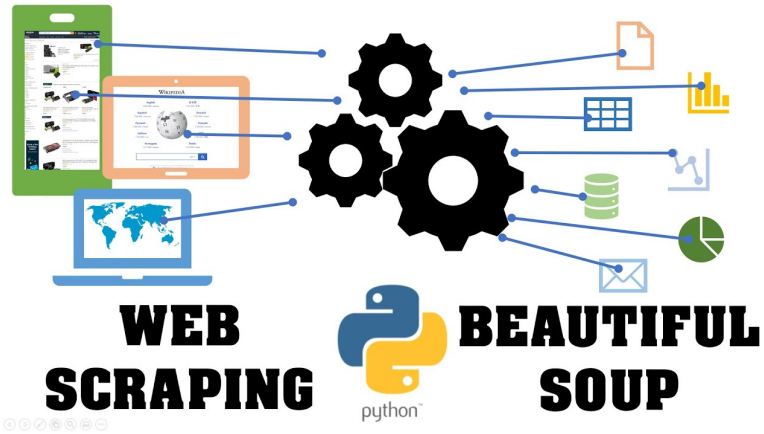
Web Scrapping en *Python* con *beautiful soup*



Realizado por:

Francisco José Rodríguez Cerezo

Índice

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 obtendrémos 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*).

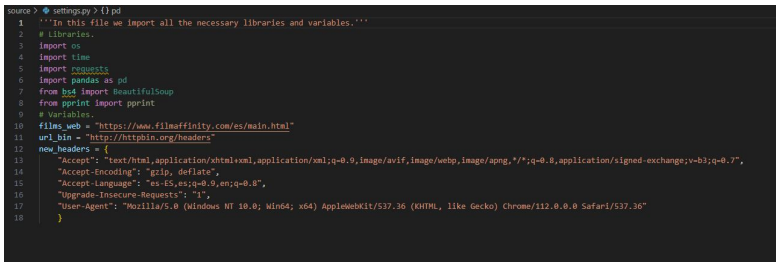


Figura 1: *Script* settings.py.

- **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.

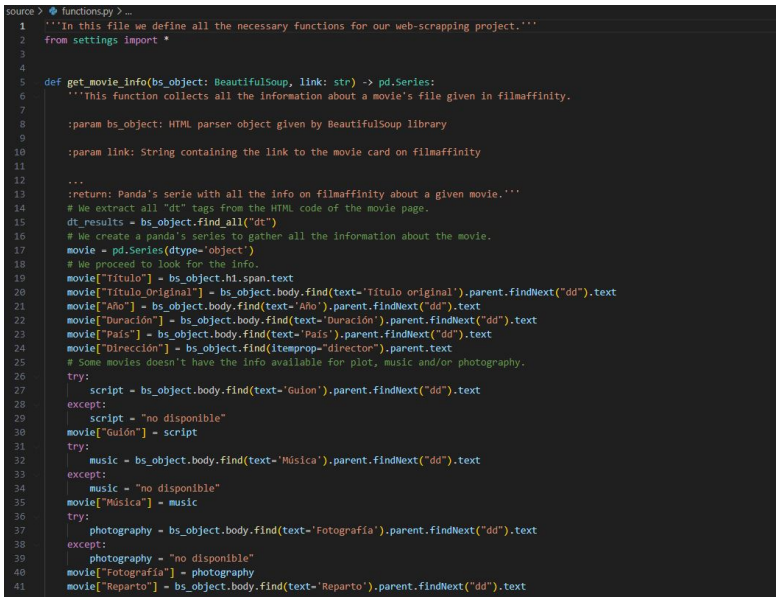


Figura 2: *Script* functions.py (1).

