

Discount Expert

資管二 陳佳妤 109401027

資管二 鍾侑璇 109401030

一、專題說明

【背景動機與目的】

在網路日漸普及的年代，購物形式也在悄悄變化。商人為了跟進這樣的消費模式，也紛紛架起網路商店。在如此多競爭者的壓力下，現今購物網站上常有特價或特惠等活動來吸引顧客購買，拍賣商也常常打出「最低 XX 折起」看似極度優惠消費者的標示，實際上真正使用最低折數的商品卻只占了一小部分。消費者難以看出折扣幅度，還需要透過手動計算才能不被商家用這樣的手段哄騙做下不划算的決定。因此，我們希望可以利用 Python 爬蟲來整理出所有商品不同的折扣數，藉此幫助消費者能更迅速簡潔的在眾多商品中找到最特惠的商品。

以下我們將以「Lativ 的暖心推薦」作為實作範例。在 Lativ 的網站中 (lativ.com.tw/OnSale/3P80DUP)，我們可以看到總共分有五大區，分別為「Women」、「Men」、「Kids」、「Baby」和「Sports」，且每一個分區的商品有 3~6 頁不等，我們利用爬蟲動態換頁的方法，獲取各區域內每一樣商品的原價和折扣價，再計算成折扣數，並以表格呈現，讓消費者可以一覽無遺的瀏覽所有商品的價格資訊。最後，消費者可以依照不同折扣數，去查找個人能接受的商品，使精打細算的消費者們，能夠花費更少的購物時間，找到最划算的商品，提升整體購物效率。

二、專題介紹

【使用之套件介紹】

1. Selenium

我們使用 Selenium 來實現啟動瀏覽器、尋找網站中的特定元素和自動化爬蟲。

A	WebDriver	啟動瀏覽器
---	-----------	-------

B	<code>find_element_by_XX</code>	元素定位
C	<code>ActionChains(driver).click(XX).perform()</code>	模擬滑鼠在瀏覽器中的使用

➤ WebDriver

WebDriver 是用來控制瀏覽器的行為，各瀏覽器都會有各自相對應的驅動程式(Driver)，而 Selenium 透過這個驅動程式，讓瀏覽器看起來像是真人在使用一樣，進一步實現爬蟲自動化。因此，我們不再需要將我們想達成的動作一個一個用爬蟲的程式碼寫出來執行，而是透過 Selenium 的 WebDriver 來達到瀏覽器自動化。

比較需要注意的是，在使用 WebDriver 之前，我們要先依照所使用的瀏覽器(ex. Google Chrome)分別下載各自的驅動程式(ex. Chromedriver.exe)，並且放在與專案同資料夾底下或者是在宣告使用時記得加上自己所放置的路徑(ex. `driver = webdriver.Chrome("C:Users/USER/chromedriver.exe")`)

➤ `find_element_by_XX`

在此專題中，我們是使用到了 `find_element_by_css_selector` 和 `find_elements_by_xpath` 來分別定位「下一頁」的按鈕以及各分類名稱，這些都是幫助我們能快速定位在網頁原始碼中的特定元素，並執行某些動作。除了我們所使用的這兩個之外，也可依照需求來對應加上不同的元素名稱，包括：`id`、`name`、`link_text`、`tag_name`、`class_name` 等。另外，`element` 和 `elements` 分別是獲取第一個搜尋到的元素和獲取所有元素，可依據所需來使用這個指令。

➤ ActionChains

在介紹 ActionChains 前，要先提到 Ajax(非同步動態網頁，Asynchronous、JavaScript 和 XML 的縮寫)。在實作 Python 網頁爬蟲爬取網頁時，可以發現有些網頁會使用到 Ajax 的技術，非同步向伺服器傳送參數，讓使用者對伺服器端送出 request 之後，不需要等待結果，仍可以持續處理其他事情，甚至繼續送出其他 request。回傳值之後，便能被融合進當下頁面或應用中。

Lativ 的網頁為了讓消費者能將商品放入購物車，在送出資料時，也能繼續

瀏覽網頁，因此他們的按鈕採用的是 Ajax（附圖一），包括「上、下一頁」以及「分類的按鈕」，所以在同一頁的網址均不會改變（附圖二、三），導致我們無法直接使用 requests 來進行換頁存取，因此我們改採用 Selenium 的 ActionChains。

