



Practica: 06
Nombre de la Practica: Selenium y
Beautiful Soap

Instituto Politécnico Nacional.
Escuela Superior de Cómputo.
Licenciatura en ciencia de datos.

Nombre de la materia: Desarrollo de Aplicaciones para el Análisis de Datos
Grupo: 4AV1
Profesora: Sandra Luz Morales Guitron

Lopez Mendez Emiliano

INDICE

INTRODUCCION	3
DESAROLLO	3
Web scraping 3: selenium y beautifulsoup	3
Web scraping 4: configurar chrome y scraping con selenium.....	7
CONCLUSIONES	11

INTRODUCCION

En esta práctica, profundizamos en el uso de herramientas para la automatización y el web scraping: Selenium y BeautifulSoup. Ambas permiten interactuar con páginas web y extraer información estructurada o dinámica. El objetivo fue entender cómo interactuar con sitios web que dependen de JavaScript para cargar contenido y cómo manejar interacciones complejas, como clics y desplazamientos. Estas habilidades son fundamentales en tareas como la extracción de datos dinámicos, análisis de mercado y automatización de tareas rutinarias en la web.

DESAROLLO

Herramientas principales

1. **Selenium:** Utilizado para automatizar la interacción con navegadores web. Esencial para manejar contenido dinámico y realizar acciones en la página como clics y desplazamientos.
2. **BeautifulSoup:** Permite analizar y extraer información específica del HTML generado por Selenium. Fue crucial para transformar el contenido en datos procesables.
3. **webdriver_manager:** Facilitó la instalación y configuración automática del controlador de Chrome para Selenium, asegurando compatibilidad con la versión del navegador.
4. **Requests:** Aunque no fue el foco principal, esta librería fue utilizada como referencia para entender cómo interactuar con el contenido estático de la web en otros contextos.

Web scraping 3: selenium y beautifulsoup

```
import requests
from webdriver_manager.chrome import ChromeDriverManager
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from bs4 import BeautifulSoup
```

✓ 0.0s

import requests:

- La librería requests se utiliza para realizar solicitudes HTTP.

from webdriver_manager.chrome import ChromeDriverManager:

- webdriver_manager automatiza la descarga y configuración del controlador para el navegador Chrome. Esto evita la necesidad de instalar manualmente el archivo chromedriver, asegurando que la versión del controlador sea compatible con el navegador instalado.

from selenium import webdriver:

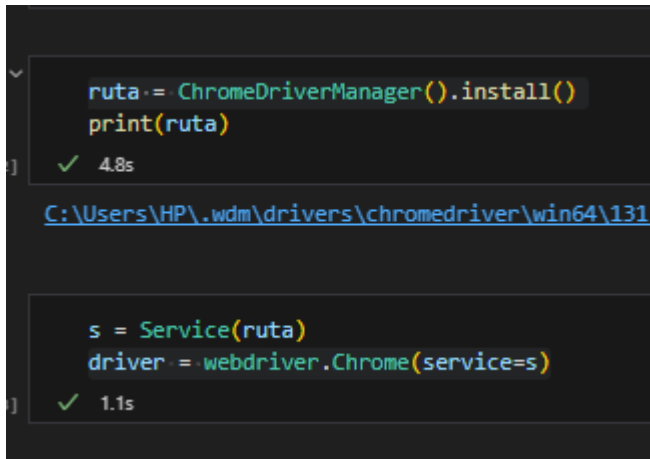
- selenium.webdriver es el módulo principal de Selenium para controlar navegadores web. Este módulo permite abrir, navegar e interactuar con páginas web de manera automatizada.

from selenium.webdriver.chrome.service import Service:

- La clase Service se usa para definir y gestionar el servicio del controlador del navegador (en este caso, Chrome). Facilita el manejo del proceso subyacente que comunica Selenium con el navegador.

from bs4 import BeautifulSoup:

- BeautifulSoup es una librería para analizar documentos HTML y XML.



```
ruta = ChromeDriverManager().install()
print(ruta)

C:\Users\HP\.wdm\drivers\chromedriver\win64\131...

s = Service(ruta)
driver = webdriver.Chrome(service=s)
```

ChromeDriverManager().install():

- Utiliza webdriver_manager para descargar automáticamente la versión adecuada de chromedriver compatible con el navegador Chrome instalado
- Devuelve la ruta donde se encuentra el ejecutable chromedriver.

print(ruta):

- Muestra en consola la ruta del ejecutable chromedriver. Esto es útil para confirmar que se ha descargado correctamente y para verificar dónde se encuentra el archivo.

s = Service(ruta):

- Crea un servicio basado en la ruta del archivo chromedriver. Este servicio es responsable de iniciar y gestionar el proceso del controlador del navegador.

driver = webdriver.Chrome(service=s):

- Inicializa una instancia del navegador Chrome utilizando Selenium. La instancia está vinculada al servicio creado en la línea anterior.
- Una vez ejecutado, abre una ventana del navegador

```
url = "https://www.mercadolibre.com.mx/apple-iphone-16-pro-max-  
driver.get(url)  
✓ 1.0s  
  
driver.page_source  
soup = BeautifulSoup(driver.page_source, "html.parser")  
print(soup)  
✓ 0.3s
```

- **Primera parte:** Carga la página web de MercadoLibre en el navegador mediante Selenium.
- **Segunda parte:** Procesa el HTML de la página cargada con BeautifulSoup para facilitar la extracción de información estructurada.

Esto te prepara para buscar datos específicos dentro del HTML, como el nombre del producto, su precio o cualquier otra información relevante.

Ya de aquí hicimos como en la practica anterior, obtuvimos el nombre del producto, precio, ranking de usuarios, imagen del producto y sus características

```
soup.find("h1", class_="ui-pdp-title").text  
✓ 0.0s  
'Apple iPhone 16 Pro Max (256 GB) - Titanio natural - Distribuidor Autorizado'  
  
precio = soup.find("span", class_="andes-money-amount ui-pdp-price__part andes-money-amount--cents-superscript andes-money-amount--  
print(precio)  
precio_float = float(precio)  
print(precio_float)  
✓ 0.0s  
30999  
30999.0  
  
soup.find("span", class_="ui-pdp-review__rating").text  
✓ 0.0s  
'4.8'
```

```

✓ soup.find(class_="ui-pdp-image ui-pdp-gallery__figure__image").attrs
✓ 0.0s

{'data-zoom': 'https://http2.mlstatic.com/D_NQ_NP_768410-MLU78891930890_092024-F.webp',
 'data-index': '0',
 'width': '410',
 'height': '500',
 'fetchpriority': 'high',
 'decoding': 'sync',
 'src': 'https://http2.mlstatic.com/D_NQ_NP_768410-MLU78891930890_092024-O.webp',
 'srcset': 'https://http2.mlstatic.com/D_NQ_NP_2X_768410-MLU78891930890_092024-F.webp 2x',
 'class': ['ui-pdp-image', 'ui-pdp-gallery__figure__image'],
 'alt': 'Apple iPhone 16 Pro Max (256 GB) - Titanio natural - Distribuidor Autorizado'}
+ Code - + Markdown

✓ soup.find(class_="ui-pdp-image ui-pdp-gallery__figure__image").attrs.get("src")
✓ 0.0s

'https://http2.mlstatic.com/D_NQ_NP_768410-MLU78891930890_092024-O.webp'

1] ✓ soup.find("ul", class_="ui-vpp-highlighted-specs__features-list").text
✓ 0.0s

'Memoria RAM: 8 GBMemoria interna: 256 GBImponente diseño de titanio.Diseñado para Appl

2] ✓ result = soup.find("ul", class_="ui-vpp-highlighted-specs__features-list").text
formatted_result = result.replace(".", ".\n")
print(formatted_result)
✓ 0.0s

Memoria RAM: 8 GBMemoria interna: 256 GBImponente diseño de titanio.
Diseñado para Apple Intelligence.
Control de la cámara.
Capturas soñadas.
Estilos fotográficos.
La potencia del chip A18 Pro.
Un salto enorme en batería.
Personaliza tu iPhone.
Funcionalidades esenciales de seguridad.

✓ soup.find("div", class_="ui-pdp-description pl-24 pr-24 ui-pdp-collapsible--is-collapsed").text
✓ 0.0s

'DescripcióniPhone 16 Pro Max. Diseñado para Apple Intelligence. Imponente diseño de titanio. Control de

