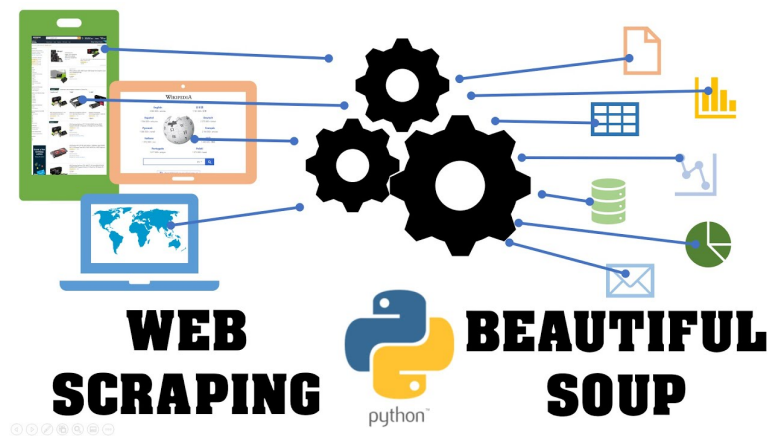

Web Scrapping en Python con beautiful soup



Realizado por:

FRANCISCO JOSÉ RODRÍGUEZ CEREZO

Índice

1. Contexto:	2
2. Descripción del <i>dataset</i> :	2
3. Contenido del <i>dataset</i> :	2
4. Código desarrollado para llevar a cabo el <i>web scrapping</i> :	3

1. Contexto:

Nos disponemos a realizar un breve proyecto de *web scrapping* en el que vamos a extraer información del portal *web* de *Filmaffinity* ya que el mismo carece de *API*. De este modo obtendremos un *dataset* que en el futuro podría tener utilidad para otro proyecto o para otro equipo de personas que, por ejemplo, pretenda hacer estadística sobre tendencias en las carteleras de cine o intente relacionar alguna variables con el éxito de los *films* en taquilla.

2. Descripción del *dataset*:

En este proyecto recopilamos los datos de aquellas películas en las categorías *En cartelera* y *Ya para alquilar* de *Filmaffinity*, lo cual constituye, como ya hemos mencionado, las películas más recientes a fecha del 16 de abril de 2023 (fecha en la que ejecutamos el código para realizar el *web scrapping*).

3. Contenido del *dataset*:

- **Título:** Título de la película en España.
- **Título original:** Título original de la película sin traducir.
- **Año:** Año de estreno de la película.
- **Duración:** Duración de la película en minutos.
- **País:** País donde fue producida la película.
- **Dirección:** Directores/as de la película.
- **Guión:** Guionistas de la película.
- **Música:** Compositores de la banda sonora de la película.
- **Fotografía:** Director/a del apartado cinematográfico de la película.
- **Reparto:** Nombre de los principales actores de la trama de la película.
- **Compañías:** Compañías productoras y distribuidoras de la película.
- **Género:** Género/s de la película.
- **Sinopsis:** Breve descripción de la trama argumental de la película.
- **Posición_ranking:** Posición en el *ranking* en base a la puntuación que tiene la película en *Filmaffinity*.
- **Número_de_votos:** Número de veces que se ha votado por dicha película en *Filmaffinity*.
- **Calificación_media:** Puntuación media obtenida a través del promedio de los votos de puntuación que los usuarios de *Filmaffinity* han usado para calificar la película.
- **Link:** *URL* que redirecciona a la página de la película dentro del portal *web* de *Filmaffinity*.

4. Código desarrollado para llevar a cabo el *web scrapping*:

- **settings.py:** En este archivo definimos las variables que creamos convenientes e importamos las librerías necesarias (en especial *requests* y *BeautifulSoup*).

```
source > settings.py > { } pd
1  """In this file we import all the necessary libraries and variables."""
2  # Libraries..
3  import os
4  import time
5  import requests
6  import pandas as pd
7  from bs4 import BeautifulSoup
8  from pprint import pprint
9  # Variables..
10 files_web = "https://www.filmaffinity.com/es/main.html"
11 url_bin = "http://httpbin.org/headers"
12 new_headers = {
13     "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
14     "Accept-Encoding": "gzip, deflate",
15     "Accept-Language": "es-ES;q=0.9,en;q=0.8",
16     "Upgrade-Insecure-Requests": "1",
17     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
18 }
```