ActionChains 可以完成簡單的動作，並模擬比較複雜的行為，像是滑鼠停在某連結上或是拖曳的行為，例如滑鼠移動、點擊、輸入等。而使用 ActionChains 來完成這些動作後，這些動作將會在使用到 perform() 時，依序來觸發。我們便是使用 ActionChains(driver).click(XX).perform()，在瀏覽器中點擊後觸發來達到換頁或是換分類的動作。

2. BeautifulSoup

BeautifulSoup 可以解析網頁 HTML 碼，並擷取網頁內容的資料，然後再利用 .text 或 .string 取得純文字。

A	BeautifulSoup(res.text, 'html parser')	HTML 解析器
B	soup.select	選取特定的標籤
C	soup.find_all	找到全部特定的 html tag
D	.text	.text 屬性為 html 檔案

3. requests

requests 會建立適當的 HTTP 請求，透過 HTTP 請求從網頁伺服器下載指定的資料。

A	requests.get	使用 GET 方式下載普通網頁
B	requests.post	用 POST 請求來處理(通常是網頁中有讓使用者填入資料的表單)

4. pandas

利用 Pandas 建立 Dataframe，將原本存放資料的 list，用表格來呈現，讓資

料更好瀏覽。

A	<code>pandas.DataFrame</code> (<code>data=None</code> , <code>index=None</code> , <code>columns=None</code>)	可自定義各項參數值來建立希望取得的表格。 <code>data</code> 為欲放入的資料， <code>index</code> 為列的名稱， <code>columns</code> 為欄的名稱，
B	<code>DataFrame.drop_duplicates</code> (<code>subset=None</code> , <code>keep='first'</code> , <code>inplace=False</code>)	可將重複的資料剔除、自定義各項參數值。 <code>Subset</code> 為要從哪裡判斷重複值的欄位名稱， <code>keep</code> 可設為 <code>first</code> , <code>last</code> , <code>false</code> 分別是留下第一個重複的資料，其餘刪去、留下最後一個重複的資料，其餘刪去和刪去所有重複的資料，預設值為 <code>first</code> 。 <code>Inplace</code> 可設要返回重複項還是副本，預設值為 <code>false</code> 。
C	<code>DataFrame.T</code>	列行資料交換
D	<code>DataFrame.sort_values</code> (<code>by = " "</code> , <code>ascending=True</code>)	可自定義各項參數值並做排序。 <code>By</code> 可決定要以哪項欄位來做排序基準， <code>ascending</code> 為升序排列，預設值為 <code>false</code> 。
E	<code>DataFrame.reset_index</code> (<code>drop=False</code> , <code>inplace=False</code>)	因為經過 <code>drop_duplicates</code> ，所以資料編號要經過重新編碼。 <code>reset_index</code> 可設定各參數值， <code>inplace</code> 為是否用一個新物件來儲存資料，預設為否， <code>drop</code> 為是否要用 Pandas 預設的 <code>int index</code> 來做 <code>reset_index</code> ，預設為否。
F	<code>DataFrame.at[i, '欄位名稱']</code>	呈現欄位的資料，前面為第 <code>i</code> 列，後面為欄位。

5. time

在 `time` 中我們只使用了 `sleep`，目的是為了網頁可以全部獲取完才繼續執行自動化爬蟲的動作。使用這個是因為在撰寫程式的過程中，發現如果沒有做這樣的動作，程式很容易出現報錯訊息（圖四、五），原因就是因為網頁還沒獲取完，爬蟲就已經結束了，因此找不到對應元素，便無法進行原本的動作。

經過我們不斷的嘗試，在換分類時，sleep(2)，停止兩秒；在同分類換頁時，sleep(1)，停止 1 秒，便能穩定的擷取到網頁，完成爬蟲。

【自定義函數和一特殊迴圈】

```
def snapshot():
    # 擷取單頁所有商品資訊
    soup = BeautifulSoup(driver.page_source, 'lxml')
    name = soup.select("h3[class='any_display_name']")
    ad_price = soup.select("span[class='currency symbol pricing']")
    ori_price = soup.select("span[class='currency noAboutSymbol']")

    for t in name:
        tag.append(t.text.strip())
    for p in ori_price:
        ori_pricing.append('$ ' + p.text.strip('$\xa0'))
    for a in ad_price:
        ad_pricing.append('$ ' + a.text.strip('$\xa0'))
    return tag, ori_pricing, ad_pricing
```

