講者:許智凱(Jalen)

- Spider原理
- 套件介紹
- Web Scraper簡介(傻瓜級的工具)
- 從案例中學習
- 各種Spider的爬法
- 視覺化運用
- 補充

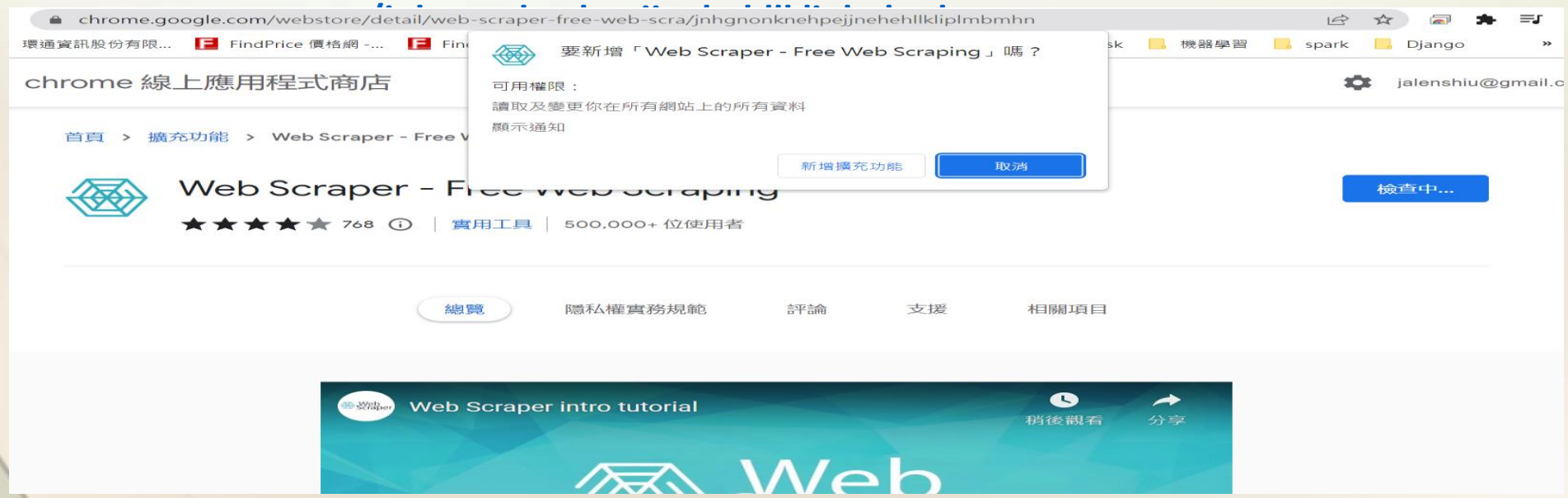
爬蟲原理


Client
reply
request
Server


(1) request (URL)
(2) response (source code)
(3) extract data
from source code
Spider
Server
Data Storage

爬蟲工具及套件

- 爬蟲工具(Web Scraper傻瓜級的工具)
  - Web Scraper介紹與原理： Web Scraper是一個網頁抓取工具，不需要複雜的安裝配置，是以Chrome 外掛的形式執行在Chrome瀏覽器上。原理:自己定義一些抓取規則爬取目標資料
  - **1. 安裝Web Scraper**
    - https://chrome.google.com/webstore/detail/web-scraper-free-web-



Web Scraper 簡介

- Web Scraper分為chrome插件和雲服務兩種，雲服務是收費的，chrome插件是免費的
- Web Scraper插件，可以讓你以“所見即所得”的方式挑選要提取的網頁數據，形成模版，以後可以隨時執行該模版，並且執行結果可以匯出成Csv格式檔案。

優點:

- 抓取需要登錄的數據資料較方便，因為這個插件是運行在瀏覽器上的
- 只要抓取頻率慢一點，被網站反制爬蟲的機率較小，也因為是瀏覽器的原因，這就像是真實的用戶訪問一樣。
- 學習成本低
- 適合一次性/短期/非爬蟲專業選手爬資料的需求。

缺點:

- 有驗證碼識別網站抓取效率較低
- 相對於爬蟲程序大量級的數據抓取用Web Scrpaer不適合，慢慢抓大幾千網頁還是可以。插件本身是不支持配置定時任務的，雲服務提供了這種功能，不過是收費的，到是可以嘗試使用Python驅動谷歌來進而來操作。

spider套件

套件	import	說明
requests	requests	各種HTTP 請求
fake_useragent	fake-useragent	機產生 User-Agent 字串
BeautifulSoup	bs4	解析HTML結構
datetime	**datetime**	**日期時間相關**
urllib.parse	urllib	解析URL
logging	**logging**	記錄日誌文件

spider套件

套件	import	說明
json	json	json文件
OS	os	OS系統操作(資料夾或路徑,檔案屬性)
time	time	時間相關
webdriver	**selenium**	自動開啟瀏覽器爬蟲利器
threading	threading	執行序
lxml	etree	Xml文件

從實例中爬蟲所必要準備知識

- 利用webdriver取得目標網站的cookies
- 取得隨機userangent
- 分析目標網站的結構(原始碼, **Application Programming Interface**)
- 分析目標網站headders標頭內容
- 利用requests嘗試訪問目標網站取得原始碼內容
- 截斷字串法則(斷頭斷尾取內容)def xxx()
- 善用replace()

使用requests套件介紹

- 安裝套件 (pip install requests 、從開發工具安裝install)

1. requests.post： HTTP POST 傳遞參數 ,返回一個 requests.Reponse 型別的物件
2. requests. put 類似 requests.post

```
r = requests.post('http://httpbin.org/post', data = {'key':'value' })
```

3. requests.get ： HTTP GET參數 ,返回一個 requests.Reponse 型別的物件

```
r = requests.get('https://api.github.com/events' )
```

4. 傳遞 URLs 參數加入cookies

```
payload = {'key1': 'value1', 'key2': 'value2' }
r = requests.get('http://httpbin.org/get', params=payload)
 print(r.url) #查看傳送的 URL
```

使用requests套件回應資料分析

- print(r.text) *#列出文字*
- print(r.encoding) *#列出編碼*
- print(r.status_code) *#列出 HTTP 狀態碼*
- print(r.headers) *#列出 HTTP Response Headers*
- print(r.headers['Content-Type']) *#印出 Header 中的 Content-Type*

自訂 Header

1.回應json

```
headers = {
                        'content-type' : 'application/json; charset=utf-8',
                        'Cookie': _cookies,
                        'Referer':_Referer,
                        'user-agent':  _useragent
          }
```

2.回應html或文件內容

```
headersItems = {
                'Content-Type': 'text/html; charset=UTF-8',
                'Cookie': _cookies,
                'Referer': _Referer,
                'user-agent': _useragent
              }
```

Html Code原始碼-標籤

參考: https://www.runoob.com/tags/ref-byfunc.html

標籤下的屬性和內文：

```
1.<div class="n-top__ad n-skeleton__bg" style="width:100%;height:65px">
  </div>
2. <ul class="n-left" > .... </ul>
3. <a href="/Login?url=%2F" class="sendGA" data-
   category="All_TopRight" data-action="All_TopRight_登入" data-
   label="All_TopRight_登入">登入</a>
```