Figura 1: *Script settings.py.*

- **functions.py:** En el que definimos las funciones empleadas en el proceso de *web scrapping*, este archivo será la parte clave de nuestro proyecto ya que aquí se implementa la mayoría del código.

```
source > functions.py > ...
1  """In this file we define all the necessary functions for our web-scrapping project."""
2  from settings import *
3
4
5  def get_movie_info(bs_object: BeautifulSoup, link: str) -> pd.Series:
6      """This function collects all the information about a movie's file given in filmaffinity.
7
8      :param bs_object: HTML parser object given by BeautifulSoup library
9
10     :param link: String containing the link to the movie card on filmaffinity
11
12     ...
13     :return: Panda's serie with all the info on filmaffinity about a given movie."""
14     # We extract all "dt" tags from the HTML code of the movie page.
15     dt_results = bs_object.find_all("dt")
16     # We create a panda's serie to gather all the information about the movie.
17     movie = pd.Series(dtype="object")
18     # We proceed to look for the info.
19     movie["Titulo"] = bs_object.h1.span.text
20     movie["Titulo Original"] = bs_object.body.find(text="Título original").parent.findNext("dd").text
21     movie["Año"] = bs_object.body.find(text="Año").parent.findNext("dd").text
22     movie["Duración"] = bs_object.body.find(text="Duración").parent.findNext("dd").text
23     movie["País"] = bs_object.body.find(text="País").parent.findNext("dd").text
24     movie["Dirección"] = bs_object.find(itemprop="director").parent.text
25     # Some movies doesn't have the info available for plot, music and/or photography.
26     try:
27         script = bs_object.body.find(text="Guión").parent.findNext("dd").text
28     except:
29         script = "no disponible"
30     movie["Guión"] = script
31     try:
32         music = bs_object.body.find(text="Música").parent.findNext("dd").text
33     except:
34         music = "no disponible"
35     movie["Música"] = music
36     try:
37         photography = bs_object.body.find(text="Fotografía").parent.findNext("dd").text
38     except:
39         photography = "no disponible"
40     movie["Fotografía"] = photography
41     movie["Reparto"] = bs_object.body.find(text="Reparto").parent.findNext("dd").text
```

Figura 2: *Script functions.py (1).*

```
42     try:
43         companies = bs_object.body.find(text="Compañías").parent.findNext("dd").text
44     except:
45         companies = "no disponible"
46     movie["Compañías"] = companies
47     movie["Género"] = bs_object.body.find(text="Género").parent.findNext("dd").text
48     movie["Sinopsis"] = bs_object.body.find(text="Sinopsis").parent.findNext("dd").text
49     # Some movies doesn't have the info available for ranking position, average rating and/or rating count.
50     try:
51         pos_ranking = bs_object.body.find(text="Posición en rankings FA").parent.findNext("dd").text
52     except:
53         pos_ranking = "no disponible"
54     movie["Posición_Ranking"] = pos_ranking
55     try:
56         rating_count = bs_object.find(itemprop="ratingCount").text
57     except:
58         rating_count = "no disponible"
59     movie["Número_de_Votos"] = rating_count
60     try:
61         avg_rating = bs_object.find(id="movie-rat-avg").text
62     except:
63         avg_rating = "no disponible"
64     movie["Calificación_Media"] = avg_rating
65     movie["link"] = link
66     return movie
```

Figura 3: *Script functions.py (2).*

```

70 def get_movies_from_current_page(bs_object: BeautifulSoup) -> list:
71     '''This function collects all the info from movie cards in a given page.
72
73     :param bs_object: HTML parser object given by BeautifulSoup library
74
75     ...
76     :return: List that contains panda's series with the info of the movies on the given page.'''
77     # Empty list to append the links from the page.
78     movies_series = []
79     # We look for movie titles to get the link for their cards with the info.
80     movie_links = bs_object.find_all(class_='movie-title')
81     link_list = [l.a['href'] for l in movie_links]
82     # Iterate along all movies in the page once we got all the movie cards links.
83     for link in link_list:
84         # We iterate along all the movie cards on the current page.
85         page = requests.get(link, headers=new_headers)
86         soup = BeautifulSoup(page.content, "html.parser")
87         # In each iteration we apply "get_movie_info" for the current movie and append
88         # the info in our "movie_series" list.
89         movies_series.append(get_movie_info(soup, link))
90         # We introduce a 2 seconds pause to avoid server saturation and/or detection of our script.
91         time.sleep(2)
92     return movies_series