snapshot()是定義來擷取單頁所有商品資訊。我們利用 soup.select 定位元素後，將每一筆特定元素的資料新增到各自的列表裡(tag, ori_pricing, ad_pricing)，將衣服名稱和價格分開儲存可方便之後的資料整理。其中因為所獲得的資料裡有我們不需要的空白和文字，因此我們使用.strip()來去除。

```
def get_all():
    # 獲取全部商品資訊
    all.append(tag)
    all.append(ori_pricing)
    all.append(ad_pricing)
    print(len(all[0]))
    return all
```

get_all()是定義來將所有分類的所有商品資料儲存至單一的列表，可方便之後創立 DataFrame。

```
def changepage():
    # 換頁
    Next_Link = driver.find_element_by_css_selector('#AjaxDiv > div:nth-child(1) > a.next')
    ActionChains(driver).click(Next_Link).perform()
```

changepage()是定義來專門做換頁的動作的。我們利用 find_element_by_css_selector 找到「下一頁」的按鈕後，再使用 ActionChains

來做點擊的動作，以達到動態爬蟲中換頁的效果。

```
def switchcategory():
# 換類別
    categories = soup.find_all('li', current = 'False')
    for category in categories:
        Category_Link = driver.find_element_by_css_selector('#onSaleNav > ul > li.' + str(category.text.lower()))
        ActionChains(driver).click(Category_Link).perform()
        sleep(2) #Sleep 可更保證在擷取商品時，網頁是有完整讀取完畢的
        snapshot()
        for x in range(numOfPage()-1):
            changepage()
            sleep(1)
            snapshot()
```

switchcategory()是定義來換分類的。我們先利用 soup.find-all 來找到所有分類標籤的名字，接下來同樣利用 find_element_by_css_selector 來依次找到各分類的按鈕，再使用 ActionChains 來做點擊的動作。成功換分類後，就可以重複執行爬取每一頁的商品資訊的這項動作了。另外，因為網站中分類名稱的文字是大寫，但元素標籤卻是使用小寫，所以我們使用了.lower()來達到大寫換小寫的效果。

```
def numOfPage():
# 獲取當下頁面的頁數
    pages = driver.find_elements_by_xpath('//*[@id="AjaxDiv"]/div[1]/child::*')
    i = 0
    for page in pages:
        i+=1
    i-=2
    return i
```

numOfPage()是定義來讀取各分類的頁數的，原因是因為有些衣服銷售一空之後，商品總數減少，頁數總數也會跟著改動，因此我們利用 find_elements_by_xpath 來得到頁數的數量，以達到在各分類擷取商品資訊時，可以動態得調整迴圈數，增加爬蟲的效率。

```

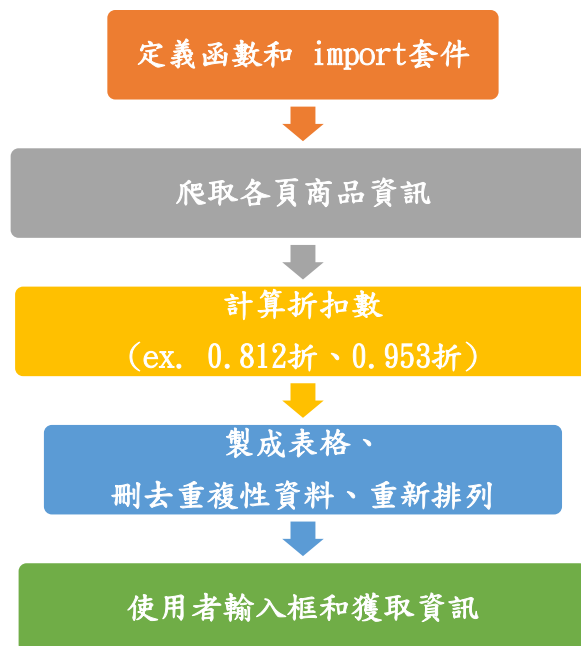
for i in range(len(all_array[0])):
    num1 = [str(temp) for temp in all_array[2][i].split() if temp.isdigit()]
    num1_string = "".join(num1)
    num1_int = int(num1_string)

    num2 = [str(temp) for temp in all_array[1][i].split() if temp.isdigit()]
    num2_string = "".join(num2)
    num2_int = int(num2_string)
    final_num.append(str(round((num1_int/num2_int),3)))