Html Code原始碼-標籤(在爬蟲上的解析)

```
<div id="HomeLabel">
<ul class="n-nav__promo ">
    <li><a href="/" title="標"><span></span></a></li>
    <li><a href="/" title="標"><span></span></a></li>
    <li><a href="/" title="標"><span></span></a></li>
</ul>
</div>
```

div.contemts

ul.contemts

li.contemts

另一種爬取資料方法webdriver套件

```
    chrome_options = webdriver.ChromeOptions()
    chrome_path = "C:\selenium_driver\chromedriver.exe"
chrome_options.add_argument('user-agent="' + _useragent + ' “ ‘)
    chrome_options.add_argument('--headless')#無介面模式
    driver = webdriver.Chrome(chrome_path,chrome_options=chrome_options)
    driver.maximize_window()
    driver.implicitly_wait(30) # seconds
    driver.get('https://www.books.com.tw/')
    time.sleep(3) ## delays for 3 seconds
 page_source=  driver.page_source  #取出原始碼
BeautifulSoup(page_source, 'lxml')
```

一、從案例中學習(spider books(博客來) from spider code)

1. 專案分析；將目標網站中的所指定目錄項下 商品資料爬取後產生JSON檔案放在伺服器下指定位置。
2. 解析目標網站結構(URL結構,頁次,總商品數, 標籤結構,網站設計方式)
3. Import os, json, bs4, selenium, fake_useragent, datetime, pandas

spider books(博客來)（解析目標網站結構）

1. URL參數值(頁次,每頁產生多少目標資訊,橫向或縱向顯示，可爬取資料總數值)
2. 目標網站所爬取資訊是否動態後端產生(使用webdriver套件爬取)
3. 從目標網站中的指定網址搜尋商品分類開始spider

https://www.books.com.tw/web/sys_categoryIndex

spider books(博客來)（商品分類)

1. 全站商品分類:

https://www.books.com.tw/web/sys_categoryIndex

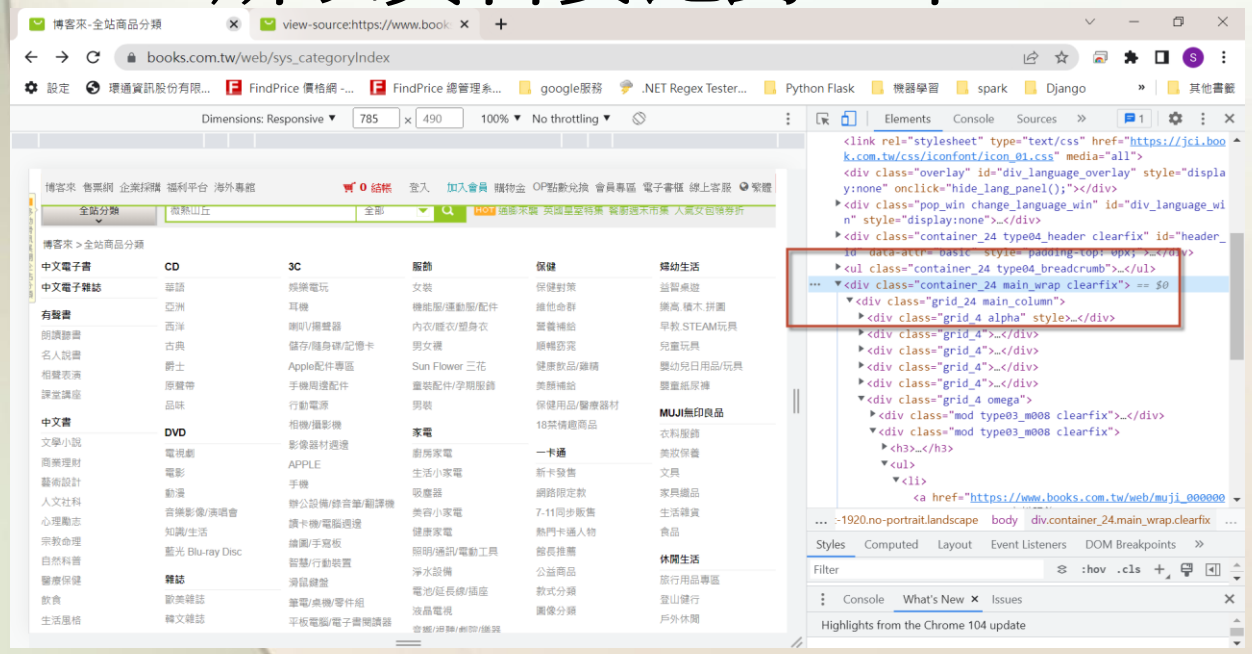
1. 爬至底層才有商品資訊

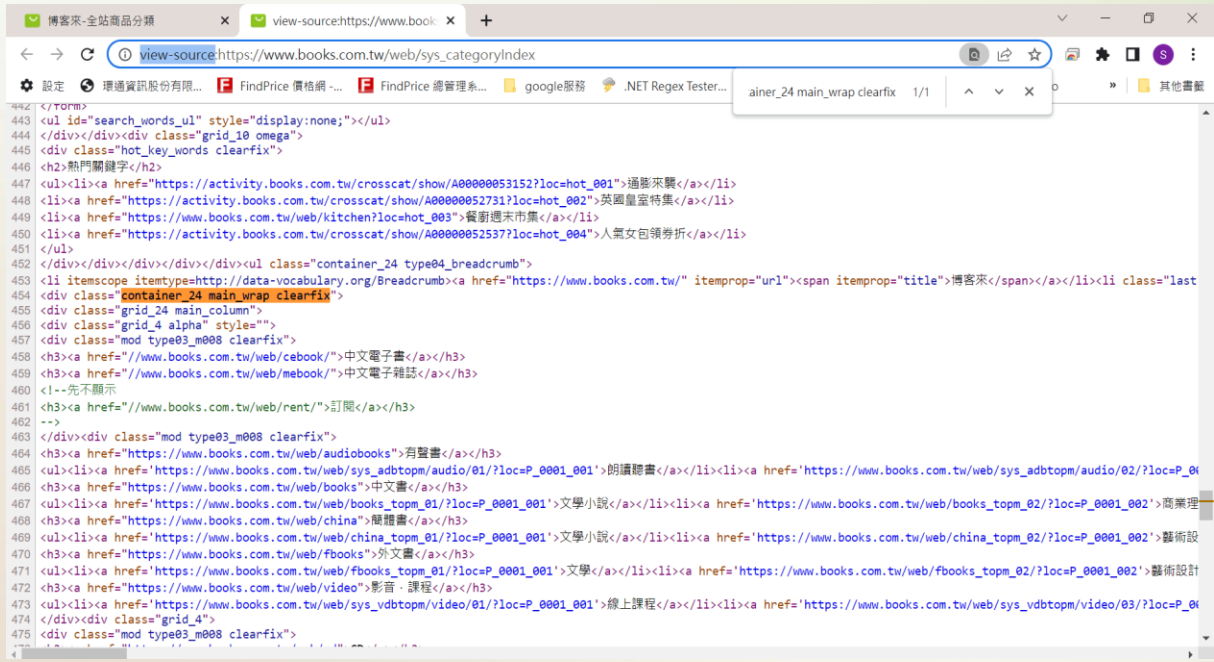
Google
F12,點頁面 檢視原始碼OR檢查

spider books(博客來)

