Tarea 3: Web Scraping

Sergio Lagos Vergara

Contents

1 Introducción **2**

2 Desarrollo **4**

2.1 CheckName 4
2.2 ITB 5
2.3 ITB ajustado a la inflación 5
2.4 Peliculas 6
2.5 PeliculasMayorIngreso 9
2.6 PeliculasMenorIngreso 10

3 Conclusión **11**

4 Bibliografía **11**

1 Introducción

En esta actividad se nos propuso extraer información de la página web **box office mojo**, buscando patrones con expresiones regulares en el html de la página, en donde dado un actor, debemos retornar el Ingreso Total Bruto de su carrera (ITB), el ITB ajustado a la inflación, las 5 últimas películas, las 10 películas con mayor ingreso bruto y las 10 películas con menor ingreso bruto.

A modo de ejemplo, este es el modelo de página del actor Robert Downey Jr. de donde se obtendrá la **Información**.

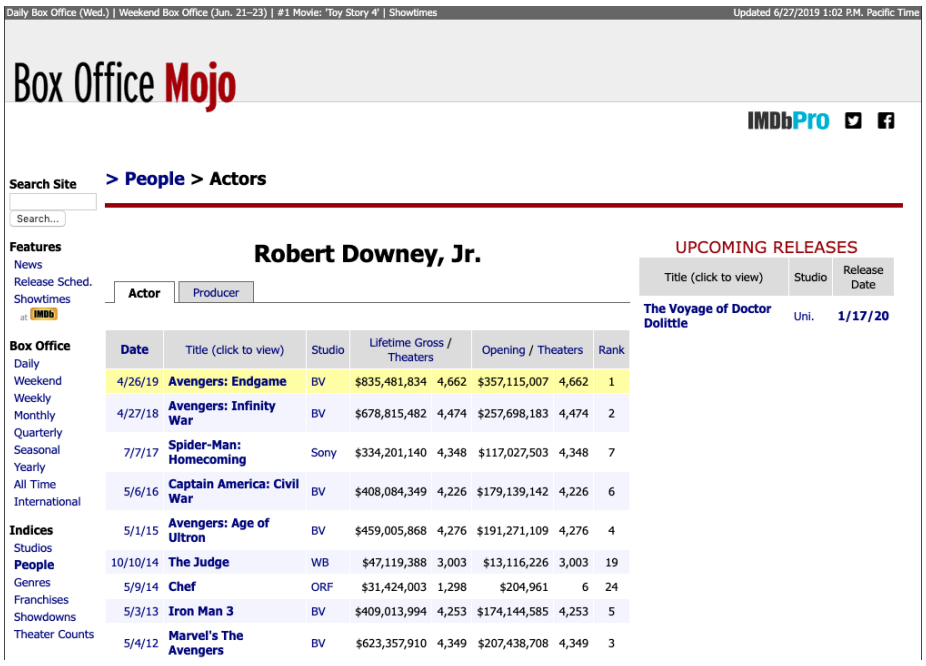


Figura 1: Robert Downey Jr. web page

A primera vista, ya podemos observar que en la tabla principal las películas están ordenadas desde la más reciente hasta la primera cinta del actor. De esta misma manera, si clickeamos en **Lifetime Gross**, nos redireccionará a una página con la misma tabla, pero las películas estan ordenadas descendentemente respecto al ingreso que tuvo y si presionamos denuevo **Lifetime Gross**, obtendremos la tabla con las películas ordenadas de manera ascendente respecto al ingreso. Esto es muy importante, ya que utilicé estos links para obtener información solicitada. Además, si seguimos bajando por la página principal del actor, nos podremos encontrar con el ITB y el ITB ajustado a la inflación.

1/11/85	**Tuff Turf**	Orion	$9,369,329	653	$1,509,380	440	37
10/26/84	**Firstborn**	Par.	$6,250,994	929	$1,606,214	800	39
3/4/83	**Baby, It's You**	Par.	$1,867,792	30	$12,420	1	-

Note: Titles in grey are cameo or bit parts and not counted in totals and averages.

