UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Manejo de Datos

Web scraper

García Ramírez Alison

Navarro Castillo Rafael Alberto Carlos

Olivares Herrera Luis Félix

Rodríguez Hernández Fernanda

Web Scraper

Como su nombre sugiere, se le conoce como *web scraping (raspado web)* a una técnica de extracción de datos de páginas web mediante un software. Resulta muy útil para obtener grandes cantidades de información de manera automática.

Veremos cómo elaboramos el nuestro, bajo la temática de tiendas de ropa.

Para empezar, debemos instalar las paqueterías necesarias. En nuestra implementación, utilizamos:

- *Pandas*
- *Re*
- *Time*
- *Datetime*
- *Selenium*

Además, encontramos útil trabajar en *Jupyter Notebook*, pues en otro entorno no reconocía la última paquetería.

También, descargamos *chromedriver.* Dependiendo de la versión de *Google Chrome* que se tenga, es la versión que se debe seleccionar en el siguiente link: [ChromeDriver - WebDriver for Chrome - Downloads (chromium.org)]

Decidimos que trabajaríamos con las marcas de *HyM, Bershka* y *Shein,* por lo que realizamos una función para cada sitio. Aún así, si se desea revisar diferentes páginas, el procedimiento sería muy similar.

Ya en el código, lo primero que hacemos es importar las librerías y asignar un *path* con la ruta donde se encuentra el *chromedriver* en nuestro dispositivo:

In [200]:

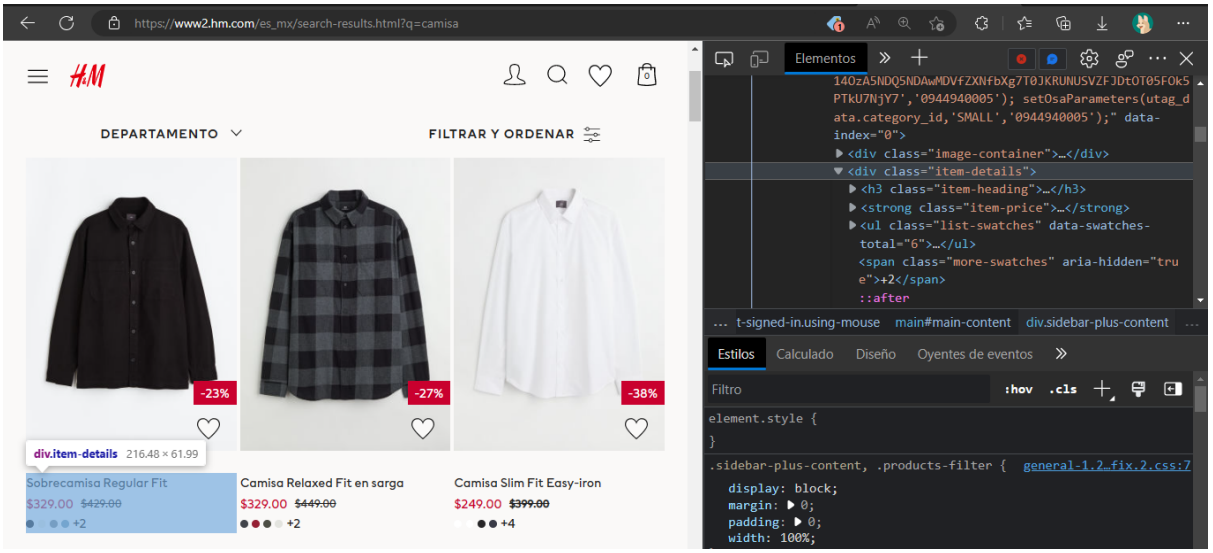
```
import pandas as pd
import time
from datetime import datetime
import re
from datetime import date
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options


#Ruta correspondiente
path = 'C:/Users/argea/Documents/Rafa/MDD/PF/chromedriver.exe'

```

Empecemos describiendo cómo se realizó la función para *HyM.*

Lo primero que hicimos fue realizar una búsqueda en el sitio oficial de la marca, para reconocer cómo luce el link en la pantalla de la lista de productos. Posteriormente, inspeccionamos la página para reconocer el nombre de la clase donde se almacena la información más importante, de la siguiente manera:



Como se aprecia en el link, el ejemplo fue tras buscar '*camisa'.* Vemos además que la clase que nos interesa se llama *item-details*.