1. 剪不斷理還亂(標籤要唯一，不拖泥帶水)
2. 目的要清楚(爬東卻爬西)
3. Elements 原始碼 / view-source 有可能不同標籤

所以資料要比對一下





spider books(博客來)-開始實作

1. 建立自己共用函示庫
2. requests.get url 是否回應status_code=200

- 200 - 請求成功
- 301 - 資源（網頁等）被永久轉移到其它URL
- 404 - 請求的資源（網頁等）不存在
- 500 - 內部服務器錯誤

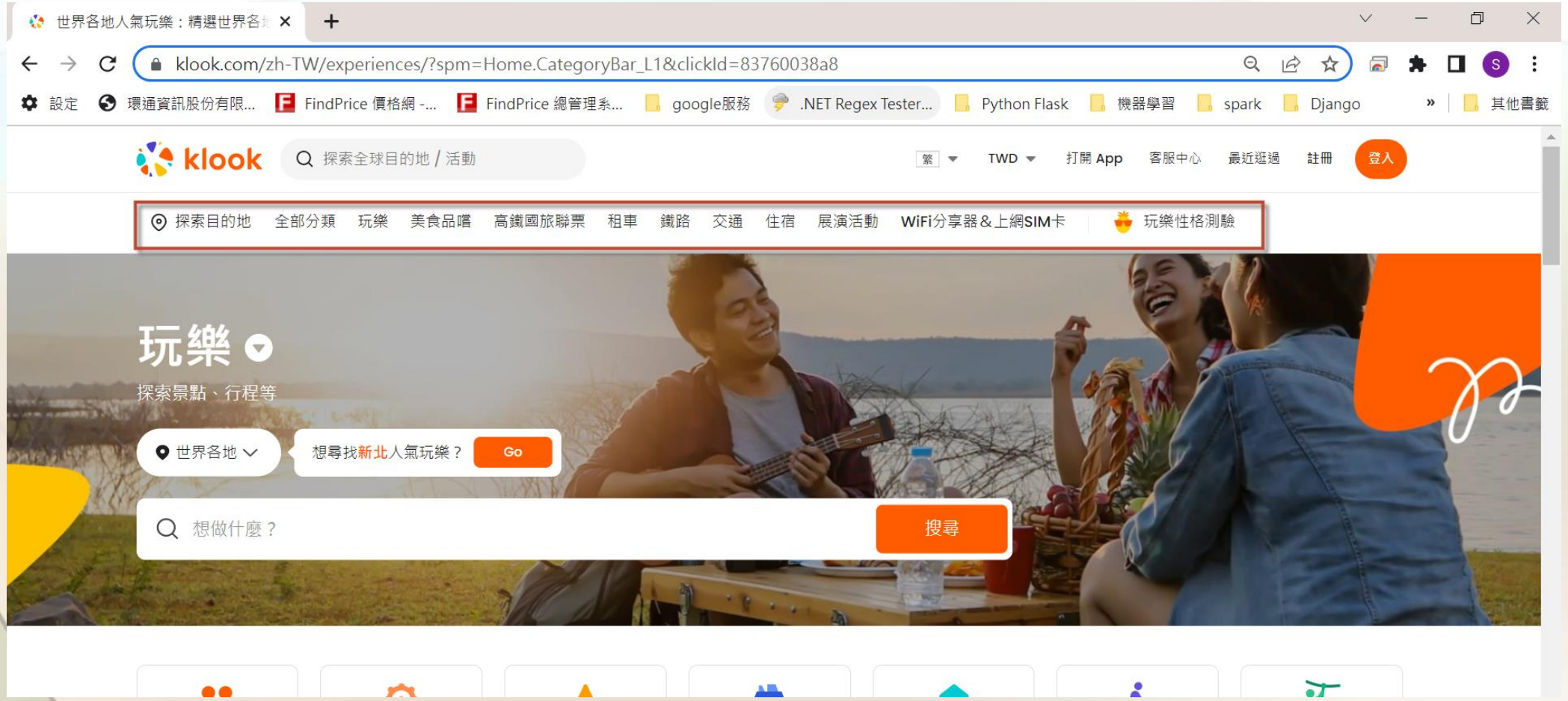
```
1.  # import requests module
import requests     # Making a get request
response = requests.get('https://api.github.com/') # print response
print(response)     # print request status_code
print(response.status_code)
```

二、從案例中學習(spider klook from api)

1. 專案分析；將目標網站中的所指定目錄項下 商品資料爬取後產生JSON檔案放在伺服器下指定位置。
2. 解析目標網站結構(URL結構,頁次,總商品數, API URL)
3. Import os, json, bs4, selenium, fake_useragent, datetime, pandas