Lifetime Gross Total (56): $5,795,881,106
Average: $103,497,877
Opening Gross Average (37): $55,360,534 (Wide Releases Only)

Figura 2: Lifetime Gross

54	**Game 6**	KMG	**$178,400**	$129,664	3/10/06
55	**Friends and Lovers**	Lions	**$59,500**	$33,543	4/16/99
56	**Hugo Pool**	N.Arts	**$26,200**	$13,330	12/12/97

Note: Titles in grey are cameo or bit parts and not counted in totals and averages.

Adjusted Total: $6,821,775,600
Average: $121,817,400

Figura 3: Adjusted Lifetime Gross

2 Desarrollo

Tomando en cuenta los requerimientos, decidí escribir un programa en Python 3, utilizando principalmente la librería **re**, para las expresiones regulares y **requests**, para obtener el HTML de la página.

Comenzando, dividí mi programa en 6 funciones que me ayudaron a recopilar información, pero para mejorar la explicación del código, en la figura 4 se muestra lo que se pide por consola:

```
if __name__ == '__main__':
    actor = input("Nombre del actor: ")
    datos = requests.get('https://www.boxofficemojo.com/people/chart/?id='+actor.replace(" ","")+'.htm')
    CheckName(actor,datos)
```

Figura 4: Main code

En donde 'datos' es el request.get() a la página asociada al actor.

2.1 CheckName

Esta función recibe como parámetros al actor y el requests.get() a la url asociada. Principalmente lo que hago acá es asegurarme de que el actor ingresado sea válido.

```
def CheckName(actor,datos):

    if datos.url == "https://www.boxofficemojo.com/people/":
        actor = input("Oops! Al parecer ingresaste mal el nombre, intenta denuevo: ")
        print ("\n")
        datos = requests.get('https://www.boxofficemojo.com/people/chart/?id='+actor.replace(" ","")+'.htm')
        CheckName(actor,datos)
    else:
        ITB(datos)
        ITBInflacion(datos)
        Peliculas(actor)
        PeliculasMayorIngreso(actor)
        PeliculasMenorIngreso(actor)
    return
```

Figura 5: CheckName code

Para esto lograr esto, noté que al ingresar un actor que no se encuentra en la página, el requests.get() redirecciona a **https://www.boxofficemojo.com/people**. Por lo tanto, basta con hacer una comparacion entre datos.url y la página redireccionada para saber si el nombre del actor está bien o mal. Al pasar esta comparación, se llaman a las siguientes funciones.

2.2 ITB

Para obtener el ITB del actor, hice una expresión regular que haga match con la parte del HTML que se encuentra el LifetimeGross como se muestra en la Figura 2.

```
def ITB(datos):
    ITB = re.findall(r"Lifetime\sGross\sTotal\s\((.*?)\):\s(\$([0-9]{1,3},([0-9]{3},)*[0-9]{3}|[0-9]+)(\.[0-9][0-9])?)", datos.text)
    print ("Ingreso total bruto: ", ITB[0][1])
    print ("\n")

    return
```

Figura 6: ITB code

Aquí ITB, es una lista con una lista de este estilo:

```
[('9', '$2,329,782,367', '2,329,782,367', '782,', '')]
```

Figura 7: ITB

En donde el segundo elemento de la lista de la lista, corresponde al Ingreso Total Bruto.

2.3 ITB ajustado a la inflación

Esta función es prácticamente igual a la anterior, solo que se busca el match con el Adjusted Lifetime Gross como se ve en la Figura 3.

```
def ITBInflacion(datos):
    AdjustedTotal = re.findall(r"Adjusted\sTotal\:\s(\$([0-9]{1,3},([0-9]{3},)*[0-9]{3}|[0-9]+)(\.[0-9][0-9])?)", datos.text)
    print ("Ingreso total bruto ajustado a la inflacion: ", AdjustedTotal[0][1])
    print ("\n")

    return
```