```

折數的計算：這段迴圈的程式碼的主要目的是因為原先資料顯示的並不是 int 整數型態，故無法直接相除，所以我們必須先做 str 和 int 之間的轉換，用來得到我們想要的折扣數。其中我們利用的是.isdigit()的判斷，這邊將會將原本的[\$199]，變成[199]，list 沒辦法直接轉為整數型態，因此，我們將它先轉型為字串，再由字串轉型為整數型態。

【流程圖】



<1>定義函數和 import 套件

1.1 引入以上介紹之套件和自定義函數。

1.2 用 requests.get()獲取網頁。

1.3 用 BeautifulSoup 套件來產生 HTML 解析檔，將用於後面的元素標籤搜尋。

<2>爬取各頁商品資訊

2.1 在迴圈內呼叫 `changepage()`、`sleep(1)`和 `snapshot()`，爬取 Women 分類內的所有商品資訊。迴圈的 `range` 由函數 `numOfPage()`取得。

2.2 呼叫自定義函數 `switchcategory()`。函數內，有一個雙層迴圈，第一層的目的是切換不同分類，第二層的目的是爬取各分類內的所有商品資訊。迴圈的 `range` 同樣由函數 `numOfPage()`取得。

<3>計算折扣數(ex. 0.812 折、0.953 折)

3.1 在迴圈內，將折扣價和活動價轉換成整數型態後，相除得到折扣數，並存到叫 `Discount` 的 `list` 裡面。

<4>製成表格、刪去重複性資料、重新排列

4.1 利用 `pandas` 套件創立一個 `DataFrame`，將資料變的更容易瀏覽

4.2 用 `T`、`drop_duplicates`、`sort_values` 和 `reset_index` 來達到旋轉、刪除重複性資料、降序排列和命名。

<5>使用者輸入框和獲取資訊

5.1 使用者輸入框，讓他們輸入想要看到的固定折數商品，分別是沒怎麼打折、九折以下、八折以下和七折以下。

5.2 利用 `if` 和 `elif` 判斷，列印出使用者所選定的固定折數商品。

5.3 使用者可從商品清單中找出自己認為划算的商品，接著貼回 `lativ` 網站的搜尋框，便能快速的找到並購買商品了。

三、成果畫面和實作影片連結

【實作影片解說連結】

<https://drive.google.com/file/d/1X22UzYxTQgkeqYUWYdfJS9BCA6B4LTvJ/view?usp=sharing>

【成果畫面—使用者輸入框和特定商品資訊】

請輸入1, 0.9, 0.8 or 0.7: 0.7

[7折商品]

牛仔雙口袋襯衫-男 (折扣價: \$ 399) 0.676 折
燈芯絨寬版外套-童 (折扣價: \$ 399) 0.676 折
磨毛棉質飛行外套-男 (折扣價: \$ 399) 0.676 折
寬版半開襟連帽外套-男 (折扣價: \$ 399) 0.676 折
牛仔雙口袋長袖襯衫-男 (折扣價: \$ 399) 0.676 折
牛仔條紋襯衫-男 (折扣價: \$ 399) 0.676 折
BARBAPAPA Fleece寬鬆上衣-Baby (折扣價: \$ 168) 0.675 折
哆啦A夢刷毛圓領衫-03-女 (折扣價: \$ 330) 0.673 折
KODAK刷毛圓領衫-01-女 (折扣價: \$ 330) 0.673 折
天竺鼠車車刷毛圓領衫-02-女 (折扣價: \$ 330) 0.673 折
牛仔襯衫式洋裝-童 (折扣價: \$ 330) 0.673 折
漫威系列毛圈連帽外套-童 (折扣價: \$ 330) 0.673 折
Superman毛圈連帽外套-童 (折扣價: \$ 330) 0.673 折
Batman毛圈連帽外套-童 (折扣價: \$ 330) 0.673 折
棉質低腰條紋平口內褲-男 (折扣價: \$ 99) 0.664 折
高腰長褲-Baby (折扣價: \$ 99) 0.664 折
棉質條紋三角褲-男 (折扣價: \$ 99) 0.664 折
BARBAPAPA印花T恤-05-Baby (折扣價: \$ 99) 0.664 折
磨毛羅紋圓領長袖T恤-Baby (折扣價: \$ 99) 0.664 折
棉質三角褲-男 (折扣價: \$ 99) 0.664 折
棉質低腰平口內褲-男 (折扣價: \$ 99) 0.664 折
棉質平口內褲-男 (折扣價: \$ 99) 0.664 折
棉質條紋平口內褲-男 (折扣價: \$ 99) 0.664 折
BARBAPAPA高腰長褲-Baby (折扣價: \$ 99) 0.664 折
純棉圓領T恤-童 (折扣價: \$ 99) 0.664 折
BARBAPAPA印花T恤-12-Baby (折扣價: \$ 99) 0.664 折
彈力九分內搭褲-Baby (折扣價: \$ 99) 0.664 折
磨毛羅紋條紋圓領T恤-Baby (折扣價: \$ 99) 0.664 折
商務防皺長袖襯衫-男 (折扣價: \$ 390) 0.661 折
西裝喇叭褲-女 (折扣價: \$ 390) 0.661 折
綁帶寬褲-女 (折扣價: \$ 390) 0.661 折
BARBAPAPA條紋印花T恤-10-Baby (折扣價: \$ 129) 0.648 折
BARBAPAPA Fleece家居套裝-Baby (折扣價: \$ 249) 0.624 折
BARBAPAPA Fleece家居套裝-童 (折扣價: \$ 249) 0.624 折