spider klook（解析目標網站結構）

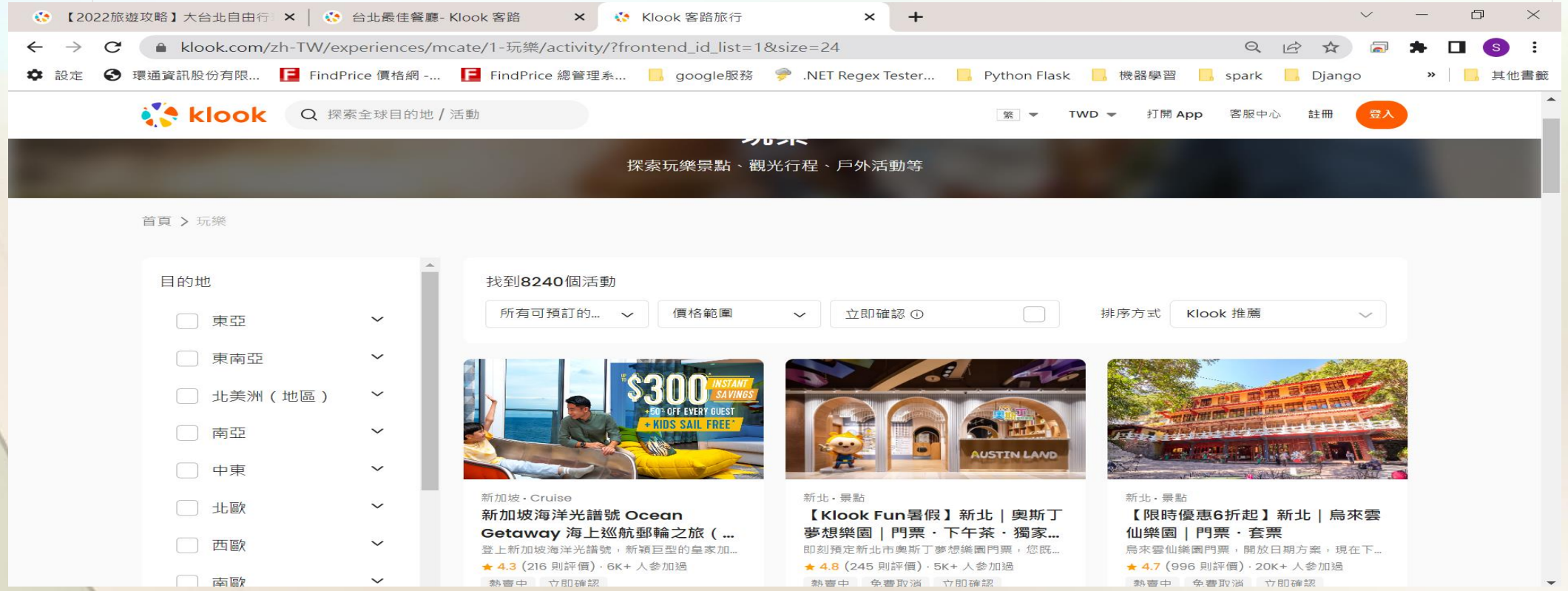
1. 先了解那些目錄可以爬到你要的商品資訊



spider klook（解析目標網站結構）-開始實作

1. 從單一目錄下就可以爬到整站或目標分類下所有商品資訊
2. 關鍵字:全部商品、上層分類
3. EX:[玩樂/總覽/顯示全部]

https://www.klook.com/zh-TW/experiences/mcate/1-%e7%8e%a9%e6%a8%82/activity/?frontend_id_list=1&size=24

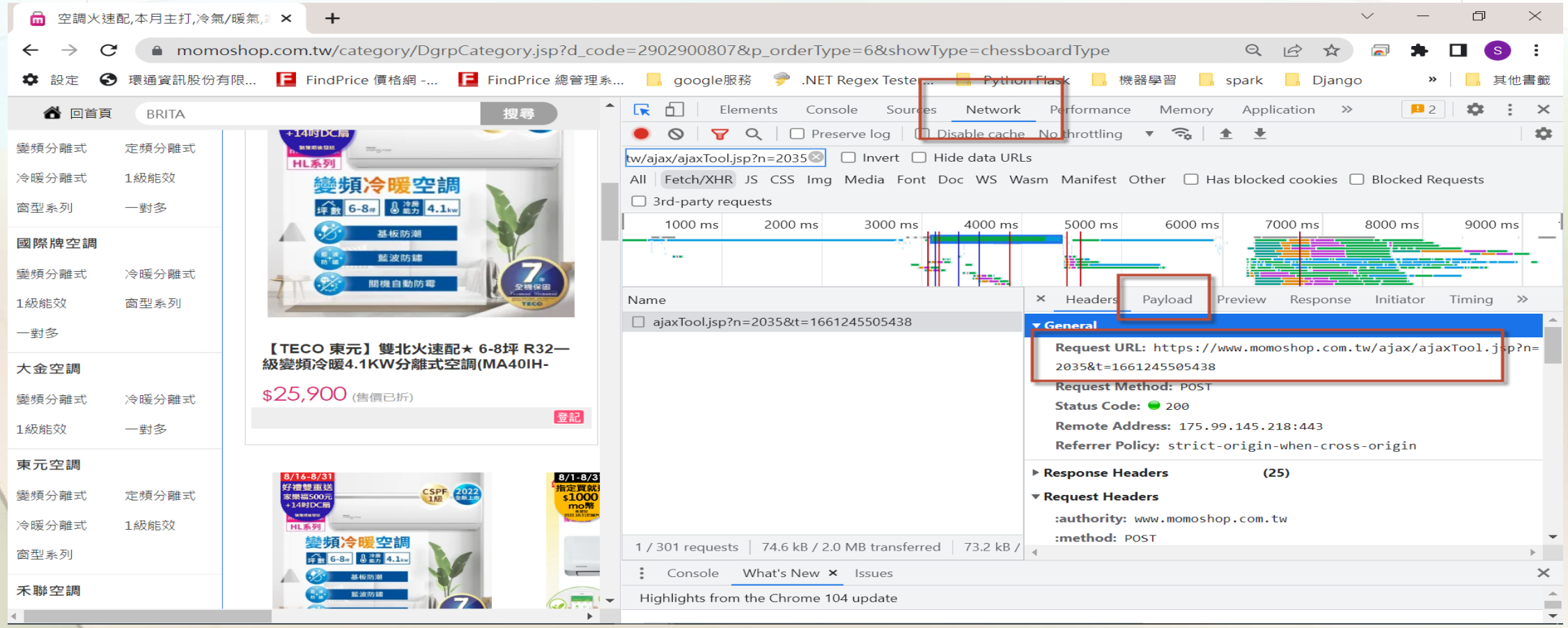


目標網站-POST 爬法

以momoshop為例：

Apiurl: https://www.momoshop.com.tw/ajax/ajaxTool.jsp?n=2035 + [13碼時間戳]

分類代碼:d_code ；



目標網站-POST 爬法

- 以momoshop為例： 傳遞參數
-
- d_code =comm.StartEndStrTrun(self.url,'?d_code=','&')
- timeStamp = int(round(time.time() * 1000)) #13碼時間戳
- apiurl='https://www.momoshop.com.tw/ajax/ajaxTool.jsp?n=2035' + str(timeStamp) #&t=1643162298549
-
- result_Items = 'data=' + urllib.parse.quote(str('{"flag":2035,"data":{"params":{"keyword":"","checkedBrands":[],"checkedBrandsNo":[],"checkedAttrs":{},"checkedAttrsNo":{},"isBrandZone":false,"specialGoodsType":"","cateCode":"'+ d_code +'","cateLevel":"3","cp":"N","NAM":"N","normal":"N","first":"N","freeze":"N","superstore":"N","tvshop":"N","china":"N","tomorrow":"N","stockYN":"N","prefere":"N","threeHours":"N","curPage":"' + str(pagecnt) +'","priceS":"","priceE":"","brandName":[],"sortType":"6","d_code":"'+ d_code +'","p_orderType":"6","showType":"chessboardType"}}}'))
- #try:
- headers = {
- 'Content-Type' : 'application/x-www-form-urlencoded; charset=UTF-8',
- 'Accept' : 'application/json, text/javascript, */*; q=0.01',
- 'X-Requested-With' : 'XMLHttpRequest',
- 'referer': self.url,
- 'Cookie': _cookies,
- 'user-agent': _useragent
- }
- res = requests.post(apiurl, data = result_Items, headers = headers, timeout=1200)
- if str(res.status_code) == '200’ :

目標網站-JSON解析

以momoshop為例： json.loads()

```
Import josn

Headers={
…..
}

 res = requests.post(apiurl,data = result_Items, headers = headers, timeout=1200)
 if str(res.status_code) == '200’ :
     prods = json.loads(res.text.replace( ‘\n’ , “)) #從網路上文件取出會有斷行要濾掉
    for goodsdata in prods['rtnData']['rtnGoodsData']['rtnGoodsData']['goodsInfoList’ ]:
        ……
```

