

# 大數據分析

## 作業一

(Programming Assignment1)

410921202 資工四 林芷萱

## 一、 The problem description

此次作業的要求為使用 python 寫一個程式爬取 [https://www.imdb.com/chart/top/?ref=mv\\_mv\\_250](https://www.imdb.com/chart/top/?ref=mv_mv_250) 的資料，分為兩個部分；Part A 為爬取前 250 個電影的 rank、name、year、duration、rating、rate 並存入 csv 檔，檔名為 films250.csv；Part B 為將前 250 個電影的 duration 全部加起來。

## 二、 Highlight of the way you write the program

### Part A:

這部分的重點在於怎麼處理 missing data，我先爬取 year、duration、rating，然後先將每一個都設為 None，接著再判斷裡面的值。duration 的資料會出現 h 或 m，所以如果有 h 或 m 的資料就是 duration；觀察 year 跟 rating 兩個資料的差異會發現 rating 的資料是英文、數字、符號混和，而 year 全部由數字組成，因此只要資料只有數字就是年份，否則則為 rating。

接著再將三個值依序放入對應的位置，如果沒有這個值，他就是 missing data。

```
# 處理year, duration, rating
metadata_element = movie.find(class_='sc-6fa21551-7 jLjTzn cli-title-metadata')
#print(metadata_items)
metadata_items = metadata_element.find_all(class_='sc-6fa21551-8 bnyjtw cli-title-metadata-item')

year = None
duration = None
rating = None

for item in metadata_items:
    text = item.text.strip()
    if 'h' in text or 'm' in text: # 如果有h或m就代表是duration
        duration = text
    elif text.isdigit(): # 如果只有數字就是年份
        year = text
    else:
        rating = text

# 處理missing data
movie_year = year if year else "missing data"
movie_duration = duration if duration else "missing data"
movie_rating = rating if rating else "missing data"
```

## Part B:

這部分我的作法是分別把 h 跟 m 前面的數字加起來，如下：

```
for durations in all_durations:
    parts = durations.split()
    for part in parts:
        if part.endswith("h"):
            hours = int(part[:-1]) # 取得小時
            total_hours += hours
        elif part.endswith("m"):
            minutes = int(part[:-1]) # 取得分鐘
            total_minutes += minutes
```

接著再做進位：

```
# 把分鐘換成小時，有超過進位
if total_minutes >= 60:
    additional_hours = total_minutes // 60
    total_hours += additional_hours
    total_minutes %= 60
```

## 三、 The program listing

```
In [27]: import requests
        from bs4 import BeautifulSoup as BF
        import csv

In [28]: url = 'https://www.imdb.com/chart/top?ref_=nv_mv_250'
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.0.0 Safari/537.36',
            'Cookie': 'language=en_US',
            'Accept-Language': 'en-US,en;q=0.5'
        }

        html = requests.get(url, headers=headers)

        sp = BF(html.text, 'html.parser')
```

## Part A

```
In [32]: movie_item = sp.find_all(class_='ipc-metadata-list-summary-item__c')

In [36]: all_durations = []
with open('films250.csv', 'w', newline='', encoding='utf-8') as csvfile:
    fieldnames = ['ranking', 'name', 'year', 'duration', 'rating', 'rate']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames, quoting=csv.QUOTE_ALL)
    writer.writeheader()

    for movie in movie_item:
        # 處理rank與name
        name = movie.find(class_='ipc-title__text').text.strip()
        parts = name.split('.', 1)

        if len(parts) == 2:
            movie_rank = parts[0]
            movie_name = parts[1] # 獲取電影名部分

        # 處理year, duration, rating
        metadata_element = movie.find(class_='sc-6fa21551-7 jlJ7zn cli-title-metadata')
        #print(metadata_items)
        metadata_items = metadata_element.find_all(class_='sc-6fa21551-8 bnyjtW cli-title-metadata-item')

        year = None
        duration = None
        rating = None

        for item in metadata_items:
            text = item.text.strip()
            if 'h' in text or 'm' in text: # 如果有h或m就代表是duration
                duration = text
            elif text.isdigit(): # 如果只有數字就是年份
                year = text
            else:
                rating = text

        # 處理missing data
        movie_year = year if year else "missing data"
        movie_duration = duration if duration else "missing data"
        movie_rating = rating if rating else "missing data"
        # 把時長加進去陣列, part 0計算用
        all_durations.append(movie_duration)

        # 處理rate
        rate_element = movie.find(class_='ipc-rating-star--base')
        rate = rate_element.get_text(strip=True)
        rate_split = rate.split('.')

        movie_rate = rate_split[0]
        writer.writerow({'ranking': movie_rank, 'name': movie_name, 'year': movie_year,
                        'duration': movie_duration, 'rating': movie_rating, 'rate': movie_rate})
```

