

Centro de investigación en computación Instituto politécnico nacional

python científico 2023

15 de julio 2023

Profesor asignado: Alan Badillo Salas

Alumno: Sergio Antonio Velázquez Cauich
correo: sergiov7035@outlook.es

Práctica 108
**Extraer los productos de la página de sanborns usando
selenium**



Introducción:

El problema consiste en usar herramientas de *web scraping* para sacar información específica en este caso de la página web de sanborns, durante el curso se utilizó la herramienta scrapy y *scrapydo* para el scraping, sin embargo este ejercicio consiste en la investigación de un nuevo plugin éste siendo *selenium* el cual trabaja de forma similar a *scrapydo*

La mayor diferencia entre *scrapy* y *selenium* es el hecho que *selenium* es una herramienta de automatización web **que puede ser utilizada** para *scraping*, mientras que *scrapy* es una herramienta exclusivamente dedicada a *scraping*.

los rubros que este ejercicio presenta son los siguientes:

- Abrir el navegador usando selenium
- Visita la página de sanborns
- Busca los nodos donde está la información de cada producto
- Recolecta la información de esos nodos
- Guarda la información en un dataframe
- Guarda la información en un archivo CSV
- Cierra el navegador en Selenium

los cuales voy a desglosar a lo largo del reporte

Justificación:

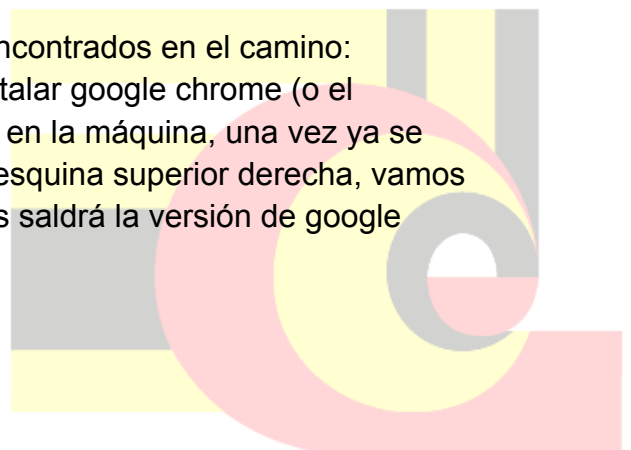
Selenium tiene muchas aplicaciones a diferencia de *Scrapydo*, ya que éste puede hacer de forma automatizada cualquier cosa que utilice un navegador web (no necesariamente chrome), básicamente “simulando” una interacción con un usuario, este puede ser usado para probar las capacidades de una página de forma automática por ejemplo.

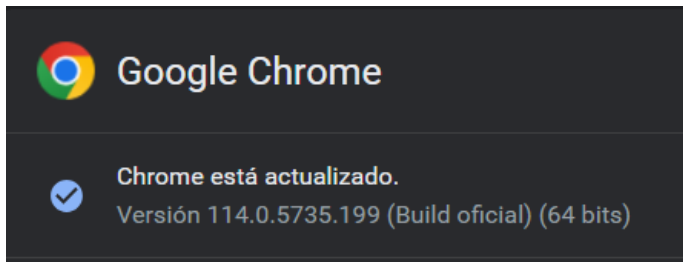
sin embargo el ejercicio se enfoca en el *scraping*, el cual consiste en la recolección de datos de páginas web, esto tiene un sinnúmero de utilidades, tal como la obtención automatizada de datos disponibles en bases abiertas para un estudio, estudiar el comportamiento en los precios, disponibilidad, venta de productos de empresas rivales, estudiar y recopilar mediante redes sociales información de usuarios, entre otras.

Resolución:







empecemos con los paso a paso y los problemas encontrados en el camino:

1: aunque suene trivial, recomiendo bastante en instalar google chrome (o el navegador que se piense usar) aunque ya se tenga en la máquina, una vez ya se tenga instalada damos clic en los tres puntos en la esquina superior derecha, vamos a ayuda y clic en “acerca de google chrome” ahí nos saldrá la versión de google chrome,





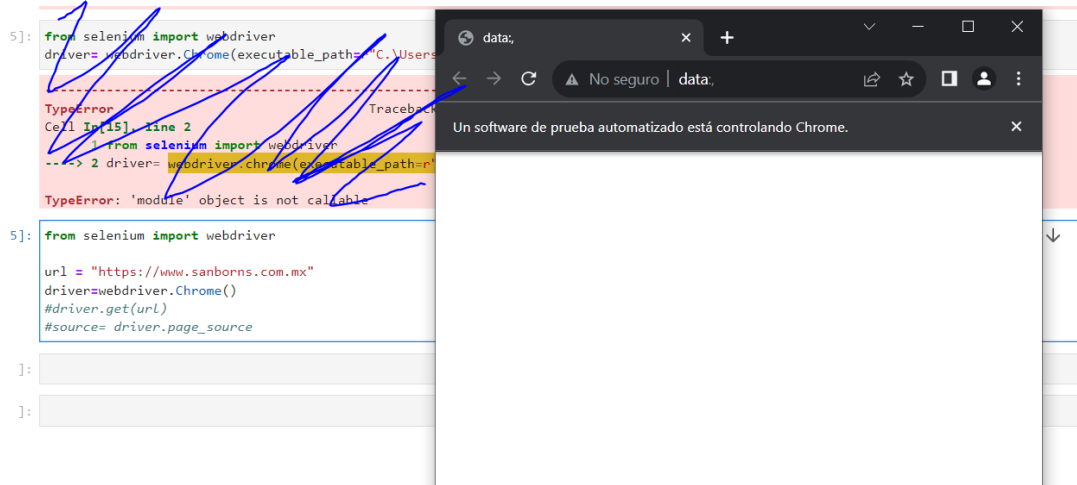
esto es importante ya que procederemos a instalar los drivers del navegador que pretendemos usar, y el sistema en este caso de chromium.org, procedemos a instalar la versión que coincida con la de nuestro navegador (o la más reciente en mi caso)