目標網站-識別網站圖像驗證碼

- **網站圖像驗證：反制爬蟲一種手段**
- 常見的驗證碼有這麼幾種

圖像驗證
語音驗證
簡訊驗證
極驗驗證
推理驗證

目標網站-識別網站圖像驗證碼

套件:pip install pytesseract
from PIL import Image
import pytesseract

- 一、selenium截取驗證碼
- 二、安裝識別環境pytesseract+Tesseract-OCR（C:\Program Files\Tesseract-OCR)
  - 驗證識別環境是否正常
- 三、處理驗證碼圖片
  - 圖片處理識別
- pytesseract是一個基於Google『Tesseract-OCR的獨立封裝包
- pytesseract功能是識別圖片文件中文字，並作爲返回參數返回識別結果
- pytesseract默認支持tiff、bmp格式圖片，只有在安裝PIL之後，才能支持jpeg、gif、png等其他圖片格式
  -

http://jasonyychiu.blogspot.com/2020/09/pythonpytesseract-pytesseractpytesserac.html

目標網站-識別網站圖像驗證碼

1.黑白圖
captcha = Image.open(圖片路徑)
result = pytesseract.image_to_string(captcha)
print(result)

目標網站-識別網站圖像驗證碼

2.處理彩色的數字圖片:將圖片以灰接處理

```
def convert_img(img,threshold):
    img = img.convert("L")  # 處理灰白
    pixels = img.load()
    for x in range(img.width):
        for y in range(img.height):
            if pixels[x, y] > threshold:
                pixels[x, y] = 255
            else:
                pixels[x, y] = 0
    return img

convert_img(captcha,150)#像素範圍
result = pytesseract.image_to_string(result)
print(result)
```

目標網站-識別網站圖像驗證碼

3..處理有雜訊的數字圖片

```
data = img.getdata()
   w,h = img.size
   count = 0
   for x in range(1,h-1):
      for y in range(1, h - 1):
         # 找出每個像素方向
         mid_pixel = data[w * y + x]
         if mid_pixel == 0:
            top_pixel = data[w * (y - 1) + x]
            left_pixel = data[w * y + (x - 1)]
            down_pixel = data[w * (y + 1) + x]
            right_pixel = data[w * y + (x + 1)]
            if top_pixel == 0:
               count += 1
            if left_pixel == 0:
               count += 1
            if down_pixel == 0:
               count += 1
            if right_pixel == 0:
               count += 1
            if count > 4:
               img.putpixel((x, y), 0)
result = pytesseract.image_to_string(captcha)
print(result)
```

目標網站-動態資料 爬法

1. **動態網頁爬蟲重要的等待機制：**網頁載入所要使用的元素，在執行其它的操作。若沒有處理好，就會時常發例外錯誤或影響執行效率。甚至爬不到資料數據。
2. 在目標網站在轉圈圈的圖示時，代表網頁正在載入內容元素，在這個圖示消失前，網頁不會顯示所有的內容，所以等待(Waits)就很重要。
3. 另外一種方法找腳本尾端的標籤移動位置。

https://loveiizakka.com/

•sleep(強制等待)
•Implicit Waits(隱含等待)
•Explicit Waits(明確等待)

目標網站-動態資料 爬法

1. driver.find_element_by_partial_link_text(m_c_name) # 尋找 <a href 連結 標題 name

2. search_button = driver.find_element_by_id(dropMenu_id) # 找名稱並點擊

3. ActionChainsDriver = ActionChains(driver).move_to_element(search_button)
   time.sleep(8)
   ActionChainsDriver.perform()#滑鼠滑動至指定元素
   time.sleep(5)
   page_source_click = driver.page_source
   page_source_click_sub = self.BeautifulSoup(page_source_click, 'lxml')

主要目的:模擬用戶動作並點擊標籤或指定到的元素

目標網站-動態資料 爬法

- 以東森購物網為例

https://www.etmall.com.tw/s/12

點擊[更多] button

```
cate_next_button = driver.find_elements_by_xpath("//span[@class='span-n-right']")[0]

time.sleep(3)
cate_next_button.click()
page_source_click = driver.page_source
 page_source_click_sub = self.BeautifulSoup(page_source_click, 'lxml')
```

主要目的:模擬用戶動作並點擊按鈕

資料視覺化—**Matplotlib**

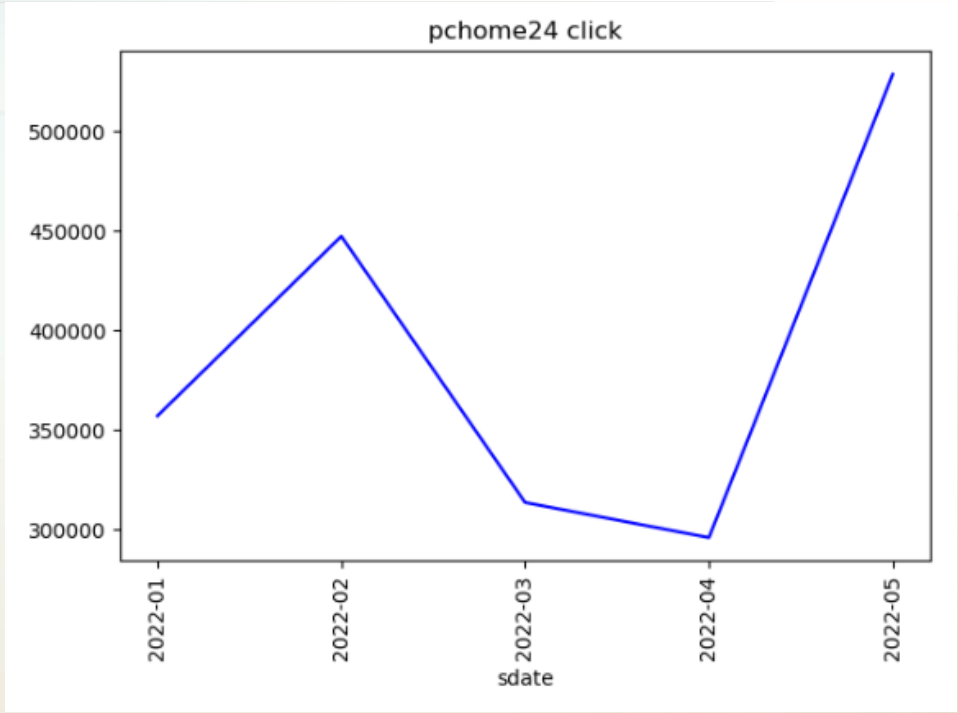
1. **常見的數據圖:**折線圖、散布圖、長條圖、直方圖、盒狀圖以及圓餅圖
2. import matplotlib.pyplot as plt
   import pandas as pd