```

Web scraping 4: configurar chrome y scraping con selenium

```
import requests
from webdriver_manager.chrome import ChromeDriverManager
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
```

import requests:

- Importa la librería requests, que se utiliza para realizar solicitudes HTTP.

from webdriver_manager.chrome import ChromeDriverManager:

- Importa el gestor de controladores para Chrome. Este módulo permite descargar y configurar automáticamente la versión correcta del controlador chromedriver compatible con el navegador Chrome instalado.

from selenium import webdriver:

- Importa el módulo principal de Selenium que permite controlar navegadores web. Es el componente que se utiliza para abrir, navegar e interactuar con páginas web.

from selenium.webdriver.chrome.service import Service:

- Importa la clase Service, que se utiliza para definir y gestionar el servicio que ejecuta el controlador del navegador Chrome. Es una forma más estable y modular de iniciar el controlador.

from selenium.webdriver.chrome.options import Options:

- Importa la clase Options, que permite configurar las opciones del navegador Chrome. Esto se usa, por ejemplo, para ejecutar el navegador en modo headless (sin interfaz gráfica) o para agregar extensiones.

from selenium.webdriver.common.by import By:

- Importa el módulo By, que proporciona métodos para localizar elementos en una página web. Esto incluye búsquedas por ID, nombre de clase, etiqueta, XPath, entre otros.

```
def iniciar_chrome():
    ruta = ChromeDriverManager().install()
    options = Options()
    user_agent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)"
    options.add_argument(f"user-agent={user_agent}")
    # options.add_argument("--window-size=1000,1000")
    options.add_argument("--headless")
    options.add_argument("--start-maximized")
    options.add_argument("--disable-web-security")
    options.add_argument("--disable-extension")
    options.add_argument("--disable-notifications")
    options.add_argument("--ignore-certificate-errors")
    options.add_argument("--no-sandbox")
    options.add_argument("--log-level=3")
    options.add_argument("--allow-running-insecure-content")
    options.add_argument("--no-default-browser-check")
    options.add_argument("--no-first-run")
    options.add_argument("--no-proxy-server")
    options.add_argument("--disable-blink-features=AutomationControlled")

    exp_opt = [
        'enable-automation',
        'ignore-certificate-errors',
        'enable-logging'
    ]
    options.add_experimental_option("excludeSwitches", exp_opt)

    prefs = {
        "profile.default_content_values_notifications": 2,
        "intl.accept_languages": ["es-ES", "es"],
        "credentials_enable_service": False
    }
    options.add_experimental_option("prefs", prefs)

    s = Service(ruta)
    driver = webdriver.Chrome(service=s, options=options)
    return driver
```

Esta function hace varias cosas:

- Descarga automáticamente la versión correcta del controlador chromedriver para el navegador Chrome instalado en el sistema y almacena su ruta.
- Crea un objeto Options que permite configurar diversas características y comportamientos del navegador Chrome.
- Configura el User-Agent del navegador para simular una solicitud proveniente de un navegador real. Esto ayuda a evitar bloqueos o restricciones de algunos sitios web.
- **--headless**: Ejecuta el navegador sin interfaz gráfica (modo "headless").
- **--start-maximized**: Abre el navegador en pantalla completa.
- **--disable-web-security**: Desactiva las políticas de seguridad web, útil para pruebas locales.
- **--disable-extension**: Desactiva las extensiones instaladas en Chrome.
- **--disable-notifications**: Desactiva las notificaciones emergentes.
- **--ignore-certificate-errors**: Ignora errores de certificados SSL.

