WEB SCRAPER

¿Qué es un web scraper?

Es una técnica utilizada mediante programas de software para extraer información de sitios web. Simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.

El web scraping se utiliza en muchas empresas digitales que se dedican a la recopilación de bases de datos.

Para la elaboración del proyecto es importante descargar las paqueterías necesarias, de lo contrario tendremos problemas una vez que queramos ejecutar el programa. Ya instaladas, importamos las paqueterías a usar.

```
#Importamos las paqueterias a utilizar
import time
import numpy as np
import pandas as pd
import pandasql as ps
from selenium import webdriver
import matplotlib.pyplot as plt
```

Creamos un DataFrame y lo convertimos a “xlsx”. (Un dataframe es una herramienta de organización de datos que se utiliza para almacenar cualquier tipo de información)

```
"""
Creamos un DataFrame donde se guardaran los datos obtenidos de las
distintas paginas web
"""
aux1=pd.DataFrame() # Se guarda la variable aux1
aux1.to_excel("df_productos.xlsx",index=False) # Se convierte el DataFrame a un archivo
```

Nos basaremos en tiendas departamentales como lo son palacio de hierro, SEARS y coppel, para esto crearemos funciones respectivas para dichas tiendas que se encargaran de extraer la información.

Dentro de nuestra función se debe definir la variable que llevará la ubicación del webdriver, así como también se deberá conectar el webdriver a Chrome.

Se ingresará el URL de la pagina web de Palacio de hierro más el producto a buscar.

```
Se crea una funcion para extraer la informacion de las pagina web de Palaciode Hierro
"""
def Buscador_Precios_Selenium_Palacio(producto):
    path="C:\webdriver\chromedriver.exe" # Se define la variable que lleva la ubicaci
    driver=webdriver.Chrome(path) # Se conecta el webdriver a Chrome
    url="https://www.elpalaciodehierro.com/buscar?q="+producto # url de la pagina web
    driver.get(url) # Se consulta el URL a través del driver y su metodo get
```

Dentro de nuestra misma función se deberá buscar el nombre de la clase en donde se encuentra la información de los productos. En seguida de esto se crea una lista vacía en donde se almacenarán los url de cada producto *lista_urls=list()*

Con ayuda de for se ingresa a cada elemento de la lista en donde se buscan los url porel nombre de la etiqueta y se agrega a lista_urls en caso de no contar con url se le agregará nan. *(np.nan)*

Se crea otra lista vacia para los nombres de los productos y con ayuda de un *for* se ingresarán, se buscará los nombres los productos por el nombre de la etiqueta y se añade el elemento que se encuentra en la posición 1 a *lista_nombres.*

```
productos=driver.find_elements_by_class_name("b-product") # Se busca el nombre de la clase donde se encuentra l

lista_urls=list() # Se crea una lista vacia donde se añadiran los url de cada producto
for i in range(len(productos)): # Se ingresa a cada elemento de la lista productos
    try:
        lista_urls.append(productos[i].find_element_by_tag_name("a").get_attribute("href")) # Se buscan los url
    except:
        lista_urls.append(np.nan) # En caso de no tener url se agrega nan en su lugar

lista_nombres=list() # Lista vacia para los nombres de cada producto
for i in range(len(productos)):  # Se ingresa a cada elemento de la lista productos
    try:
        lista_nombres.append(productos[i].find_elements_by_tag_name("a")[1].text) # Se buscan los nombres de lo
    except:
        lista_nombres.append(np.nan) # En caso de no tener nombre se agrega nan en su lugar
```

Haremos dos listas, una en donde se añadirán los precios originales de los productos, la otra tendrá los precios, pero ya con descuento.

```
lista_precios=list() #
lista_promos=list() # 
```

Con ayuda de un for se ingresa a cada elemento de la lista, utilizamos *try* pues Se buscan los precios por el nombre de la clase, se separa y se añade el elemento que se encuentra en la posición 0 a *lista_precios* y en caso de no tener precio entonces se agrega un nan.

