WEB SCRAPING CON PYTHON

Ejercicios de Web Scraping con Python

🔧 Requisitos previos

- Python instalado.
- Librerías: `requests`, `beautifulsoup4`, `lxml` o `html.parser`.

Instalación (si no están):

```
pip install requests beautifulsoup4 lxml
```

📘 Ejercicio 1: Extraer titulares de una web de noticias

Objetivo: Obtener los títulos de las últimas noticias de https://elpais.com o similar.

Instrucciones:

1. Usar `requests` para obtener el HTML.
2. Parsear con BeautifulSoup.
3. Imprimir los titulares (por ejemplo, etiquetas `<h2>`).

Código base:

```
import requests
from bs4 import BeautifulSoup

url = "https://elpais.com/"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

titulos = soup.find_all('h2')

for i, titulo in enumerate(titulos[:10]):
    print(f"{i+1}. {titulo.get_text(strip=True)}")
```

📑 Ejercicio 2: Obtener precios de productos (simulación de ecommerce)

Objetivo: Extraer nombre y precio de productos desde una tienda como https://books.toscrape.com

Instrucciones:

1. Scrapea título del libro, precio y disponibilidad.
2. Muestra al menos 5 productos.

Código base:

```
import requests
from bs4 import BeautifulSoup

url = "https://books.toscrape.com/"
res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

libros = soup.select("article.product_pod")

for libro in libros[:5]:
    titulo = libro.h3.a['title']
    precio = libro.select_one(".price_color").text
    print(f"{titulo} - {precio}")
```

📅 Ejercicio 3: Recoger eventos de una página (tipo agenda cultural)

Objetivo: Acceder a una web como https://www.malaga.es/agenda/ y obtener los eventos del día.

Instrucciones:

- Scrapea título, fecha y descripción breve.
- Permite guardar los resultados en un `.txt`.

- para realizar este ejercicio hay que usar selenium, debido a que la web de la agenda cultural está realizada con javascript y eso impide que se pueda usar los ejercicios anteriores.
- pip install selenium, también hay que importar webdriver: pip install selenium webdriver-manager.

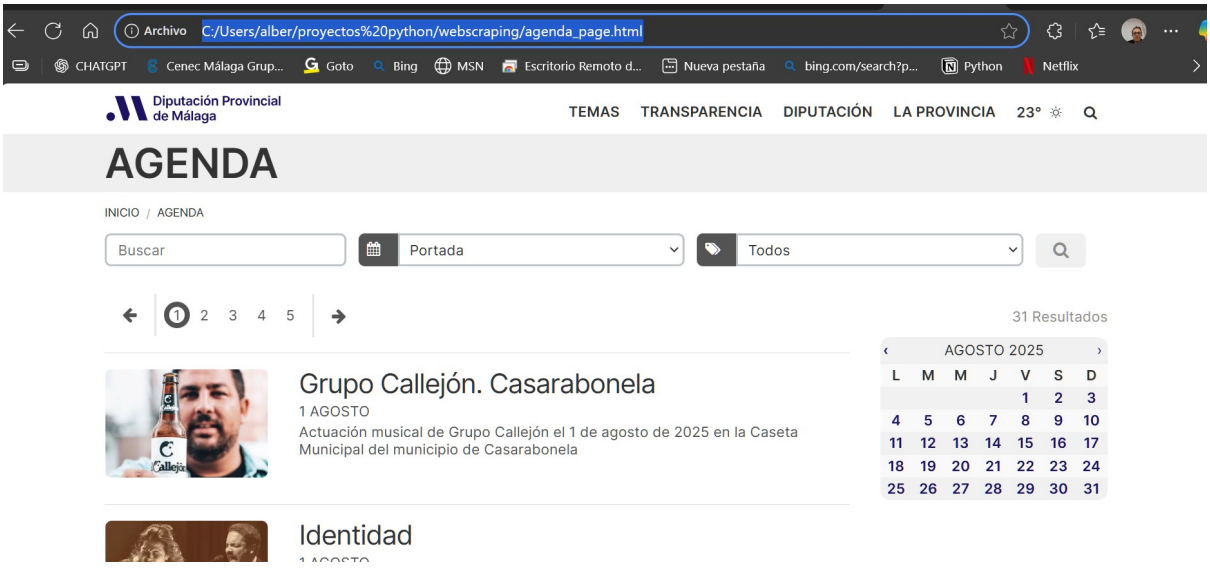
  Solución

```
import os

import time



from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.chrome.service import Service

from webdriver_manager.chrome import ChromeDriverManager



options = Options()

options.add_argument("--headless")

options.add_argument("--window-size=1920,1080")



driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install())
, options=options)

driver.get("https://www.malaga.es/agenda/")

time.sleep(5)



html = driver.page_source



ruta = os.path.join(os.getcwd(), "agenda_page.html")

print(f"Guardando HTML en: {ruta}")
```

```python
try:
    with open(ruta, "w", encoding="utf-8") as f:
        f.write(html)
        print("✅ HTML guardado correctamente.")
except Exception as e:
    print("❌ Error al guardar el archivo:", e)

driver.quit()
```

Genera un fichero en esta ruta

file:///C:/Users/alber/proyectos%20python/webscraping/agenda_page.html



📊 Ejercicio 4: Guardar resultados en CSV

Modifica el **Ejercicio 2** para guardar los resultados en un archivo `.csv`.

Pista: Usa la librería `csv` de Python.