四、參考資料

【書籍】

- 1、The Python Workshop 跟著實例有效學習 Python (Andrew Bird, Dr Lau Cher Han, Mario Corchero Jimenez, Graham Lee, Cory Wade 著, 2020 年)
- 2、Python 大數據特訓班 = Python for big data training course (文淵閣工作室編著, 2018 年)
- 3、Python x Excel VBA x JavaScript: 網路爬蟲 x 實戰演練 (廖敏宏著, 2021 年)
- 4、Python 從初學到生活應用超實務: 讓 Python 幫你處理日常生活與工作中繁瑣重複的工作 (陳會安著, 2020 年)

【網路參考資料】

Find_elements:

<https://selenium-python-zh.readthedocs.io/en/latest/locating-elements>

.html

ActionChains—

https://python-selenium-zh.readthedocs.io/zh_CN/latest/7.2%E8%A1%8C%E4%B8%BA%E9%93%BE/

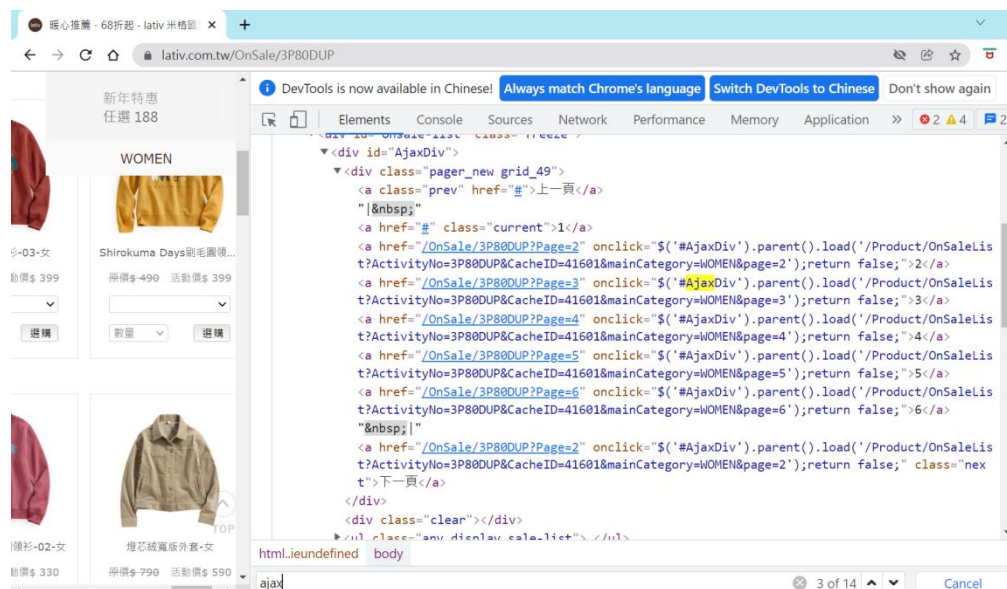
Ajax—<https://tw.alphacamp.co/blog/ajax-asynchronous-request>