- **--no-sandbox**: Desactiva el sandboxing de Chrome, mejorando compatibilidad en entornos restringidos.
- **--disable-blink-features=AutomationControlled**: Elimina características de automatización detectables, reduciendo la probabilidad de ser identificado como un bot.
- **excludeSwitches**: Excluye ciertos switches de comando para evitar que el navegador sea detectado como automatizado.
- **profile.default_content_values_notifications**: Desactiva las notificaciones del navegador.
- **intl.accept_languages**: Configura el idioma preferido del navegador (español en este caso).
- **credentials_enable_service**: Desactiva el gestor de credenciales de Chrome.
- **Service(ruta)**: Inicializa el servicio del controlador Chrome utilizando la ruta obtenida anteriormente.
- **webdriver.Chrome(service=s, options=options)**: Crea una instancia del navegador Chrome con las opciones configuradas.
- Devuelve el objeto driver, que es la instancia del navegador configurado y listo para su uso.

```
if __name__ == '__main__':
    driver = iniciar_chrome()
    url = "https://www.mercadolibre.com.mx/apple-iphone-16-pro-max-256-gb-titanio-natural-distribuidor"
    driver.get(url)

    nombre_producto = driver.find_element(By.CSS_SELECTOR, "h1.ui-pdp-title").text
    print(f'Nombre_producto: {nombre_producto}')

    precio = driver.find_element(By.CSS_SELECTOR, "span.andes-money-amount__fraction").text
    print(f'Precio_producto: {precio}')

    caract = driver.find_element(By.CSS_SELECTOR, "section.ui-vpp-highlighted-specs").text
    print(f'Caracteristicas_producto: {caract}')

    imagen = driver.find_element(By.CSS_SELECTOR, "img.ui-pdp-image").get_attribute("src")
    print(f'URL de la imagen del producto: {imagen}')

    driver.quit()
```

- Llama a la función `iniciar_chrome()` (definida previamente) para configurar e iniciar una instancia del navegador Chrome controlado por Selenium.
- Define la URL del producto en MercadoLibre.
- Usa `driver.get(url)` para abrir la página web en el navegador automatizado.
- Usamos `driver.find_element()` con un selector CSS (`By.CSS_SELECTOR`) para localizar el elemento HTML `<h1>` con la clase `ui-pdp-title`.
- Extraemos el texto del elemento (el nombre del producto) y lo imprimimos.
 - Hacemos eso para cada elemento que quisimos sacar de la página web y del producto

29 de noviembre de 2024

```
9.6s
Nombre_producto: Apple iPhone 16 Pro Max (256 GB) - Titanio natural - Distribuidor Autorizado
Precio_producto: 30,999
Caracteristicas_producto: Lo que tienes que saber de este producto
Memoria RAM: 8 GB
Memoria interna: 256 GB
Imponente diseño de titanio.
Diseñado para Apple Intelligence.
Control de la cámara.
Capturas soñadas.
Estilos fotográficos.
La potencia del chip A18 Pro.
Un salto enorme en batería.
Personaliza tu iPhone.
Funcionalidades esenciales de seguridad.
Ver características
URL de la imagen del producto: https://http2.mlstatic.com/D\_Q\_NP\_768410-MLU78891930890\_092024-R.webp
```

Finalmente aquí podemos ver los prints que hicimos

CONCLUSIONES

Esta práctica destacó la versatilidad de Selenium y BeautifulSoup en el análisis de datos web dinámicos. Aprendimos a manejar contenido cargado por JavaScript y a interactuar con elementos complejos de una página. El uso combinado de estas herramientas nos permitió superar las limitaciones de métodos más básicos, como Requests.

Además, esta experiencia reforzó la importancia de respetar las políticas de uso de los sitios web y mantener un enfoque ético en la recolección de datos. Con estas técnicas, estamos mejor preparados para enfrentar problemas reales en el ámbito del análisis y desarrollo basado en datos.