Solución:

```python
import csv

import requests
from bs4 import BeautifulSoup

url = "https://books.toscrape.com/"
res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

libros = soup.select("article.product_pod")

# Creamos el archivo CSV
with open("libros.csv", "w", newline="", encoding="utf-8") as f:
    writer = csv.writer(f)
    writer.writerow(["Título", "Precio"])  # Encabezados

    for libro in libros[:5]:
        titulo = libro.h3.a['title']
        precio = libro.select_one(".price_color").text
        writer.writerow([titulo, precio])

print("✅ Datos guardados en libros.csv")
```

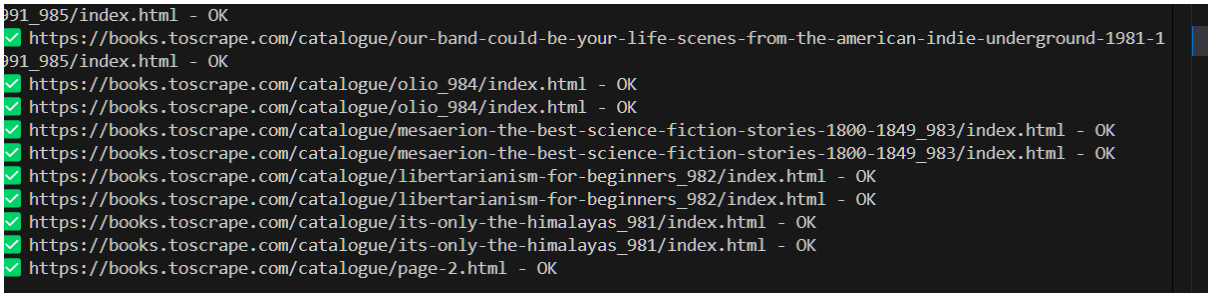
🧪 Ejercicio 5: Detección de enlaces rotos

Objetivo: Comprobar si los enlaces de una web funcionan.

Instrucciones:

1. Extrae todos los `<a href="...">`.
2. Usa `requests.head()` para comprobar si el enlace devuelve un código 200.

3.



Ejemplo básico comprueba solo un enlace de example.com

#!/usr/bin/env python3

```
# -*- coding: utf-8 -*-

import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urldefrag
import sys

# URL base: si pasas una URL por argumento la usa, si no usa example.com
BASE_URL = sys.argv[1] if len(sys.argv) > 1 else "https://example.com"

def fetch_html(url):
    """Descarga el HTML de la URL dada y lo devuelve como texto."""
    r = requests.get(url, timeout=10)
    r.raise_for_status()  # si no es 2xx lanza excepción
    return r.text

def extract_links(base_url, html):
    """
    Extrae todos los <a href="...">, resuelve enlaces relativos a absolutos
    y elimina fragmentos (#lo-que-sea).
    """
    soup = BeautifulSoup(html, "html.parser")
    links = []
    for a in soup.select('a[href]'):
        href = a.get("href", "").strip()
        # Ignorar mailto:, tel:, javascript:, y anclas puras '#'
        if not href or href.startswith(("mailto:", "tel:", "javascript:", "#")):
            continue
        # Convertir relativo -> absoluto
        absolute = urljoin(base_url, href)
        # Quitar el fragmento (parte tras #)
        absolute, _ = urldefrag(absolute)
        links.append(absolute)

    # Quitar duplicados manteniendo orden
    seen = set()
    unique = []
    for u in links:
        if u not in seen:
            seen.add(u)
            unique.append(u)
    return unique
```

```python
def check_link_head(url):
    """Hace HEAD al enlace y devuelve (status_code, ok_200)."""
    try:
        r = requests.head(url, allow_redirects=True, timeout=10)
        return r.status_code, (r.status_code == 200)
    except requests.RequestException:
        return None, False  # error de red, DNS, timeout, etc.

def main():
    print(f"Analizando enlaces en: {BASE_URL}\n")

    try:
        html = fetch_html(BASE_URL)
    except Exception as e:
        print(f"[ERROR] No se pudo descargar {BASE_URL}: {e}")
        sys.exit(1)

    links = extract_links(BASE_URL, html)
    print(f"Encontrados {len(links)} enlaces.\n")

    ok, fail = 0, 0
    for url in links:
        status, is_ok = check_link_head(url)
        if is_ok:
            ok += 1
            print(f"[OK  ] 200 -> {url}")
        else:
            fail += 1
            st = status if status is not None else "ERROR"
            print(f"[FAIL] {st} -> {url}")

    print("\nResumen:")
    print(f"  OK (200): {ok}")
    print(f"  No 200 / error: {fail}")

if __name__ == "__main__":
    main()
```

books.toscrape.com

```
nt https://books.toscrape.com?
~/Documents/IFCT0109/Cristobal/PYTHON ❯ python3 Ejercicio_scrap3.py
Analizando enlaces en: http://books.toscrape.com

Encontrados 73 enlaces.

[OK  ] 200 -> http://books.toscrape.com/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books_1/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/travel_2/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/mystery_3/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/historical-fiction_4/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/sequential-art_5/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/classics_6/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/philosophy_7/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/romance_8/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/womens-fiction_9/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/fiction_10/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/childrens_11/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/religion_12/index.html
[OK  ] 200 -> http://books.toscrape.com/catalogue/category/books/nonfiction_13/index.html
```

💡 Propuesta de proyecto final:

Crea un script que:

1. Busque información meteorológica (por ejemplo desde https://www.tiempo.com o similar).

2. Extraiga temperatura, estado del tiempo y ciudad.

3. Guarde los datos en un CSV para análisis posterior.