Pandas.DataFrame Document—

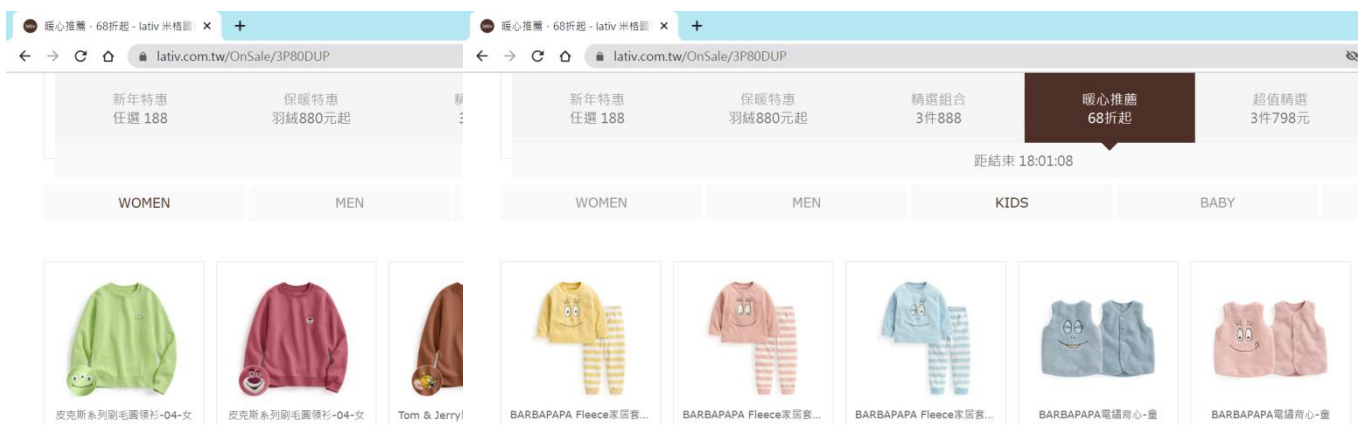
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>