```

Figura 4: *Script functions.py* (3).

```

95 def scrape_movies(url: str) -> pd.DataFrame:
96     '''This function navigate through the website applying "get_movies_from_current_website" function.
97
98     :param url: String with the url to the website.
99
100    ...
101    :return: Panda's dataframe that contains all the info about all the movies from the website specific sections.'''
102    print("\nScraping in progress. Please wait. This might take a few minutes.")
103    # We get sent the petition to the website using modified header.
104    page = requests.get(url, headers=new_headers)
105    # We parse the HTML body of the main website page.
106    soup = BeautifulSoup(page.content, "html.parser")
107    # We look for the desired sections, in this case, the sections are "Películas en cartelera" and "Ya para alquilar"
108    # which were studied before.
109    url_alquilar = soup.body.find(text='Ya para alquilar').parent['href']
110    url_cartelera = soup.body.find(text='Películas en cartelera').parent['href']
111    # We begin with "Películas en cartelera" which is a simple page.
112    page = requests.get(url_cartelera, headers=new_headers)
113    soup = BeautifulSoup(page.content, "html.parser")
114    # We apply the "get_movies_from_current_page" on this website section.
115    movies_series = get_movies_from_current_page(soup)
116    # The following website section ("Ya para alquilar") requires interactive navigation.
117    page = requests.get(url_alquilar, headers=new_headers)
118    soup = BeautifulSoup(page.content, "html.parser")
119    # We look for the tag "a" which are the "next page" button on the page.
120    next_links = soup.find(class_='pager').find_all('a')
121    # Create a list with all "next pages" from from the section.
122    next_links_list = list(set([l['href'] for l in next_links]))
123    # Reapply the "get_movies_from_current_page" on "movies_series" to expand it
124    # with the info from this new page.
125    movies_series += get_movies_from_current_page(soup)
126    # Iterate over the "next pages" links to navigate through them and gather the info from
127    # the movies on all of them.
128    for next_page in next_links_list:
129        page = requests.get(next_page, headers=new_headers)
130        soup = BeautifulSoup(page.content, "html.parser")
131        movies_series += get_movies_from_current_page(soup)
132
133    # Concatenate the panda's series with the movies info in a panda's dataframe
134    # with the convenient shape.
135    results_df = pd.concat(movies_series, axis=1).transpose()
136    print("\nDONE!")
137    return results_df

```

Figura 5: *Script functions.py* (4).

```

140 def dataset_writer(df: pd.DataFrame, name: str) -> None:
141     '''This function creates a dataset in .csv format with the info from a given panda's dataframe
142     using a desired name on the correct path of this project which is "dataset/name.csv".
143
144     :param df: Panda's dataframe with the necessary info to create the dataset.
145
146     :param name: String with the desired name for the .csv file output.'''
147     # Define the directory for the dataset and create it if necessary.
148     dataset_directory = "./dataset"
149     if not os.path.exists(dataset_directory):
150         os.mkdir(dataset_directory)
151     # Define the full name including the path for our csv dataset file.
152     file_path = os.path.join(dataset_directory, name + ".csv")
153     # Convert the panda's dataframe into .csv file with the path "dataset" of this project
154     # making sure to avoid "UnicodeEncode errors" using "utf-8" since the website is in spanish.
155     df.to_csv(file_path, encoding='utf-8')

```

Figura 6: *Script functions.py* (5).

- **main.py:** Este archivo es el que lleva a cabo el *web scrapping* en sí utilizando las librerías, funciones y variables de los otros dos archivos. Además, en un paso inicial hemos incluido un breve testeo en cuanto a como funciona la modificación de los *headers* y *user-agent* de las peticiones hechas a través de la librería *requests* de *Python*.

```

source > main.py -
1  from settings import *
2  from functions import *
3
4
5  # First we gonna test and show the default header of the requests Python module with an example website which is "url_bin".
6  page = requests.get(url_bin)
7  print("\nHeader and user-agent sent by default with requests Python's library:")
8  pprint(page.text)
9  # Now we gonna change the headers and user-agent to avoid bot detection.
10 print("\nmodified header and user-agent:")
11 page = requests.get(url_bin, headers=new_headers)
12 pprint(page.text)
13
14 # After this brief test, we follow the same strategy with headers and user-agent during web scrapping.
15 df_movies = scrape_movies(films_web)
16 # Finally we create our dataset in csv format.
17 dataset_writer(df_movies, "recent_movies_info")

