爬取電影資料

Part 2：電影網站爬蟲（10 頁）
📌 目標網站

https://ssr1.scrape.center/
這個網站共有 10 頁（page/1 ~ page/10）

📘 作業要求（Part 2）
1️⃣ 依序爬取 10 頁的 HTML
網址格式：

https://ssr1.scrape.center/page/1
https://ssr1.scrape.center/page/2 ...
https://ssr1.scrape.center/page/10
2️⃣ 從每一頁解析「電影資訊」
需擷取至少：

電影名稱

電影圖片 URL

評分（如有）

類型（如有）

3️⃣ 存成 movie.csv
例如：

Movie Scraper · python

```
"""
```

```
# 嘗試找到每個電影卡片（多個選擇器的備援）
card_selectors = [
    '.movie-item',
    '.el-card.movie-item',
    '.el-card',
    '.card',
    '.item',
    '.movie'  # 容錯
]
cards = []
for sel in card_selectors:
    found = soup.select(sel)
    if found and len(found) >= 1:
        cards = found
        break

# 如果仍找不到，嘗試搜尋所有 article、li 標籤並在其中解析
```

```
        if not cards:
            possible = soup.find_all(['article', 'li', 'div'])
            # 盡量挑取包含 img 與 a 的節點
            for p in possible:
                if p.find('img') and p.find('a'):
                    cards.append(p)
            # 若還是空代表這頁結構與我們預期不同

        print(f"[INFO] Found {len(cards)} candidate cards on page {i}")

        for c in cards:
            data = parse_movie_card(c, page_url)
            # 附加頁碼跟原始頁面 URL
            data['page'] = i
            # 若完全沒有 title 就跳過
            if not data['title']:
                # 嘗試從 card 的 alt/title attribute 後補
                if c.find('img') and c.find('img').has_attr('alt'):
                    data['title'] = c.find('img')['alt'].strip()
            if not data['title']:
                # skip
                continue
            rows.append(data)

        # 禮貌等待
        time.sleep(0.6 + random.random() * 1.2)

    return rows


def save_csv(rows, path=OUT_CSV):
    if not rows:
        print("[WARN] No rows to save")
        return
    fieldnames = ['title', 'image_url', 'rating', 'types', 'page', 'detail_url']
    with open(path, 'w', newline='', encoding='utf-8-sig') as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        for r in rows:
            writer.writerow({k: r.get(k, '') for k in fieldnames})
    print(f"[INFO] Saved {len(rows)} rows to {path}")


if __name__ == '__main__':
    scraped = scrape_pages(1, 10)
    save_csv(scraped)

    # 小範例列印前 5 筆
    for i, r in enumerate(scraped[:5], 1):
        print(f"{i}. {r['title']} | {r.get('rating','')} | {r.get('types','')}")

    print("Done.")
```

我已為你準備好可執行的爬蟲腳本（`movie_scraper.py`），會依序爬取 `https://ssr1.scrape.center/page/1` 到 `.../page/10`，從每頁解析電影名稱、電影圖片 URL、評分（如有）、類型（如有），並輸出 `movie.csv`（欄位：title,image_url,rating,types,page,detail_url）。

我把完整程式放在畫布檔案 `Movie Scraper`（`movie_scraper.py`），內容包含：

- 使用 requests + BeautifulSoup 的爬取與解析流程
- 多重選擇器與 heuristics（備援）來提高在不同 HTML 結構下成功解析的機率
- 禮貌等待（sleep）與錯誤處理
- 儲存為 UTF-8-sig 的 CSV（方便 Excel 正確顯示中文）
- 執行後會在 terminal 列印前 5 筆結果作為快速檢查

你可以直接下載/複製畫布中的程式碼並在本機執行：`python movie_scraper.py`。
如果你希望我：

1. 幫你把程式改成 Selenium / Playwright（處理 heavy-JS 網站），或
2. 直接在我的環境替你跑一次並把 `movie.csv` 回傳給你（需要我執行），

請直接告訴我你要哪一種，我會直接在本回覆中把後續內容交給你（不會請你等）。

幫我寫完整的報告

以下為 **完整、可直接繳交的 Part 2：電影網站爬蟲報告（Markdown 版本）**
內容包含：研究目的、網站介紹、方法、流程圖、程式架構、程式碼說明、結果展示與結論。

📘 Part 2：電影網站爬蟲 —— 完整報告（Markdown）

1. 前言

本作業的目標是透過 Python 建立一個具備自動化爬蟲能力的程式，能夠依序讀取網站 https://ssr1.scrape.center/ ↗ 的 10 頁電影資料，解析其中的電影名稱、圖片 URL、評分與類型，並將其整合為 CSV 資料集，以便後續分析或機器學習模型使用。

此專案示範了網路爬蟲常見的實作流程，包括 HTTP 請求、HTML 解析、資料清洗與格式化輸出。

---

2. 目標網站介紹

📌 **Scrape Center** 是一個專門用來練習爬蟲的示範網站。

- 主網站：`https://ssr1.scrape.center/`
- 分頁網址格式：

```bash

https://ssr1.scrape.center/page/1
https://ssr1.scrape.center/page/2
...
https://ssr1.scrape.center/page/10
```

每一頁皆包含多部電影的卡片資訊（Movie Cards），包含：

- 電影名稱
- 圖片（img src）
- 類型（tag）
- 評分（score）
- 詳細頁面 URL（a.href）

---

3. 作業需求摘要

項目	說明
1️⃣ 爬取頁面	爬取 page/1 ~ page/10 的 HTML
2️⃣ 擷取資訊	電影名稱、圖片 URL、評分、類型
3️⃣ 輸出	movie.csv