Figura 8: ITB ajustado code

2.4 Peliculas

En esta sección de código, utilizo la url para poder imprimir por pantalla las 5 últimas películas de algún actor.

```
def Peliculas(datos):
    Totalpeliculas = re.findall(r"<b>(.*?)</b>", datos.text)
    PeliculasGrises = re.findall(r"<font color=\"\#666666\">(.*?)</font>", datos.text)
    cont = 0

    print ("Ultimas 5 peliculas: ")

    for peliculas in range (1,6):

        match = re.match(r"<font color=", Totalpeliculas[peliculas])
        match2 = re.match(r"Average",Totalpeliculas[peliculas])

        if match2:
            print("\n")
            return

        if match:
            if PeliculasGrises[cont-1] != '(Cameo)' and PeliculasGrises[cont-1] != '(Voice)' and PeliculasGrises[cont-1] != '(Narrator)':
                print (PeliculasGrises[cont])
                cont = cont + 2
            else:
                print(PeliculasGrises[cont+1])
        else:
            print (Totalpeliculas[peliculas])

    print ("\n")

    return
```

Figura 9: Peliculas code

En este nivel se me presentaron diversos 'problemas'. Resulta que Totalpeliculas es una lista que contiene todos los strings encerrados entre <b >y <\b >. Esto nos retorna una lista de estas características:

```
['Date', '<font color="#666666">The Man Who Killed Don Quixote</font>', 'Absolutely Anything', 'Average: $20,169',
 for Ticket Price Inflation', '<font color="#666666">The Man Who Killed Don Quixote</font>', '$389,100', 'Absolute
g', '$20,300', 'Average: $20,300', 'Worldwide', '<font color="#666666">The Man Who Killed Don Quixote</font>', '$3
```

Figura 10: Totalpeliculas

Traducido a la página web, sería:

Terry Gilliam

Actor | Director | Writer

Date	Title (click to view)	Studio	Lifetime Gross / Theaters		Opening / Theaters		Rank
4/10/19	**The Man Who Killed Don Quixote** (Voice)	Scre.	$389,080	756	$12,756	10	-
5/12/17	**Absolutely Anything** (Voice)	ADC	$20,169	46	n/a	-	1

Note: Titles in grey are cameo or bit parts and not counted in totals and averages.

Lifetime Gross Total (1): $20,169
Average: $20,169

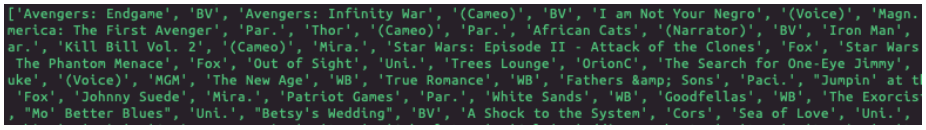
Figura 11: Terry Gilliam

Después de varios test cases, llegué a la conclusión de que el siempre el primer ítem de Totalpeliculas es 'Date', de que seguido de la última película del actor viene el string 'average' y que las películas marcadas en gris vienen con el tag <font.... Si solo imprimía por pantalla Totalpeliculas desde el segundo ítem, también se imprimían los tags de las películas en gris. Como solución, creé PelículasGrises, que es una lista con todas los string grises dentro del HTML de la página. Lamentablemente no solo guarda las películas grises, sino también, los estudios, el string 'voice', cuando el actor hace la voz en la película, 'narrator' cuando la narra y 'cameo' cuando hace un cameo como podemos ver a continuación:

Samuel L. Jackson

Date	Title (click to view)	Studio	Lifetime Gross / Theaters		Opening / Theaters		Rank
6/14/19	**Shaft (2019)**	WB (NL)	$17,153,729	2,952	$8,901,419	2,952	49
4/26/19	**Avengers: Endgame**	BV	$835,481,834	4,662	$357,115,007	4,662	-
3/8/19	**Captain Marvel**	BV	$426,776,678	4,310	$153,433,423	4,310	4
1/18/19	**Glass**	Uni.	$111,035,005	3,844	$40,328,920	3,841	17
6/15/18	**Incredibles 2** (Voice)	BV	$608,581,744	4,410	$182,687,905	4,410	2
4/27/18	**Avengers: Infinity War** (Cameo)	BV	$678,815,482	4,474	$257,698,183	4,474	-
8/18/17	**The Hitman's Bodyguard**	LG/S	$75,468,583	3,377	$21,384,504	3,377	25
3/10/17	**Kong: Skull Island**	WB	$168,052,812	3,846	$61,025,472	3,846	10
2/3/17	**I am Not Your Negro** (Voice)	Magn.	$7,123,919	320	$686,378	43	-
1/20/17	**xXx: The Return of Xander Cage**	Par.	$44,898,413	3,651	$20,130,142	3,651	34