 Parent Directory				
	chromedriver_linux64.zip	2023-05-31 08:57:22	7.06MB	cd6613edf6628041684393706b62d3a6
	chromedriver_mac64.zip	2023-05-31 08:57:25	8.29MB	b44390afbdddaf8748a1d151483b2472
	chromedriver_mac_arm64.zip	2023-05-31 08:57:29	7.40MB	0d515e46bea141705e49edaba1d49819
	chromedriver_win32.zip	2023-05-31 08:57:32	6.30MB	7d455bed57ef682d41108e13d45545ca
	notes.txt	2023-05-31 08:57:38	0.00MB	1670f6dde7877ca84ecd4c56b9cc759c

Ahora que ya tenemos instalado todo, podemos iniciar con el código en python empezamos instalando el selenium con pip install e importamos selenium y el paquete “webdriver” el cual es la interfaz que simula a un usuario, después guardamos como variable el lin de la página que buscamos scrapear, en mi caso el link de sanborns en variable “URL”.

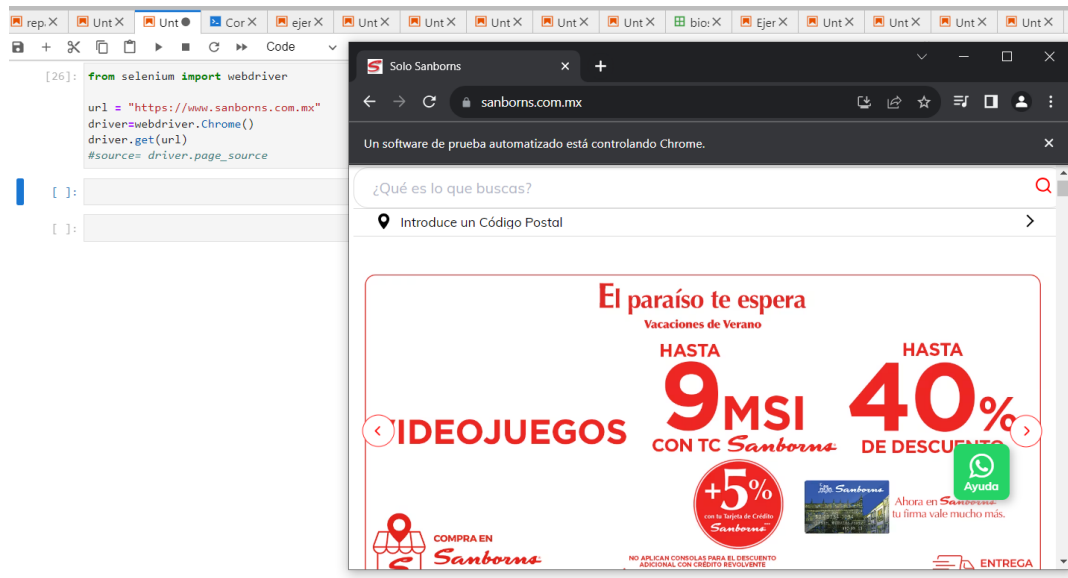
en una variable que vamos a llamar driver vamos a guardar el navegador que queremos usar y su dirección hacia su driver.

Ya con esto nuestro código abrirá el navegador sin problemas:



Ahora como lo que buscamos es que nos abra la página de sanborns a continuación vamos a atribuirle la variable URL a nuestro navegador el cual si recordamos tiene el link de nuestra página, esto con un get().

Ahora ya tenemos abierta nuestra página web de preferencia:



Sin embargo si buscamos extraer de la página vamos a ocupar el código fuente de dicha página así que lo extraemos y se lo atribuimos a una variable llamada source:

```
from selenium import webdriver

url = "https://www.sanborns.com.mx"
driver=webdriver.Chrome()
driver.get(url)
source= driver.page_source
```

me gustaría explicar algunas observaciones en el paso de buscar el driver, en este paso encontré múltiples vías para esto, entre ellas encontré una librería llamada webdriver_manager, el cual es bastante útil ya que este buscará y cargará automáticamente la versión más reciente del driver del navegador que se pretende utilizar:

los pasos a seguir son similares solo se instalará esta librería igual con pip install y después utilizaremos esta librería para importar los drivers automáticamente, y al momento de apuntar el origen de los drivers se utilizará un comando de esta nueva librería:

```
!pip install webdriver_manager
from webdriver_manager.chrome import ChromeDriverManager

import selenium
from selenium import webdriver
driver = webdriver.Chrome(ChromeDriverManager().install())
```

En esta captura podemos ver como funciona, esta opción puede ayudar a automatizar aún más el proceso ya que esta librería siempre va a procurar descargar la versión más reciente de los drivers en caso de actualización, haciendo posible que el algoritmo corra cada determinado tiempo con un mínimo de necesidad de asistencia humana.

Los siguientes pasos se necesita localizar y extraer nodos de la página, esto con localizadores de Selenium, en concreto de xpath si no me equivoco, sin embargo ha sido difícil para mi seguir las guías por mi poca familiaridad con html sin embargo en mi tiempo libre va a ser algo que estudie ya que la automatización con selenium se me hizo bastante interesante y con muchos usos.