Revisamos a detalle cómo luce el formato de la información de cada producto:

```
In [203]:
#Abrimos la página
driver=webdriver.Chrome(path)
url1 = 'https://www2.hm.com/es_mx/search-results.html?q='+'camisa'
driver.get(url1)
time.sleep(12)

opciones = driver.find_elements_by_class_name("item-details")
```

```
In [205]:
opciones[0].text.split('\n')
```

```
Out[205]: ['Sobrecamisa Regular Fit',
 '$329.00 $429.00',
 'Negro',
 'Crema',
 'Azul grisáceo',
 'Azul claro',
 '+2']
```

Tras separar la información por saltos de línea al hacer *split('\n'),* vemos que los datos que nos interesan se encuentran en los primeros lugares, pues primero está el nombre, y

después los precios con y sin descuentos. Sin embargo, no podemos generalizar por completo este formato, pues en productos como el que mostraremos, hay un título adicional:


```
1  opciones[21].text.split('\n')
```

```
['Premium Selection',
 'Camisa Regular Fit de manga corta',
 '$499.00',
 'Beige claro',
 'Blanco']
```

Vemos que debido al título *'Premium Selection',* la información útil se desplaza un lugar, por lo que hay que tenerlo en cuenta para no causar errores. Otra cosa importante es que si no hay descuentos, en el lugar correspondiente solo hay un dato.

Esta situación de tener etiquetas adicionales antes del nombre se presentó en los tres sitios que trabajamos, por lo que motivó a realizar un método auxiliar para reconocer si estábamos en esa situación o no. Pero primero, veamos la función principal:

```
def hym(producto):
    '''
    Función para obtener datos en sitio HyM
    ------------------------------------------------
    Parameters:
        producto: str
            Tipo de producto que buscará en el sitio
    ------------------------------------------------
    Returns:
        df: pandas.DataFrame()
            DataFrame con la información más importante de los productos
    '''
    #Abrimos la página
    driver=webdriver.Chrome(path)
    url1 = 'https://www2.hm.com/es_mx/search-results.html?q='+producto
    driver.get(url1)
    time.sleep(12)

    #Identificamos la clase de la información de los productos
    opciones = driver.find_elements_by_class_name("item-details")
```

La primera sección es clara. Abrimos la página con el formato del link que identificamos, y guardamos las opciones de productos con la clase correspondiente.

Lo que se buscará es guardar en listas diferentes la información que necesitamos, esto es, una para los nombres de los productos, otra para el precio actual, y una última para el precio sin descuento (si es que lo hay).

Haremos un ciclo por cada producto. Llamaremos a nuestro método auxiliar *verificar(),* para saber si hay una etiqueta extra o no. Este método retorna *True* si es el orden conveniente, y un *False* si no, es decir, si hay un dato extra.

Guardamos el nombre en la lista correspondiente.

Después limpiamos el lugar de los costos, quitando el signo de peso y comas, con la intención de poder convertir los datos a flotantes. Al usar el método *split()* estaremos separando los dos precios, si es que existen. Por ello, utilizamos una excepción, para agregar un *NA* a la lista de precios sin descuento en caso de que solo existiera un dato.

```
    #Generamos 3 listas vacías donde se añadirá la información correspondiente
    nombres = []
    precio_actual = []
    precio_sin_desc = []

    for i in opciones:
        pro_info = i.text.split('\n')

        #Si el orden es correcto, empezamos desde el primer índice a llenar los datos. Si no, desde el segundo
        if verificar(pro_info):
            nombres.append(pro_info[0])

            precios = pro_info[1]
            precios = re.sub("[$,]","",precios).split()
            try:
                precio_actual.append(float(precios[0]))
                precio_sin_desc.append(float(precios[1]))
            except IndexError:
                precio_sin_desc.append('NA')
```

