Exercici 2:
Memoria del Ejercicio de Web Scraping con Selenium y BeautifulSoup

Librerías Utilizadas:

- pandas
- BeautifulSoup
- Selenium
- Requests
- Webdriver de Chrome, Keys, Options
- Time
- Scrapy (no utilizado)

Exercici 1.- Web Scraping con BeautifulSoup y Selenium

1.1 Web Scraping con BeautifulSoup

1.1.1 Origen de la Información:

- *Url_quotes:* "http://quotes.toscrape.com"
  Citas inspiradoras de varios autores con etiquetas que describen el tema o contexto.
- Url_wikipedia: "https://es.wikipedia.org/wiki/Barcelona" (Búsqueda “Barcelona").
  Datos climáticos del Observatorio del Aeropuerto Josep Tarradellas Barcelona-El Prat (1991-2020).

1.1.2 Ejecución:

- ***Primera web:*** Utilización de Selenium para navegar, extraer datos y crear DataFrame con citas inspiradoras.
- ***Segunda web:*** Web scraping de la página de Wikipedia sobre Barcelona usando BeautifulSoup en Python.

1.1.3 Resultados:

- ***Primera web:*** DataFrame con 3 columnas (Frase, Autor, Etiquetas) y 10 filas, exportado a 'quotes_data_beautifulsoup_tot.csv'.

- ***Segunda web:*** DataFrame exportado a 'wiki_data_beautifulsoup.csv' con datos climáticos mensuales.

1.2 Web Scraping con Selenium de Ambas Webs

1.1.1 Origen de la Información:

- *Url_quotes:* "http://quotes.toscrape.com"
  Citas inspiradoras de varios autores con etiquetas que describen el tema o contexto.
- Url_wikipedia: "https://es.wikipedia.org/wiki/Barcelona" (Búsqueda “Barcelona").
  Datos climáticos del Observatorio del Aeropuerto Josep Tarradellas Barcelona-El Prat (1991-2020).

1.2.2 Ejecución:

- *Primera web:* Selenium navega, extrae citas, y almacena en un DataFrame exportado como 'quotes_data_selenium.csv'.
- *Segunda web:* Selenium configura opciones headless, extrae y procesa datos climáticos, creamos el DataFrame y exportamos com ‘wiki_data_selenium.csv’.

1.2.3 Resultados:

- *Primera web:* Datos extraídos y almacenados en 'quotes_data_selenium.csv'.
- *Segunda web:* Datos climáticos en DataFrame exportado como ‘wiki_data_selenium.csv'.

1.3 Comparación de BeautifulSoup y Selenium:

- *Selenium:* Navega en un navegador real o sin head, interactúa con páginas dinámicas y ejecuta JavaScript.
- *BeautifulSoup:* Analiza HTML estático, no ejecuta JavaScript ni interactúa dinámicamente.

Consideraciones:
Selenium es más versátil pero más lento y pesado.
BeautifulSoup es más ligero y rápido para páginas estáticas.

Exercici 3.- Web Scraping con Scrapy

No utilizado, ya que no era necesario ni compatible para la web seleccionad: "https://datosmacro.expansion.com/mercado-laboral/salario-medio".

3.2 Web Scraping con BeautifulSoup

3.2.1 Selección de la Página:

Url_smi: "https://datosmacro.expansion.com/mercado-laboral/salario-medio"

Datos sobre el salario medio en diferentes países.

3.2.2 Ejecución con BeautifulSoup:

Se realizó una solicitud HTTP a la URL para obtener el contenido de la página. Luego, con BeautifulSoup, se buscaron todas las tablas y se extrajeron datos. Sin embargo, se realizaron correcciones para ajustar la estructura a la original de la tabla.

3.2.3 Correcciones Realizadas:

- La fila del encabezado es la que contiene los nombres de las columnas (actualmente en la posición 1), ya que se crearon índices de columnas y filas que no están en la tabla original.
- La columna 3, inicialmente "Salario Medio Mon. Local", se ajustó para contener la divisa que acompaña a la columna 2 actual.
- La columna 4, inicialmente "salario medio", se modificó para reflejar el contenido de la columna 5 original.
- La columna 5, inicialmente "var", se movió a la última posición y se le asignó el contenido de la columna 7 original.
- Se eliminó la columna 6.

3.2.4 Resultados y Exportación:

El DataFrame resultante se guardó en un archivo CSV llamado 'smi_beautifulsoup.csv'. Además, se realizó una limpieza adicional, eliminando corchetes y espacios innecesarios en la columna 'Países', y se exportó una versión mejorada del DataFrame.

3.3 Web Scraping con Selenium:

Se realizó web scraping de la misma página utilizando Selenium para obtener la información directamente. Se aplicaron correcciones similares para adaptar la estructura de la tabla a la original.

3.3.1 Correcciones y Exportación:

No fueron necesarios los ajustes para reflejar la estructura original de la tabla. El DataFrame resultante se exportó a un archivo CSV llamado 'smi_selenium.csv'.

3.4 Comparación de Resultados:

Ambos enfoques con BeautifulSoup y Selenium proporcionaron resultados similares, capturando datos sobre el salario medio en diferentes países desde la página web proporcionada. Con Selenium no tuve los problemas de desplazamiento de columnas