## Part B

```
In [39]: total_hours = 0
total_minutes = 0

for durations in all_durations:
    parts = durations.split()
    for part in parts:
        if part.endswith("h"):
            hours = int(part[:-1]) # 取得小時
            total_hours += hours
        elif part.endswith("m"):
            minutes = int(part[:-1]) # 取得分鐘
            total_minutes += minutes

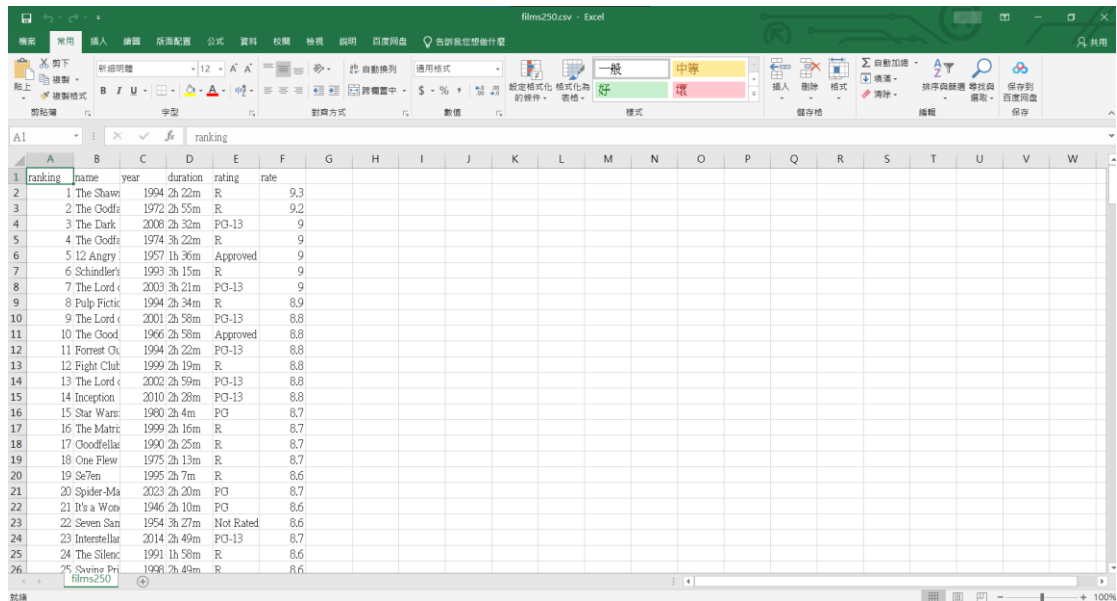
# 把分鐘換成小時, 有超過進位
if total_minutes >= 60:
    additional_hours = total_minutes // 60
    total_hours += additional_hours
    total_minutes %= 60

count_duration = f"{total_hours} hours {total_minutes} minutes"
print(count_duration)

537 hours 59 minutes
```

## 四、 Test run results

### Part A:



ranking	name	year	duration	rating	rate
1	The Shawshank Redemption	1994	2h 22m	R	9.3
2	The Godfather	1972	2h 55m	R	9.2
3	The Dark Knight	2008	2h 32m	PG-13	9
4	The Godfather Part II	1974	3h 22m	R	9
5	12 Angry Men	1957	1h 36m	Approved	9
6	Schindler's List	1993	3h 15m	R	9
7	The Lord of the Rings: The Fellowship of the Ring	2003	3h 21m	PG-13	9
8	Pulp Fiction	1994	2h 34m	R	8.9
9	The Lord of the Rings: The Two Towers	2001	2h 58m	PG-13	8.8
10	The Good, the Bad and the Ugly	1966	2h 58m	Approved	8.8
11	Forrest Gump	1994	2h 22m	PG-13	8.8
12	Fight Club	1999	2h 19m	R	8.8
13	The Lord of the Rings: The Return of the King	2002	2h 59m	PG-13	8.8
14	Inception	2010	2h 28m	PG-13	8.8
15	Star Wars: The Force Awakens	1980	2h 4m	PG	8.7
16	The Matrix	1999	2h 16m	R	8.7
17	Goodfellas	1990	2h 25m	R	8.7
18	One Flew Over the Cuckoo's Nest	1975	2h 13m	R	8.7
19	Se7en	1995	2h 7m	R	8.6
20	Spider-Man	2002	2h 20m	PG	8.7
21	It's a Wonderful Life	1946	2h 10m	PG	8.6
22	Seven Years in Tibet	1997	2h 27m	Not Rated	8.6
23	Interstellar	2014	2h 49m	PG-13	8.7
24	The Silence of the Lambs	1991	1h 58m	R	8.6
25	Saving Private Ryan	1998	2h 49m	R	8.6

### Part B:

537 hours 59 minutes

## 五、 Discussion

在這次的作業當中我學習到了如何使用爬蟲爬取需要的資料以及要怎麼將檔案存進 csv 檔，在進行的過程中遇到比較大的問題是，我發現在爬取時資料會自動爬取成中文，這會導致 csv 檔打開來是亂碼，儘管我將 encoding 設為 utf-8 也一樣，最後我在 headers 加入了這兩行程式碼：

```
'Cookie': 'language=en_US',
```

```
'Accept-Language': 'en-US,en;q=0.5'
```

這使得爬蟲爬取該網頁時會強制使用英文，就沒有出現爬取到中文資料的問題了。