Para la siguiente parte, es repetir lo mismo pero modificando los índices, pues habrá que hacerlo una posición más arriba:

```
        else:
            nombres.append(pro_info[1])   #Se agrega el nombre

            precios = pro_info[2]
            precios = re.sub("[$,]","",precios).split()  #Se limpia los datos de los precios

            #Usamos excepciones para guardar los precios
            try:
                precio_actual.append(float(precios[0]))
                precio_sin_desc.append(float(precios[1]))
            except IndexError:
                #De no tener descuento, se llena el valor con un NA
                precio_sin_desc.append('NA')

    #Llamamos a la función auxiliar para generar dataframe
    df = formato(nombres, precio_actual, precio_sin_desc, 'HyM', producto)



    return df
```

La función *formato()* también es auxiliar. Debido a que en cada método de los tres sitios debemos generar un dataframe con el mismo orden de datos, es conveniente hacer una función por separado que dé el formato correcto, para no repetir lo mismo.

La implementación de las funciones auxiliares es la siguiente:

```python
def verificar(pro_info):
    '''
    Función auxiliar para verificar que los datos se encuentren en el orden deseado
    ---------------------------------------------
    Parameters:
        pro_info: list
            Información del producto
    '''
    #Tratamos de convertir el dato del precio a flotante. Si hay error, es porque el producto tiene un título extra,
    #por lo que hay que cambiar los indices en la función principal
    precio = pro_info[1]
    precio = re.sub("[MXN$,]","",precio).split()
    try:
        temp = float(precio[0])
    except ValueError:
        return False

    return True
```

```python
def formato(l1,l2,l3,sitio,producto):
    '''
    Función auxiliar para ordenar los datos obtenidos y generar un DataFrame
    ---------------------------------------------
    Parameters:
        l1: list
            Lista con nombres de los productos
        l2: list
            Lista con precios sin descuento
        l3: list
            Lista con precios sin descuento
        sitio: str
            Nombre de la página web donde se buscó la información
        producto: str
            Tipo de producto que se consultó
    ---------------------------------------------
    Returns:
        df: pandas.DataFrame()
            Dataframe con el formato deseado
    '''

    info = list(zip(l1,l2,l3))

    #Creamos dataframe y añadimos columnas importantes
    df = pd.DataFrame(info, columns=['Descripción', 'Precio Actual', 'Precio Sin Descuento'])
    df.insert(0, "Producto", producto, allow_duplicates=False)
    df.insert(0, "Sitio", sitio, allow_duplicates=False)
    df.insert(5, "Fecha De Consulta", datetime.today().strftime('%Y-%m-%d'), allow_duplicates=False)



    return df
```

En el método *formato()* lo que hacemos es ordenar las listas para crear un DataFrame que guarde los datos correctamente. Además, agregamos una columna para guardar el sitio donde se buscaron, el tipo de producto y la fecha de consulta.

Así sería la salida tras ingresar en nuestro método el producto *'Gorro'*:

```
64 hym('Gorro')
```

	Sitio	Producto	Descripción	Precio Actual	Precio Sin Descuento	Fecha De Consulta
0	HyM	Gorro	Chamarra acolchada con gorro	299.0	549.0	2022-12-14
1	HyM	Gorro	Sudadera Relaxed Fit	449.0	NA	2022-12-14
2	HyM	Gorro	Gorro tejido	179.0	NA	2022-12-14
3	HyM	Gorro	Gorro con pompón	199.0	249.0	2022-12-14
4	HyM	Gorro	Gorro en tejido acanalado	249.0	NA	2022-12-14
5	HyM	Gorro	Sudadera Relaxed Fit	549.0	NA	2022-12-14
6	HyM	Gorro	Sudadera Regular Fit	549.0	NA	2022-12-14
7	HyM	Gorro	Sudadera con diseño	169.0	249.0	2022-12-14
8	HyM	Gorro	Chamarra en denim con gorro	699.0	NA	2022-12-14
9	HyM	Gorro	Sudadera con diseño	399.0	NA	2022-12-14
10	HyM	Gorro	Sudadera Relaxed Fit	449.0	NA	2022-12-14
11	HyM	Gorro	Conjunto de 2 piezas	749.0	NA	2022-12-14
12	HyM	Gorro	Sudadera Relaxed Fit	449.0	NA	2022-12-14
13	HyM	Gorro	Chaleco puffer con gorro	749.0	NA	2022-12-14
14	HyM	Gorro	Sudadera con diseño	449.0	NA	2022-12-14
15	HyM	Gorro	Sudadera Regular Fit	499.0	NA	2022-12-14

Para los siguientes sitios, el procedimiento es análogo. Solo hay que verificar el nombre de la clase correspondiente y el formato de los datos, pero la lógica para implementar la función es la misma.