資料視覺化—Matplotlib

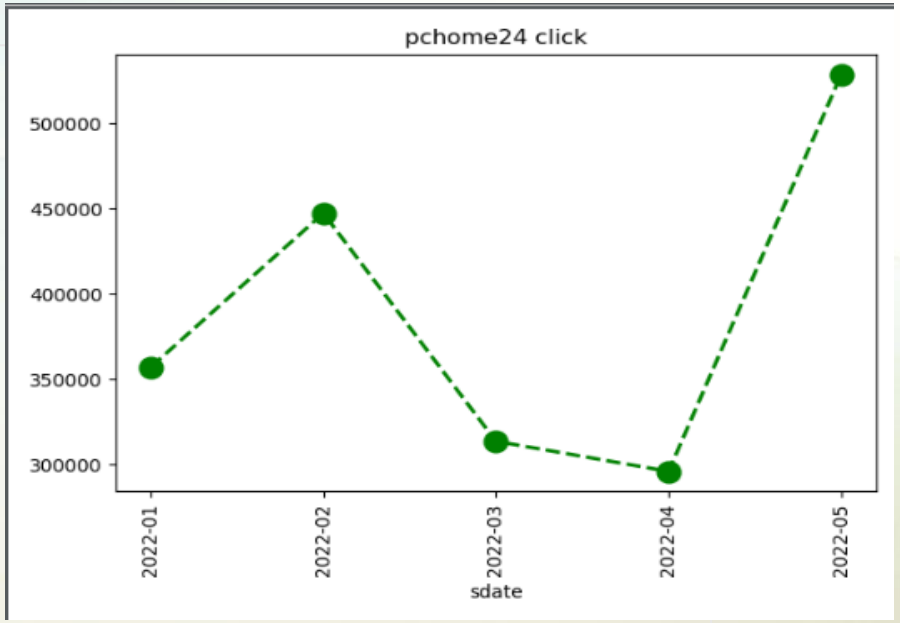
➢ 折線圖 **Line Plot**

資料:dataframe

```
pchome24 = pd.DataFrame(columns=['sdate', 'click'])
pchome24.loc[0] = ['2022-01', 357272]
pchome24.loc[1] = ['2022-02', 447440]
pchome24.loc[2] = ['2022-03', 313719]
pchome24.loc[3] = ['2022-04', 296085]
pchome24.loc[4] = ['2022-05', 528721]

plt.plot(pchome24.sdate, pchome24.click, color='b')
plt.xlabel('sdate')  # 設定x軸標題

plt.xticks(pchome24.sdate, rotation='vertical') # 設定x軸label以及垂直顯示
plt.title('pchome24 click') # 設定圖表標題
plt.show()
```



資料視覺化—Matplotlib



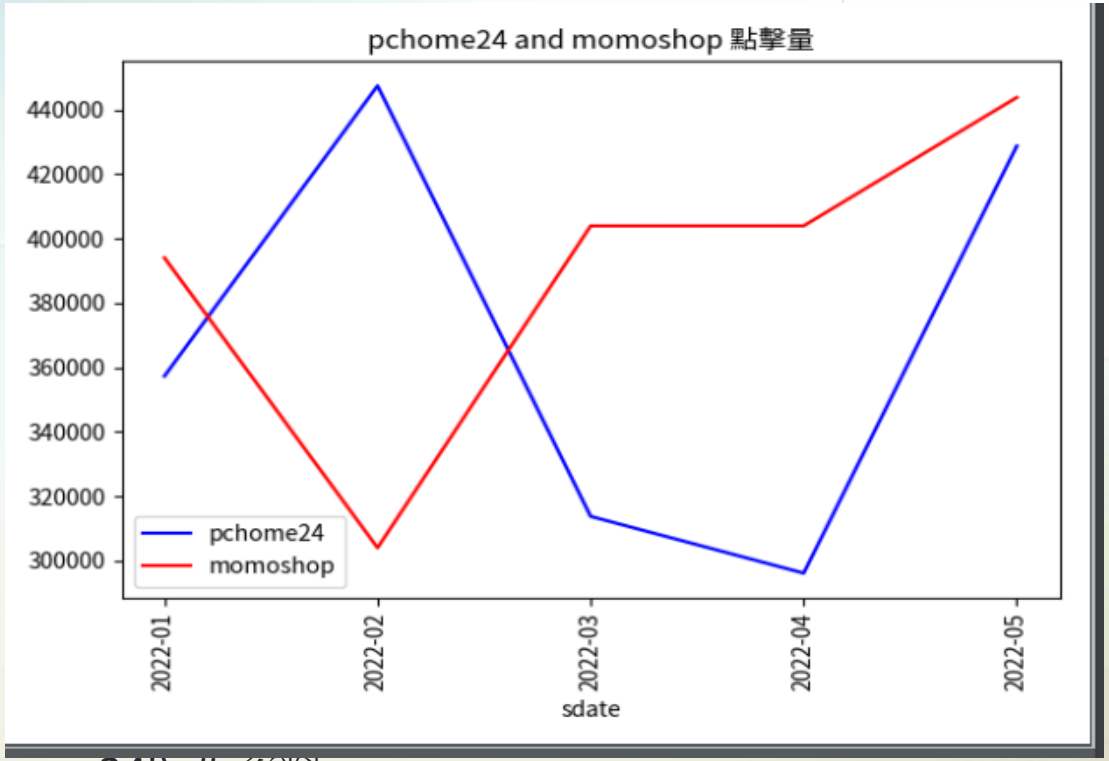
➢ 折線圖 **Line Plot**

資料:dataframe

```
pchome24 = pd.DataFrame(columns=['sdate', 'click'])
 pchome24.loc[0] = ['2022-01', 357272]
 pchome24.loc[1] = ['2022-02', 447440]
 pchome24.loc[2] = ['2022-03', 313719]
 pchome24.loc[3] = ['2022-04', 296085]
 pchome24.loc[4] = ['2022-05', 528721]

 plt.plot(pchome24.sdate, pchome24.click, color='green', marker='o',
linestyle='dashed',linewidth=2, markersize=12)

 plt.xlabel('sdate')  # 設定x軸標題

 plt.xticks(pchome24.sdate, rotation='vertical') # 設定x軸label以及垂直顯示
 plt.title('pchome24 click') # 設定圖表標題
 plt.show()
```