Para la lista de promociones con ayuda de un try se buscan los precios con descuento por el nombre de la clase, se separa y se añade el elemento que se encuentra en la posición 1 a *lista_promos* si no cuenta con descuento el producto se agregará un nan.

```
for i in range(len(productos)):  # Con el ciclo se ingresa a cada elemento de la lista productos
    try:
        lista_precios.append(productos[i].find_elements_by_class_name("b-product_price")[0].text.split("\n")[0])
    except:
        lista_precios.append(np.nan) # En caso de no tener precio se agrega nan en su lugar
    try:
        lista_promos.append(productos[i].find_elements_by_class_name("b-product_price")[0].text.split("\n")[1]) #
    except:
        lista_promos.append(np.nan) # En caso de no tener precio con descuento se agrega nan en su lugar
```

Ahora agregaremos los nombres de cada columna al DataFrame
df_productos=pd.DataFrame({…})

```
df_productos=pd.DataFrame({"Nombre":lista_nombres,"URL":lista_urls,"Precio1":lista_precios,"Precio2":lista_promos}) #
df_productos["Autoservicio"]="PH" # Se agrega la columna Autoservicio al DataFrame estableciendo que en cada renglon
df_productos["Producto"]=producto # Se agrega la columna Producto al DataFrame estableciendo que en cada renglon de
df_productos["Fecha"]=time.strftime("%d/%m/%y")  # Se agrega la columna Fecha (fecha de extracción)al DataFrame e
```

Reordenamos las columnas del DataFrame

```
df_productos=df_productos[["Fecha","Autoservicio","Producto","Nombre","URL","Precio1","Precio2"]

df_productos=df_productos.reset_index(drop=True)
```

Creamos un DataFrame vacío en formato xlsx lo concatenamos y lo convertimos a formato Excel para después imprimirlo

```
datos_webscraper=pd.read_excel("df_productos.xlsx") # Se crea un Dat

datos_webscraper=pd.concat([datos_webscraper,df_productos],axis=0) #

datos_webscraper.to_excel("df_productos.xlsx",index=False) #se convi

driver.quit()
print(df_productos) # Se imprime el DataFrame
```

En este proyecto realizamos el webscraper para 3 tiendas departamentales diferentes, por lo tanto, los pasos a seguir para la tienda de Coppel y SEARS es análoga a la de Palacio de Hierro. Una vez realizado los pasos anteriores para las respectivas tiendas, crearemos la función precios floats la cual llevara el código que cambiara los precios de string a floats así como quitar algunos símbolos o letras sobrantes. Declaramos 2 *for i in range* para lograrlo.

```
def precios_floats(datos):
    for i in range(len(datos["Precio1"])): # Con el ciclo se ingresa a cada elemento en la columna de Precios1
        try:
            datos["Precio1"].iloc[i]=datos["Precio1"].iloc[i].strip("$") # Se selecciona la columna de Precios1 y se quita el simbolo de pesos ($) en cada
                                                                         #renglon
        except:
            pass

    for i in range(len(datos["Precio2"])): # Con el ciclo se ingresa a cada elemento en la columna de Precios2
        try:
            datos["Precio2"].iloc[i]=datos["Precio2"].iloc[i].strip("$") # Se selecciona la columna de Precios2 y se quita el simbolo de pesos ($) en cada ren
        except:
            pass

    datos["Precio1"]=datos["Precio1"].replace(",","",regex=True) # Se reemplaza la coma que separa los miles en cada renglon para que quede todo junto en la
                                                                 #columna de Precio1
    datos["Precio2"]=datos["Precio2"].replace(",","",regex=True) # Se reemplaza la coma que separa los miles en cada renglon para que quede todo junto en la
                                                                 #columna de Precio2

    datos["Precio1"]=datos["Precio1"].replace("M","",regex=True) # En caso de tener alguna letra M se reemplaza en cada renglon para que unicamente quede la
                                                                 # cantidad en la columna de Precio1
    datos["Precio2"]=datos["Precio2"].replace("M","",regex=True) # En caso de tener alguna letra M se reemplaza en cada renglon para que unicamente quede la
                                                                 #cantidad en la columna de Precio2

    datos["Precio1"]=datos["Precio1"].replace("X","",regex=True) # En caso de tener alguna letra X se reemplaza en cada renglon para que unicamente quede la
                                                                 #cantidad en la columna de Precio1
    datos["Precio2"]=datos["Precio2"].replace("X","",regex=True) # En caso de tener alguna letra X se reemplaza en cada renglon para que unicamente quede la
                                                                 #cantidad en la columna de Precio2
    datos["Precio1"]=datos["Precio1"].replace("N","",regex=True) # En caso de tener alguna letra N se reemplaza en cada renglon para que unicamente quede
                                                                 #la cantidad en la columna de Precio1
    datos["Precio2"]=datos["Precio2"].replace("N","",regex=True) # En caso de tener alguna letra N se reemplaza en cada renglon para que unicamente quede la
                                                                 # cantidad en la columna de Precio2

    datos['Precio1'] = pd.to_numeric(datos['Precio1'], errors='coerce') # Se convierten las cantidades de tipo cadena a tipo numerico de la columna Precio1
    datos['Precio2'] = pd.to_numeric(datos['Precio2'], errors='coerce') # Se convierten las cantidades de tipo cadena a tipo numerico de la columna Precio2

    datos.to_excel("df_productos_limpio.xlsx",index=False) # Se crea otro excel con las columnas de Precio1 y Precio2 ya modificadas

    print(datos.dtypes) # Se visualizan los tipos de datos de cada columna
    return datos
```