Finalmente, revisemos la función principal. Se encargará de recopilar la información de una lista de productos en los tres sitios, para después unirla en un solo DataFrame y guardarlo en un archivo de excel.

```python
def main_excel(lis_productos):
    '''
    Función que genera un archivo de excel con la información de una lista de productos
    en los sitios: HyM, Bershka y Shein
    ----------------------------------------------------
    Parameters:
        lis_productos: list
            Lista con productos que se desea analizar
    '''
    #Creamos 3 dataframes donde se guardarán la información de cada búsqueda
    df1 = pd.DataFrame()
    for i in lis_productos:
        t = hym(i)
        df1 = pd.concat([df1, t])

    df2 = pd.DataFrame()
    for i in lis_productos:
        t = bka(i)
        df2 = pd.concat([df2, t])

    df3 = pd.DataFrame()
    for i in lis_productos:
        t = shn(i)
        df3 = pd.concat([df3, t])

    #Al finalizar los ciclos, unimos todos los dataframes y ajustamos índices
    df_final = pd.DataFrame()
    df_final = pd.concat([df1,df2,df3])
    df_final.index = range(df_final.shape[0])

    #Guardamos un archivo de excel con la información obtenida
    df_final.to_excel('Proyecto.xlsx')
```

Para cada sitio, se empieza con un DataFrame vacío, de tal forma que se concatenen los que se van generando. Al terminar, guarda un archivo de excel con nombre *'Proyecto.xlsx'* .

Veamos un resultado tras ingresar en esta función la lista ['Camisa', 'Falda', 'Suéter'] :

```python
archivo = 'C:/Users/argea/Documents/Rafa/MDD/PF/Proyecto.xlsx'
DF = pd.read_excel(archivo)
DF
```

	Unnamed: 0	Sitio	Producto	Descripción	Precio Actual	Precio Sin Descuento	Fecha De Consulta
0	0	HyM	Camisa	Sobrecamisa recubierta	599	NaN	2022-12-14
1	1	HyM	Camisa	Sobrecamisa Relaxed Fit	599	NaN	2022-12-14
2	2	HyM	Camisa	Sobrecamisa Regular Fit	329	429.0	2022-12-14
3	3	HyM	Camisa	Camisa en sarga de algodón	249	429.0	2022-12-14
4	4	HyM	Camisa	Sobrecamisa acolchada	599	NaN	2022-12-14
...	...	...	...	...	...	...	...
523	523	Shein	Sueter	Jersey con patrón de dibujo de hombros caídos	470	NaN	2022-12-14
524	524	Shein	Sueter	SHEIN Jersey de manga farol de cuello redondo	430	NaN	2022-12-14
525	525	Shein	Sueter	DAZY Jersey unicolor tejido de canalé	298	8.0	2022-12-14
526	526	Shein	Sueter	Jersey con patrón de Isla Fair de manga raglán...	525	NaN	2022-12-14
527	527	Shein	Sueter	SHEIN ICON Jersey con patrón de rayas de hombr...	431	NaN	2022-12-14

528 rows × 7 columns

Conclusiones

Consideramos que esta herramienta es de gran utilidad, pues extraer grandes cantidades de información de manera automática resulta primordial para realizar diferentes tareas, desde consultas personales para saber dónde es más barato comprar cierto producto, hasta estudios de empresas para analizar a su competencia.

A pesar de que cada página tiene sus particularidades, en general se puede replicar el código cambiando solo unas pequeñas cosas, por lo que extender el alcance de nuestra implementación es relativamente sencillo. Esto es muy importante, pues entre más sitios sea capaz de analizar, más aplicaciones tendrá.