五、附錄：

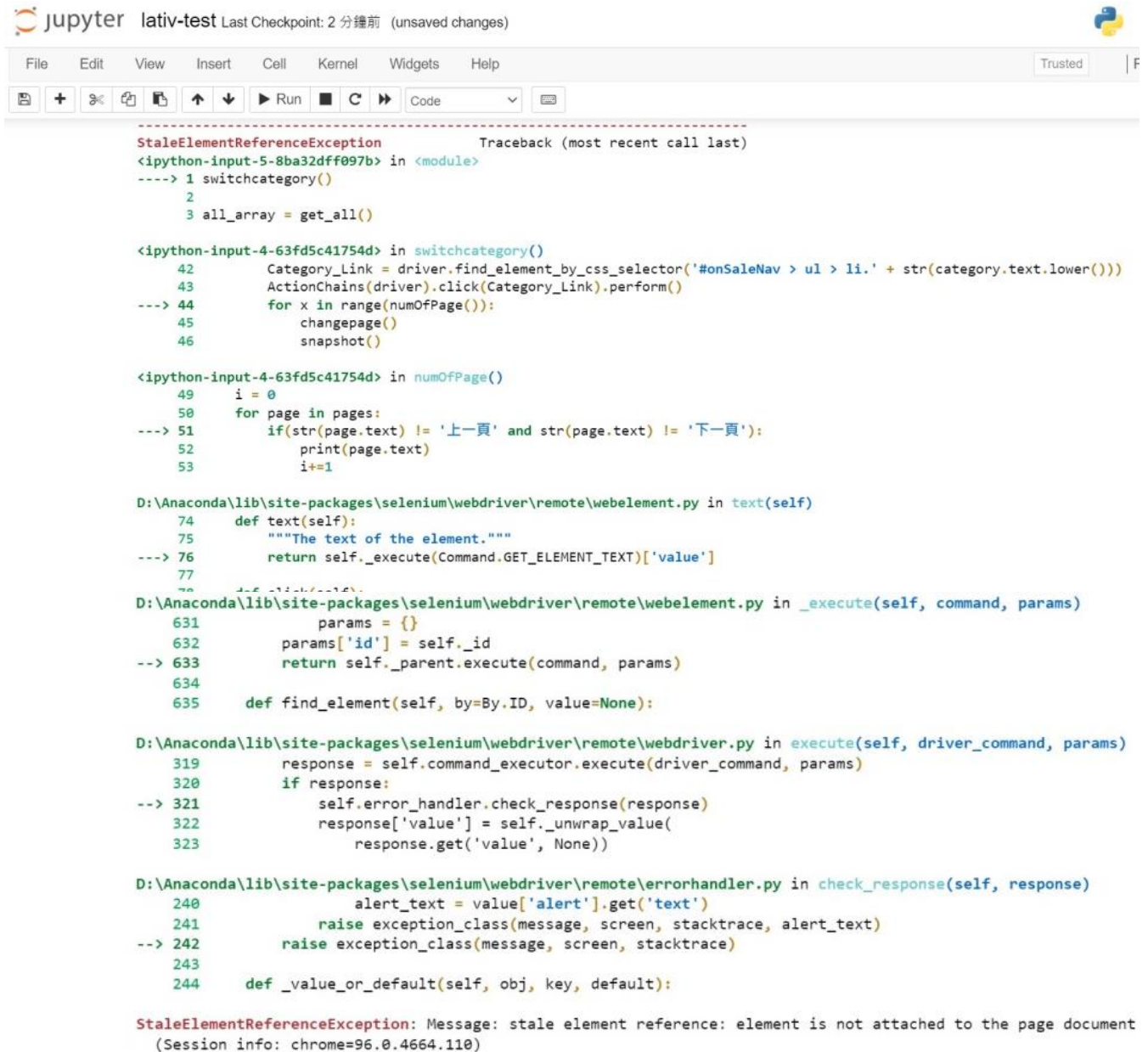
【圖一：Ajax】



【圖二、三：不同分類網址對比】



【圖五、六：報錯訊息】



```
StaleElementReferenceException                                Traceback (most recent call last)
<ipython-input-5-8ba32dff097b> in <module>
----> 1 switchcategory()
      2
      3 all_array = get_all()

<ipython-input-4-63fd5c41754d> in switchcategory()
     42     Category_Link = driver.find_element_by_css_selector('#onSaleNav > ul > li.' + str(category.text.lower()))
     43     ActionChains(driver).click(Category_Link).perform()
--> 44     for x in range(numOfPage()):
     45         changepage()
     46         snapshot()

<ipython-input-4-63fd5c41754d> in numOfPage()
     49     i = 0
     50     for page in pages:
--> 51         if(str(page.text) != '上一頁' and str(page.text) != '下一頁'):
     52             print(page.text)
     53             i+=1

D:\Anaconda\lib\site-packages\selenium\webdriver\remote\webelement.py in text(self)
     74     def text(self):
     75         """The text of the element."""
--> 76         return self._execute(Command.GET_ELEMENT_TEXT)['value']
     77
     78     def _execute(self, command, params):

D:\Anaconda\lib\site-packages\selenium\webdriver\remote\webelement.py in _execute(self, command, params)
    631         params = {}
    632         params['id'] = self._id
--> 633         return self._parent.execute(command, params)
    634
    635     def find_element(self, by=By.ID, value=None):

D:\Anaconda\lib\site-packages\selenium\webdriver\remote\webdriver.py in execute(self, driver_command, params)
    319         response = self.command_executor.execute(driver_command, params)
    320         if response:
--> 321             self.error_handler.check_response(response)
    322             response['value'] = self.unwrap_value(
    323                 response.get('value', None))

D:\Anaconda\lib\site-packages\selenium\webdriver\remote\errorhandler.py in check_response(self, response)
    240         alert_text = value['alert'].get('text')
    241         raise exception_class(message, screen, stacktrace, alert_text)
--> 242         raise exception_class(message, screen, stacktrace)
    243
    244     def _value_or_default(self, obj, key, default):

StaleElementReferenceException: Message: stale element reference: element is not attached to the page document
(Session info: chrome=96.0.4664.110)
```

六、程式碼

Discount Expert (以 Lativ 暖心推薦為例)