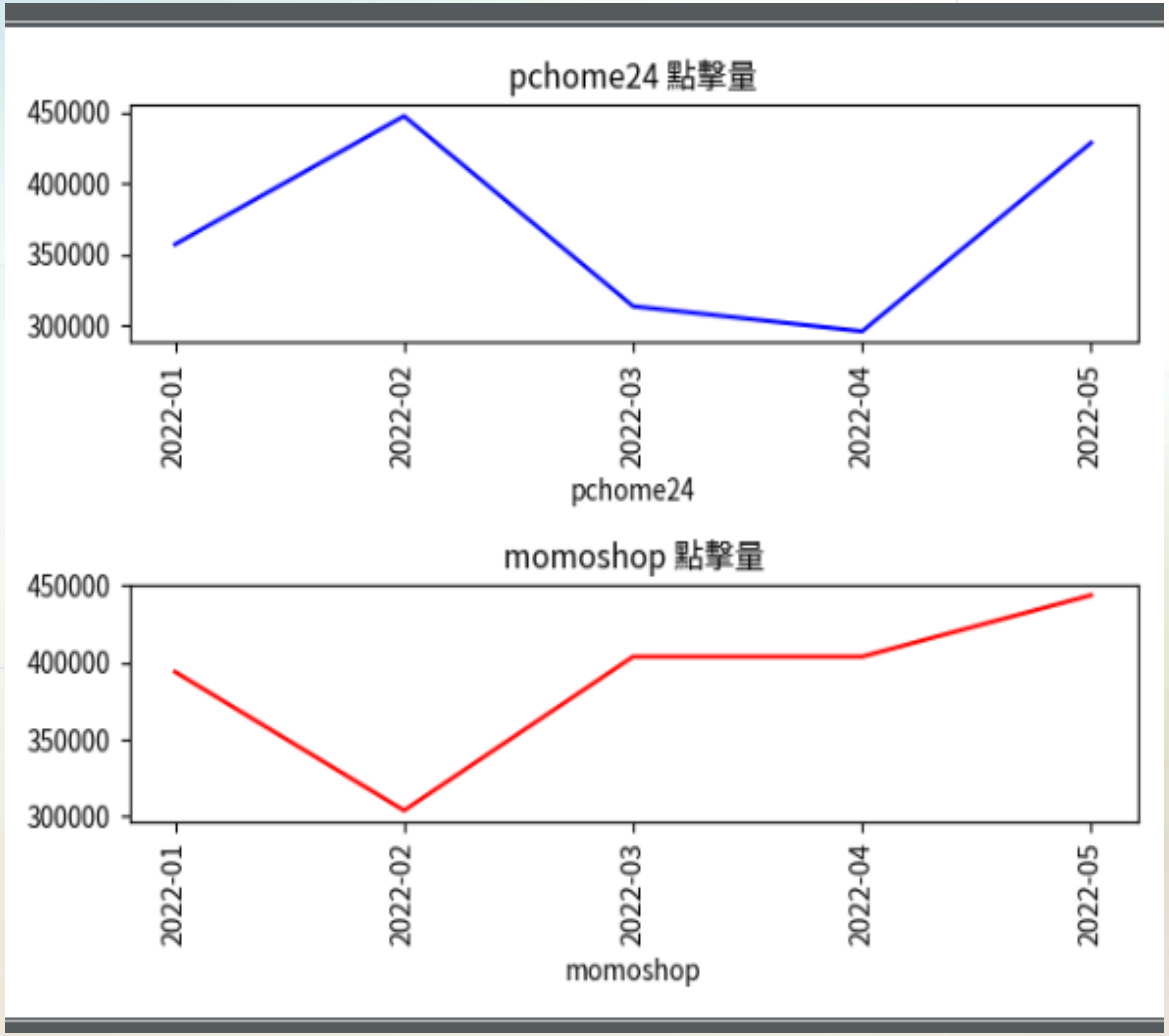
➢ **1.一張圖表上繪製兩筆不同的資料線，只需要重複 plt.plot()**

```
pchome24 = pd.DataFrame(columns=['sdate', 'click'])
 pchome24.loc[0] = ['2022-01', 357272]
 pchome24.loc[1] = ['2022-02', 447440]
 pchome24.loc[2] = ['2022-03', 313719]
 pchome24.loc[3] = ['2022-04', 296085]
 pchome24.loc[4] = ['2022-05', 428721]

 momoshop = pd.DataFrame(columns=['sdate', 'click'])
 momoshop.loc[0] = ['2022-01', 394033]
 momoshop.loc[1] = ['2022-02', 303958]
 momoshop.loc[2] = ['2022-03', 403922]
 momoshop.loc[3] = ['2022-04', 403937]
 momoshop.loc[4] = ['2022-05', 443824]

 plt.plot(pchome24.sdate, pchome24.click,'b', label ='pchome24')  # 資料一
 plt.plot(momoshop.sdate, momoshop.click, 'r', label='momoshop')  # 資料一
 plt.xlabel('sdate')
 plt.xticks(pchome24.sdate, rotation='vertical')  # 兩種資料來源相同
 plt.title('pchome24 and momoshop 點擊量')
 plt.legend(loc='lower left') #設定圖例
 plt.show()
```



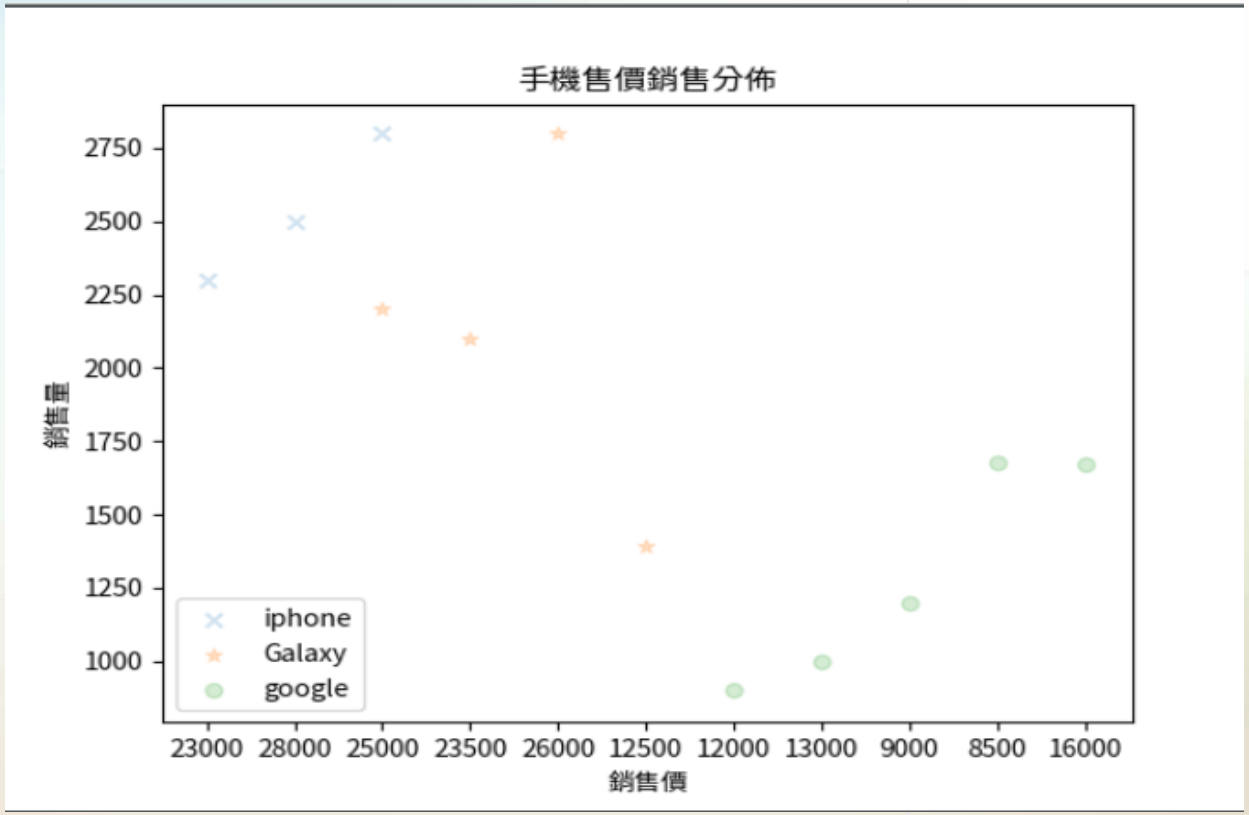
➢ **2. 一張圖表上繪製兩筆不同的資料線，只需要重複 plt.plot()**

```
plt.subplot(2, 1, 1)
 plt.plot(pchome24.sdate, pchome24.click, 'b')
 plt.xlabel('pchome24')
 plt.xticks(pchome24.sdate, rotation='vertical')
 plt.title('pchome24 點擊量')
 plt.subplot(2, 1, 2)
 plt.plot(momoshop.sdate, momoshop.click, 'r')
 plt.xlabel('pchome24')
 plt.xticks(momoshop.sdate, rotation='vertical')
 plt.title('momoshop 點擊量')
 plt.tight_layout()  # 隔開兩個圖
 plt.show()
```