Ahora se aplica la función precios floats al DataFrame donde se encuentran todos los productos de las 3 páginas web, el DataFrame actualizado es concatenado en un nuevo Excel por último se imprime el DataFrame ya con las cantidades en tipo numérico y sin los símbolos y/o letras sobrantes

```
precios_floats(df_productos) # Se aplica la funcion pre
df_productos=pd.read_excel("df_productos_limpio.xlsx")
print(df_productos) # Se imprime el DataFrame ya con la
```

Por último, realizamos nuestras 8 consultas del tipo SQL sobre el ultimo DataFrame

(el que tiene las cantidades en tipo numérico) que lleva por nombre ***df_productos***

```
print("Consultas SQL")
print(ps.sqldf("select Autoservicio, Precio1, Precio2 from df_productos where(Producto='pantalon') and (Precio2 is not null) and (Precio1<800)"))
print(ps.sqldf("select count(*) from df_productos where(Autoservicio='PH') and (599<Precio1) and (Producto='playera')"))
print(ps.sqldf("select sum(Precio1) as sumatotal from df_productos where(Producto='abrigo') and (Precio2 is not null) and(1500<Precio1) and (Autoservicio='PH'
print(ps.sqldf("select Producto, Precio1 from df_productos where(Autoservicio='SRS') and (Producto='pantalon') and (Precio1 between '200' and '500')"))
print(ps.sqldf("select count(*) from df_productos where(Autoservicio='CPP' or Autoservicio='PH') and (Precio2 is not null)"))
print(ps.sqldf("select Precio2 from df_productos where(Autoservicio='SRS' or Autoservicio='PH') and (Precio2<999)"))
print(ps.sqldf("select avg(Precio1) as preciopromedio from df_productos where(Autoservicio='PH') and (Producto='pantalon')"))
print(ps.sqldf("select Producto from df_productos where(Autoservicio='SRS' or Autoservicio='CPP' or Autoservicio='PH') and (Precio2 between '1500' and '3000'
```

Conclusiones

El internet es sin duda enorme, y cada segundo se agrega nuevo contenido a él. Si deseamos realizar búsquedas de forma repetitiva, hacerlo de forma manual no es una opción. Lo mejor que podemos hacer es simplemente apoyarnos de un web scraper el cual realizará misma tarea en mucho menos tiempo y sin mucho esfuerzo y que mejor aún si nosotros mismos lo desarrollamos.