<1>定義函數和 import 套件

import requests

from selenium import webdriver

from selenium.webdriver import ActionChains

```
from bs4 import BeautifulSoup

import pandas as pd

from time import sleep

# 定義函式

def snapshot():

    # 擷取單頁所有商品資訊

    soup = BeautifulSoup(driver.page_source, 'lxml')

    name = soup.select("h3[class='any_display_name']")

    ad_price = soup.select("span[class='currency symbol pricing']")

    ori_price = soup.select("span[class='currency noAboutSymbol']")

    for t in name:

        tag.append(t.text.strip())

    for p in ori_price:

        ori_pricing.append('$ ' + p.text.strip('$\xa0'))

    for a in ad_price:

        ad_pricing.append('$ ' + a.text.strip('$\xa0'))

    return tag, ori_pricing, ad_pricing

def get_all():

    # 獲取全部商品資訊

    all.append(tag)

    all.append(ori_pricing)

    all.append(ad_pricing)

    print(len(all[0]))

    return all

def changepage():

    # 換頁
```

```

        Next_Link = driver.find_element_by_css_selector('#AjaxDiv >
div:nth-child(1) > a.next')

        ActionChains(driver).click(Next_Link).perform()

def switchcategory():
    # 換類別

    categories = soup.find_all('li', current = 'False')

    for category in categories:

        Category_Link = driver.find_element_by_css_selector('#onSaleNav >
ul > li.' + str(category.text.lower()))

        ActionChains(driver).click(Category_Link).perform()

        sleep(2)    #Sleep 可更保證在擷取商品時，網頁是有完整讀取完畢的
        snapshot()

        for x in range(numOfPage()-1):

            changepage()

            sleep(1)

            snapshot()

def numOfPage():
    # 獲取當下頁面的頁數

    pages = driver.find_elements_by_xpath('//*[@id="AjaxDiv"]/div[1]/child::*')

    i = 0

    for page in pages:

        i+=1

    i-=2

    return i

res = requests.get('https://www.lativ.com.tw/OnSale/3P80DUP')

soup = BeautifulSoup(res.text, 'html.parser')

```

<2>爬取各頁商品資訊

```
driver = webdriver.Chrome()

driver.get('https://www.lativ.com.tw/OnSale/3P80DUP')

tag = []

ori_pricing = []

ad_pricing = []

all = []

#Women

snapshot()

for x in range(numOfPage()-1):

    changepage()

    sleep(1) #Sleep 可更保證在擷取商品時，網頁是有完整讀取完畢的

    snapshot()

#Switch to Other Category

switchcategory()

all_array = []

all_array = get_all()

discount = []
```

<3>計算折扣數(ex. 0.812 折、0.953 折)

```
#型別轉換 list->str->int

for i in range(len(all_array[0])):

    num1 = [str(temp)for temp in all_array[2][i].split() if temp.isdigit()]

    num1_string = "".join(num1)

    num1_int = int(num1_string)

    num2 = [str(temp)for temp in all_array[1][i].split() if temp.isdigit()]
```

```

num2_string = "".join(num2)

num2_int = int(num2_string)

discount.append(str(round((num1_int/num2_int),3)))

all_array.append(discount)

print(all_array)

```

<4>製成表格、刪去重複性資料、重新排列

```

df = pd.DataFrame(all_array, index=['衣服名稱', '原價格', '折扣價', '幾折'])

df = df.T

df.drop_duplicates(subset = '衣服名稱', keep='first', inplace=True)

#刪除重複性資料

df = df.sort_values(by='幾折', ascending=False)

#降序排列

df.reset_index(inplace=True, drop=True)

print(df)

```

<5>使用者輸入框和獲取資訊

```

a = float(input('請輸入 1, 0.9, 0.8 or 0.7: '))

if (a==1):

    print("[跟原價差不多]")

    for i in range(len(df)):

        if(float(df.at[i, '幾折']) > 0.9):

            print("\t"+df.at[i, '衣服名稱'], '(折扣價:', df.at[i, '折扣價'], ")", df.at[i, '幾折'], "折")

elif(a==0.9):

    print("[9 折商品]")

```

```

for i in range(len(df)):
    if(0.9 > float(df.at[i, '幾折']) > 0.8):
        print("\t"+df.at[i, '衣服名稱'], '(折扣價:', df.at[i, '折扣價'], ")", df.at[i, '
幾折'], "折")

elif(a==0.8):
    for i in range(len(df)):
        if(0.8 > float(df.at[i, '幾折']) > 0.7):
            print("\t"+df.at[i, '衣服名稱'], '(折扣價:', df.at[i, '折扣價'], ")", df.at[i, '
幾折'], "折")

elif(a==0.7):
    print("[7 折商品]")
    for i in range(len(df)):
        if(float(df.at[i, '幾折']) < 0.7):
            print("\t"+df.at[i, '衣服名稱'], '(折扣價:', df.at[i, '折扣價'], ")", df.at[i, '
幾折'], "折")

elif():
    print("Wrong Input!!!")

```