Figura 12: Samuel L Jackson Page



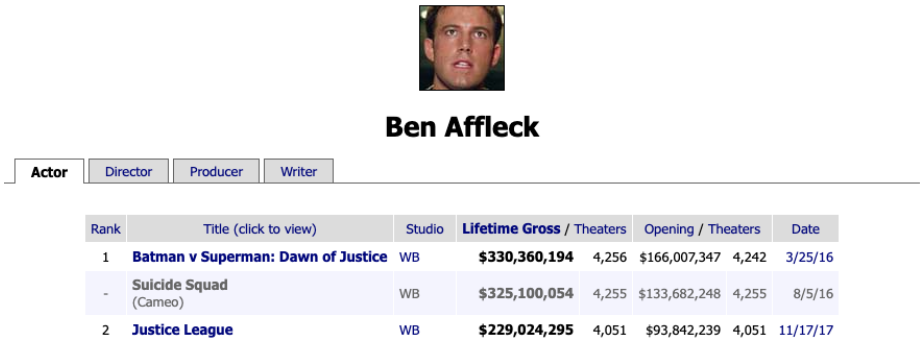
```
['Avengers: Endgame', 'BV', 'Avengers: Infinity War', '(Cameo)', 'BV', 'I am Not Your Negro', '(Voice)', 'Magn.
merica: The First Avenger', 'Par.', 'Thor', '(Cameo)', 'Par.', 'African Cats', '(Narrator)', 'BV', 'Iron Man',
ar.', 'Kill Bill Vol. 2', '(Cameo)', 'Mira.', 'Star Wars: Episode II - Attack of the Clones', 'Fox', 'Star Wars
 The Phantom Menace', 'Fox', 'Out of Sight', 'Uni.', 'Trees Lounge', 'OrionC', 'The Search for One-Eye Jimmy',
uke', '(Voice)', 'MGM', 'The New Age', 'WB', 'True Romance', 'WB', 'Fathers &amp; Sons', 'Paci.', "Jumpin' at t
 'Fox', 'Johnny Suede', 'Mira.', 'Patriot Games', 'Par.', 'White Sands', 'WB', 'Goodfellas', 'WB', 'The Exorcis
, "Mo' Better Blues", 'Uni.', "Betsy's Wedding", 'BV', 'A Shock to the System', 'Cors', 'Sea of Love', 'Uni.',
```

Figura 13: PeliculasGrises Samuel

Dadas las condiciones, tambien creé las variables match, que retorna true si la casilla de Totalpeliculas es un string con tag de color gris y match2, que retorna true si ya se acabaron las películas del actor en caso de que este tenga menos de 5. Match2 retorna true cuando el siguiente elemento en la lista Totalpeliculas hace match con la expresión regular "Average". Como se pudo ver anteriormente, la lista PeliculasGrises no solo almacena los string de las películas. Por lo tanto, para siempre imprimir una, se verifica que el elemento anterior en la lista no sea ni "(cameo)","(narrator)" o "(voice)" y si lo es, imprime el siguiente elemento de PeliculasGrises y no el actual.

2.5 PeliculasMayorIngreso

Para obtener esta información, como se menciona al principio, se usa la url que ordena de forma ascendente las películas respecto a sus ingresos. De esta forma, la manera de operar es casi la misma a la función Peliculas, solo que con 10 elementos. El problema aquí es que Totalpeliculas cambia un poco su estructura y hay que iterarla de diferente manera como se ve en la Figura 15.

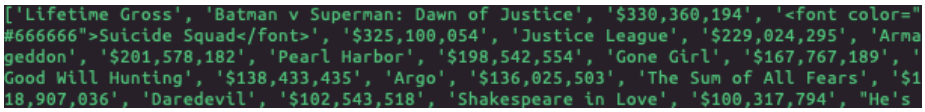


Ben Affleck

Actor | Director | Producer | Writer

Rank	Title (click to view)	Studio	Lifetime Gross / Theaters		Opening / Theaters		Date
1	Batman v Superman: Dawn of Justice	WB	$330,360,194	4,256	$166,007,347	4,242	3/25/16
-	Suicide Squad (Cameo)	WB	$325,100,054	4,255	$133,682,248	4,255	8/5/16
2	Justice League	WB	$229,024,295	4,051	$93,842,239	4,051	11/17/17