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

```python
def get_movies_from_current_page(bs_object: BeautifulSoup) -> list:
    '''This function collects all the info from movie cards in a given page.

    :param bs_object: HTML parser object given by BeautifulSoup library

    ...
    :return: List that contains panda's series with the info of the movies on the given page.'''
    # Empty list to append the links from the page.
    movies_series = []
    # We look for movie titles to get the link for their cards with the info.
    movie_links = bs_object.find_all(class_='movie-title')
    link_list = [l.a['href'] for l in movie_links]
    # Iterate along all movies in the page once we got all the movie cards links.
    for link in link_list:
        # We iterate along all the movie cards on the current page.
        page = requests.get(link, headers=new_headers)
        soup = BeautifulSoup(page.content, "html.parser")
        # In each iteration we apply "get_movie_info" for the current movie and append
        # the info in our "movie_series" list.
        movies_series.append(get_movie_info(soup, link))
        # We introduce a 2 seconds pause to avoid server saturation and/or detection of our script.
        time.sleep(2)
    return movies_series
```

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

```python
def scrape_movies(url: str) -> pd.DataFrame:
    '''This function navigate through the website applying "get_movies_from_current_website" function.

    :param url: String with the url to the website.

    ...
    :return: Panda's dataframe that contains all the info about all the movies from the website specific sections.'''
    print("\nScrapping in progress. Please wait. This might take a few minutes.")
    # We get sent the petition to the website using modified header.
    page = requests.get(url, headers=new_headers)
    # We parse the HTML body of the main website page.
    soup = BeautifulSoup(page.content, "html.parser")
    # We look for the desired sections, in this case, the sections are "Películas en cartelera" and "Ya para alquilar"
    # which were studied before.
    url_alquilar = soup.body.find(text='Ya para alquilar').parent['href']
    url_cartelera = soup.body.find(text='Películas en cartelera').parent['href']
    # We begin with "Películas en cartelera" which is a simple page.
    page = requests.get(url_cartelera, headers=new_headers)
    soup = BeautifulSoup(page.content, "html.parser")
    # We apply the "get_movies_from_current_page" on this website section.
    movies_series = get_movies_from_current_page(soup)
    # The following website section ("Ya para alquilar") requires interactive navigation.
    page = requests.get(url_alquilar, headers=new_headers)
    soup = BeautifulSoup(page.content, "html.parser")
    # We look for the tag "a" which are the "next page" button on the page.
    next_links = soup.find(class_="pager").find_all('a')
    # Create a list with all "next pages" from from the section.
    next_links_list = list(set([l['href'] for l in next_links]))
    # Reapply the "get_movies_from_current_page" on "movies_series" to expand it
    # with the info from this new page.
    movies_series += get_movies_from_current_page(soup)
    # Iterate over the "next pages" links to navigate through them and gather the info from
    # the movies on all of them.
    for next_page in next_links_list:
        page = requests.get(next_page, headers=new_headers)
        soup = BeautifulSoup(page.content, "html.parser")
        movies_series += get_movies_from_current_page(soup)

    # Concatenate the panda's series with the movies info in a panda's dataframe
    # with the convenient shape.
    results_df = pd.concat(movies_series, axis=1).transpose()
    print("\nDONE!")
    return results_df
```

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

```python
def dataset_writer(df: pd.DataFrame, name: str) -> None:
    '''This function creates a dataset in .csv fromat with the info from a given panda's dataframe
    using a desired name on the correct path of this project which is "datset/name.csv".

    :param df: Panda's dataframe with the neccesary info to create the dataset.

    :param name: String with the desired name for the .csv file output.'''
    # Define the directory for the dataset and create it if necessary.
    dataset_directory = "./dataset"
    if not os.path.exists(dataset_directory):
        os.mkdir(dataset_directory)
    # Define the full name including the path for our csv dataset file.
    file_path = os.path.join(dataset_directory, name + ".csv")
    # Convert the panda's dataframe into .csv file with the path "dataset" of this project
    # making sure to avoid "UnicodeEncode errors" using "utf-8" since the website is in spanish.
    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*.

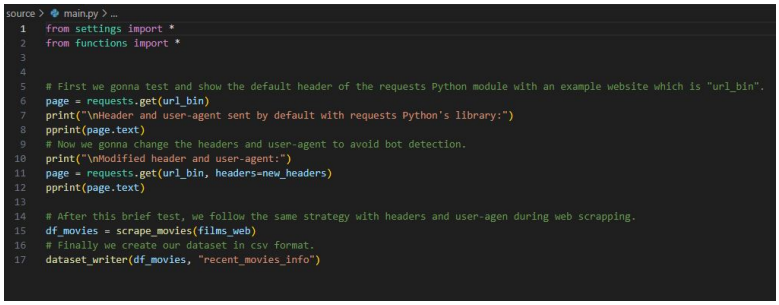


Figura 7: *Script* main.py.

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 utg-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*.