```

Figura 7: *Script main.py.*

A continuación vamos a destacar algunos aspectos concretos de nuestro código con algo más de detalle:

- La variable *new_headers* en *settings.py* es un diccionario que utilizamos para modificar las cabeceras y *user-agent* de las peticiones al servidor. Esta modificación la realizamos a través del parámetro *headers=new_headers* dentro del método *requests.get(url, headers=new_headers)* que empleamos varias veces en las funciones del archivo *functions.py* para definir los objetos *page* sobre los que “parseamos” el *HTML* del cuerpo de las páginas con *BeautifulSoup*.
- La función más básica de nuestro código es *get_movie_info()* en la que “parseamos” el *HTML* de la página web de una película y almacenamos en una “serie” de *Pandas* todos los datos que nos interesan haciendo uso métodos como *find* en los *tags* del cuerpo de la página accesible desde los objetos definidos con *BeautifulSoup* (estos objetos les hemos dado el nombre de *soup* en nuestro código).
- La siguiente función es *get_movies_from_current_page()* que utilizamos para navegar por las distintas películas de una página, básicamente se encarga de buscar los títulos de las películas y sus *url* asociadas a las páginas de estas para posteriormente hacer uso de la función *get_movie_info()* sobre ellas. Esta función guarda las “series” de *Pandas* resultantes de *get_movie_info()* en una lista y las devuelve.
- La función principal es *scrape_movies()* a la que le damos como *input* la *url de Filmaffinity* y esta función accede a las categorías de interés (*En cartelera* y *Ya para alquilar*) para luego utilizar las otras dos funciones secuencialmente y de este modo navegar a través de cada una de las películas pertenecientes a dichas categorías. Esta es la función que “calleamos” en *main.py* para obtener un *dataframe de Pandas*.
- Por último, tenemos la función *dataset_writer()* que también “calleamos” en *main.py* para exportar el *dataframe* de *Pandas* obtenido con *scrape_movies()* anteriormente en un archivo *csv* en la ruta deseada (*/dataset*) dándole como *input* el nombre deseado del archivo (*recent_movies_info.csv*). Esta función es muy sencilla y de momento únicamente convierte el *dataframe* en *dataset* con formato *csv* teniendo en cuenta el *encoding utf-8* y la ruta donde queremos alojar el *dataset* con un nombre dado como *input*. Esta función también comprueba si el directorio */dataset* existe en el proyecto y lo crea si fuera necesario para poder guardar el *dataset* en esa ruta.

Por último comentaremos algunas dificultades que hemos tenido con *Filmaffinity*:

- Hemos introducido una pausa de 2 segundos entre el *scrapping* de cada película para evitar la saturación del servidor o la detección de nuestro programa.
- Hemos tenido que añadir algunas excepciones a la función *get_movie_info()* debido a que no todas las películas tienen toda la información disponible en sus respectivas páginas de *Filmaffinity*.
- Hemos tenido que ingeniar una forma de navegar dinámicamente a través de las distintas páginas dentro de la categoría *Ya para alquilar* que solo mostraba 30 resultados por página en lugar de todas en una única página. Esto lo hemos hecho con una lista de *links* de *next page* sobre los que iteramos en un bucle *for* en la función *scrape_movies()* para la categoría ya mencionada.
- Para evitar la detección de nuestro *scrapping* hemos tenido que cambiar las cabeceras de las peticiones al servidor realizadas por la librería *requests* de *Python*. Esto lo hacemos como ya comentamos con el diccionario *new_headers* que hemos definido en el archivo *settings.py*. Para mostrar un ejemplo de como es el resultado tras modificar las cabeceras de este modo, hemos implementado al comienzo de *main.py* justo antes de realizar el *web scrapping*, un ejemplo que muestra por pantalla las cabeceras antes y después de la modificación al acceder a una página web ajena al proyecto que hemos definido como *url_bin* en *settings.py*.