Figura 14: Ben Affleck Page



```
['Lifetime Gross', 'Batman v Superman: Dawn of Justice', '$330,360,194', '<font color="
#666666">Suicide Squad</font>', '$325,100,054', 'Justice League', '$229,024,295', 'Arma
geddon', '$201,578,182', 'Pearl Harbor', '$198,542,554', 'Gone Girl', '$167,767,189', '
Good Will Hunting', '$138,433,435', 'Argo', '$136,025,503', 'The Sum of All Fears', '$1
18,907,036', 'Daredevil', '$102,543,518', 'Shakespeare in Love', '$100,317,794', "He's
```

Figura 15: PeliculasGrises Ben

En síntesis, el código es casi el mismo a Peliculas, solo que hay que iterar TotalPeliculas intercaladamente desde el segundo ítem, ya que el primero siempre es 'Lifetime Gross'.

```
def PeliculasMayorIngreso(actor):
    datos = requests.get('https://www.boxofficemojo.com/people/chart/?view=Actor&id='+actor.replace(" ","")+'.htm&sort=gross&order=DESC&p=.htm')
    Totalpeliculas = re.findall(r"<b>(.*?)</b>", datos.text)
    PeliculasGrises = re.findall(r"<font color=\"\#666666\">(.*?)</font>", datos.text)
    cont = 0

    print ("10 peliculas con mayor ingreso: ")


    for peliculas in range(0,20):
        if peliculas % 2 == 1:
            match = re.match(r"<font color=", Totalpeliculas[peliculas])
            match2 = re.match(r"Average",Totalpeliculas[peliculas])

            if match2:
                print("\n")
                return

            if match:
                if PeliculasGrises[cont-1] != '(Cameo)' and PeliculasGrises[cont-1] != '(Voice)' and PeliculasGrises[cont-1] != '(Narrator)':
                    print (PeliculasGrises[cont])
                    cont = cont + 2
                else:
                    print(PeliculasGrises[cont+1])
            else:
                print (Totalpeliculas[peliculas])


    print ("\n")

    return
```

Figura 16: PeliculasMayorIngreso code

2.6 PeliculasMenorIngreso

En esta última función, es lo mismo que PeliculasMayorIngreso pero con la url de las películas ordenadas de forma ascendente respecto a su ingreso total.

```
def PeliculasMenorIngreso(actor):
    datos = requests.get('https://www.boxofficemojo.com/people/chart/?view=Actor&id='+actor.replace(" ","")+'.htm&sort=gross&order=ASC&p=.htm')
    Totalpeliculas = re.findall(r"<b>(.*?)</b>", datos.text)
    PeliculasGrises = re.findall(r"<font color=\"\#666666\">(.*?)</font>", datos.text)
    cont = 0

    print ("10 peliculas con menor ingreso: ")

    for peliculas in range(0,20):
        if peliculas % 2 == 1:
            match = re.match(r"<font color=", Totalpeliculas[peliculas])
            match2 = re.match(r"Average",Totalpeliculas[peliculas])

            if match2:
                print("\n")
                return

            if match:
                if PeliculasGrises[cont-1] != '(Cameo)' and PeliculasGrises[cont-1] != '(Voice)' and PeliculasGrises[cont-1] != '(Narrator)':
                    print (PeliculasGrises[cont])
                    cont = cont + 2
                else:
                    print(PeliculasGrises[cont+1])
            else:
                print (Totalpeliculas[peliculas])


    print ("\n")

    return
```

Figura 17: PeliculasMenorIngreso code

3 Conclusión

En conclusión, este es un programa que dado un actor, usa expresiones regulares para identificar patrones dentro de la página web **box office mojo** y así obtener diversa información respecto al actor requerido. Quizás se puede mejorar la parte de como se imprimen las películas cuando el programa se enfrenta al problema de tener texto de otro color, pero al fin y al cabo el problema se resuelve de manera satisfactoria.

4 Bibliografía

References

[1] BOX OFFICE MOJO

[2] RE LIBRARY

[3] REQUESTS LIBRARY