[Actividad extracurricular 12] web scraping

David Pilataxi

2025-08-01

Tabla de Contenidos

Web Scrapping

- David Pilataxi
- Gr1cc
- 8 de enero de 2025

Enlace al Repositorio

LINK: https://github.com/DavidPilataxi/MetodosNumericosGr1cc/blob/main/Actividad%20Extra%2012%20W

1. Objetivos

- Revisar qué es web scraping
- Realizar una prueba en python para dos librerías diferentes
- Realizar scraping de un sitio web de su elección

2. Introducción

Es el proceso de extraer datos de sitios web de manera automatizada utilizando herramientas o scripts. Esto se logra accediendo al contenido de las páginas web, ya sea analizando el HTML, extrayendo datos estructurados, o interactuando con elementos dinámicos.

Aplicaciones comunes de web scraping:

- Análisis de precios en sitios de comercio electrónico.
- Monitorización de noticias o contenidos en tiempo real.
- Recolección de información para proyectos de investigación o estudios de mercado.
- Extracción de datos de directorios en línea o bases de datos accesibles públicamente.

Herramientas populares para web scraping:

- BeautifulSoup (Python) para extraer datos de HTML.
- Selenium para interactuar con páginas dinámicas.
- Scrapy, un framework avanzado para tareas de scraping masivo.

3. Procedimiento

Web Scraping: BeautifulSoup vs Scrapy

Configuración inicial

- !pip install requests beautifulsoup4 pandas scrapy

3.1. Web Scraping con BeautifulSoup. -

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL del sitio web
url = "https://www.python.org/blogs/"

# Enviar la solicitud HTTP
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    soup = BeautifulSoup(response.text, "html.parser")

    # Ajustar los selectores para títulos y enlaces
    titles = [title.text.strip() for title in soup.select("li h3")]
    links = [link['href'] for link in soup.select("li h3 a")]

    # Crear DataFrame
    data = pd.DataFrame({"Título": titles, "Enlace": links})
```