- **散布圖 Scatter Plot**
- 通常用於描繪兩筆資料之間的相關程度，在進行機器學習或是資料科學的運用時，這樣的特性也使其多被用在 feature engineering 時的參考。

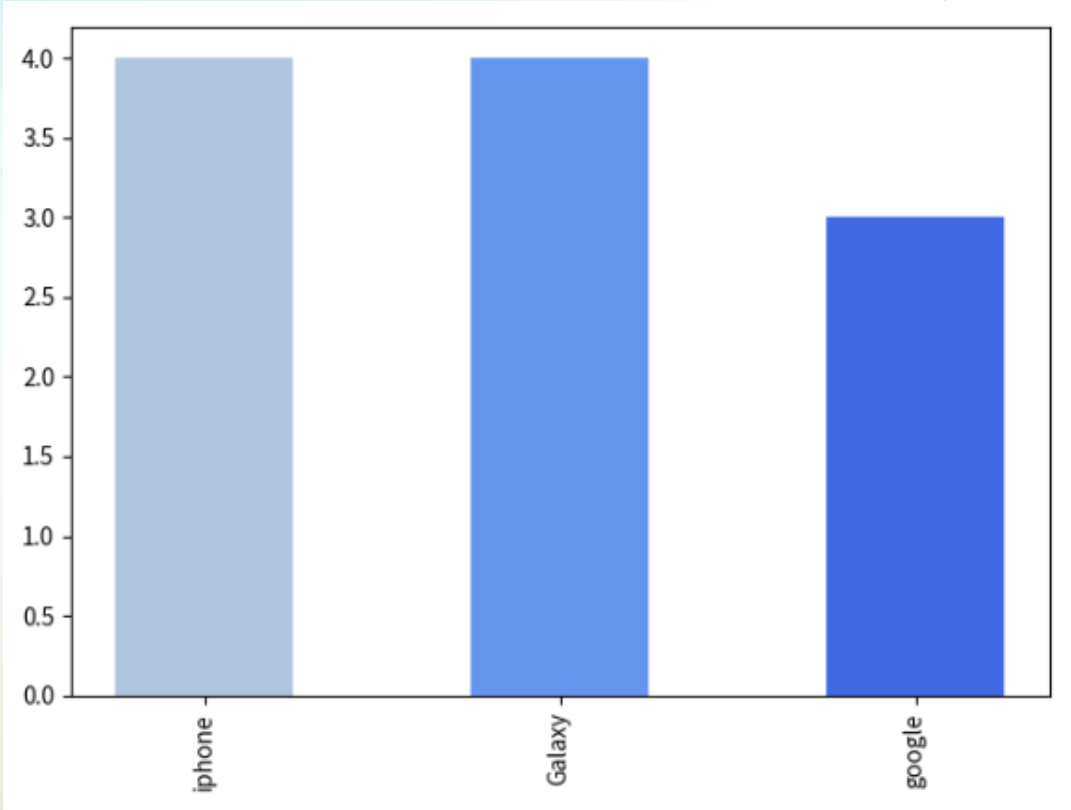
```
pchome24 = pd.DataFrame(columns=['gprice', 'buycnt','gtype'])
pchome24.loc[0] = ['23000', 2300,'iphone']
pchome24.loc[1] = ['28000', 2500,'iphone']
pchome24.loc[2] = ['23500', 2100,'Galaxy']
pchome24.loc[3] = ['12000', 900,'google']
pchome24.loc[4] = ['13000', 1000,'google']
pchome24.loc[5] = ['25000', 2200,'Galaxy']
pchome24.loc[6] = ['25000', 2800,'iphone']
pchome24.loc[7] = ['9000', 1200,'google']
pchome24.loc[8] = ['26000', 2800,'Galaxy']
pchome24.loc[9] = ['12000', 1200,'Galaxy']
pchome24.loc[10] = ['9900', 1450,'google']
pchome24.loc[11] = ['13500', 1270,'google']
pchome24.loc[9] = ['12500', 1390,'Galaxy']
pchome24.loc[10] = ['8500', 1678,'google']
pchome24.loc[11] = ['16000', 1670,'google']

df1 = pchome24[pchome24['gtype'] == 'iphone']
df2 = pchome24[pchome24['gtype'] == 'Galaxy']
df3 = pchome24[pchome24['gtype'] == 'google']
plt.scatter("gprice", "buycnt", data=df1, marker='x', alpha = 0.2,label ='iphone')
plt.scatter("gprice", "buycnt", data=df2, marker='*', alpha = 0.2,label ='Galaxy')
plt.scatter("gprice", "buycnt", data=df3, alpha=0.2,label ='google')
plt.annotate('iphone', xy=(20000, 2000), xytext=(28000, 2800), arrowprops={'color': 'green'})
plt.annotate('Galaxy', xy=(12000, 1000), xytext=(19999, 1900), arrowprops={'color': 'blue'})
plt.annotate('google', xy=(8500, 800), xytext=(11999, 999), arrowprops={'color': 'orange'})

plt.title('手機售價銷售分佈')
plt.xlabel('銷售價')
plt.ylabel('銷售量')
plt.legend(loc='lower left')  # 設定圖例
plt.show()
```



直方圖 Histogram

- 用來表達一組連續數值資料的分佈

```
plt.bar(pchome24.gtype.unique(),
        pchome24.gtype.value_counts(),
        width=0.5,
        bottom=None,
        align='center',
        color=['lightsteelblue',
               'cornflowerblue',
               'royalblue',
               'midnightblue',
               'navy',
               'darkblue',
               'mediumblue'])
plt.xticks(rotation='vertical')
plt.show()
```



補充-解決中文問題(plt)

1.其他問題:解決中文問題(plt)
https://daxpowerbi.com/%E5%A6%82%E4%BD%95%E5%9C%A8win-10%E8%A7%A3%E6%B1%BAmatplotlib%E4%B8%AD%E6%96%87%E9%A1%AF%E7%A4%BA%E7%9A%84%E5%95%8F%E9%A1%8C/

1. C:\MyData\Python教學\img\font #從google fonts下載自型檔
2. 參考:https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html
3. 解壓縮後,選一字型檔copy至 C:\Users\jalen\AppData\Local\Programs\Python\Python35\Lib\site-packages\matplotlib\mpl-data\fonts\ttf
4. 檢視C:\Users\jalen\.matplotlib*.json 打開檔案檢是內容有無
5. 參考:

https://medium.com/@yuhsuan_chou/python-%E5%9F%BA%E7%A4%8E%E8%B3%87%E6%96%99%E8%A6%96%E8%A6%BA%E5%8C%96-matplotlib-401da7d14e04

https://medium.com/seaniap/matplotlib%E8%B3%87%E6%96%99%E8%A6%96%E8%A6%BA%E5%8C%96-2-%E6%9F%B1%E7%8B%80%E5%9C%96%E8%88%87%E7%9B%B4%E6%96%B9%E5%9C%96-316c7f17f199