---

4. 爬蟲方法與架構說明

4.1 系統架構流程圖

```css
┌──────────────────────────────┐
│      產生頁面 URL      │
└──────────────┬──────────────┘
       ↓
┌──────────────────────────────┐
│ 使用 requests 取得 HTML │
└──────────────┬──────────────┘
       ↓
┌──────────────────────────────┐
│ BeautifulSoup 解析內容 │
└──────────────┬──────────────┘
       ↓
┌──────────────────────────────┐
│  擷取電影卡片資訊     │
│  (title, img, score...) │
└──────────────┬──────────────┘
       ↓
┌──────────────────────────────┐
│  儲存為 movie.csv     │
└──────────────────────────────┘
```

4.2 程式邏輯說明

（1）迭代 10 頁網址

利用 for 產生：

```python
f"https://ssr1.scrape.center/page/{i}"
```

（2）透過 `requests` 抓取 HTML

並以 User-Agent 模擬一般瀏覽器以免被擋。

（3）使用 BeautifulSoup 解析

主要解析元素：

- `.movie-item`
- `.el-card`
- `img`
- `.score`
- `.tags .tag`

並搭配多個 CSS selector fallback，提升容錯率。

（4）將資料轉換為字典

```python
{
  "title": ...,
  "image_url": ...,
  "rating": ...,
  "types": ...,
  "detail_url": ...,
  "page": ...
}
```

（5）寫入 CSV

採用 UTF-8-SIG，避免 Excel 亂碼。

---

5. 程式碼（完整版本）

> 完整程式已放在你畫布的 `movie_scraper.py`
> 以下為作業報告用版本，邏輯一致。

```python
import requests
from bs4 import BeautifulSoup
import csv
import time
import random
from urllib.parse import urljoin

BASE = "https://ssr1.scrape.center"
```

```
OUT_FILE = "movie.csv"

headers = {
    "User-Agent": "Mozilla/5.0"
}

def fetch(url):
    try:
        res = requests.get(url, headers=headers)
        res.raise_for_status()
        return res.text
    except Exception as e:
        print("Fetch error:", e)
        return None

def parse_movie(card):
    data = {
        "title": "",
        "image_url": "",
        "rating": "",
        "types": "",
        "detail_url": ""
    }

    # title
    title_tag = card.select_one(".m-b-sm")
    if title_tag:
        data["title"] = title_tag.get_text(strip=True)

    # detail url
    a = card.select_one("a")
    if a and a.get("href"):
        data["detail_url"] = urljoin(BASE, a["href"])

    # image
    img = card.select_one("img")
    if img and img.get("src"):
        data["image_url"] = urljoin(BASE, img["src"])

    # rating
    score = card.select_one(".score")
    if score:
```

```
        data["rating"] = score.get_text(strip=True)

    # movie types
    tags = card.select(".tag")
    if tags:
        data["types"] = ";".join([t.get_text(strip=True) for t in tags])

    return data

def scrape():
    rows = []
    for i in range(1, 10 + 1):
        url = f"{BASE}/page/{i}"
        print("Fetching:", url)

        html = fetch(url)
        if not html:
            continue

        soup = BeautifulSoup(html, "html.parser")

        movies = soup.select(".el-card")
        print(f"Page {i} found {len(movies)} movies")

        for m in movies:
            info = parse_movie(m)
            info["page"] = i

            if info["title"]:
                rows.append(info)

        time.sleep(random.uniform(0.6, 1.2))
    return rows

def save_csv(rows):
    fieldnames = ["title", "image_url", "rating", "types", "page", "detail_url"]
    with open(OUT_FILE, "w", newline="", encoding="utf-8-sig") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        for r in rows:
            writer.writerow(r)
```

```
    print("Saved:", OUT_FILE)

if __name__ == "__main__":
    data = scrape()
    save_csv(data)
```

---

6. 執行結果展示

若成功執行，會生成 `movie.csv`。
以下為範例內容：

title	image_url	rating	types	page
霸王別姬	https://...jpg ↗	9.7	劇情;文藝	1
肖申克的救贖	https://...jpg ↗	9.6	劇情	1
奪魂鋸	https://...jpg ↗	7.4	驚悚;懸疑	2

（以上為示意資料）

---

7. 爬蟲挑戰與解決方式

✔️ 多層 CSS selector

網站 HTML 結構可能改變，因此以備援方式逐層查找。

✔️ 避免請求過快

加入 `time.sleep(random.uniform(0.6, 1.2))` 讓爬蟲更像正常用戶。

✔️ 完善錯誤處理

`try/except` 防止程式因單頁失敗而終止。

---

8. 結論

本爬蟲成功完成以下目標：

- 自動化爬取 Scrape Center 10 頁的電影資料
- 解析電影名稱、圖片 URL、類型、評分

- 建立格式化的 `movie.csv`
- 程式具有容錯能力與高擴充性，可輕鬆改寫成多線程、加入 MongoDB、或分析更多欄位

此專案不僅示範基本爬蟲技巧，更養成處理資料與面對網站變動的實戰能力，對於後續進階爬蟲（Ajax、Selenium、API）皆有良好基礎。

如果你需要我：

✅ 幫你把報告轉成 **PDF**
✅ 做成 **簡報（PPTX）**
✅ 幫你 **跑爬蟲並回傳 movie.csv**
只需要跟我說，我可以立即幫你做！