Clase 048 — Web scraping con BeautifulSoup

Parte: 0 — Prerrequisitos · Fuente: Web Scraping with Python (Mitchell, 2ª ed.) caps. 1-3 · BeautifulSoup docs. · Duración estimada: 75 min.

Clase 048 — Web scraping con BeautifulSoup

> *Parte: 0 — Prerrequisitos · Fuente: Web Scraping with Python (Mitchell, 2ª ed.) caps. 1-3 · BeautifulSoup docs. · Duración estimada: 75 min.*

Objetivo

Que el alumno extraiga datos de páginas HTML cuando no hay API disponible, usando requests + BeautifulSoup. Y entienda los límites éticos y legales: robots.txt, rate limiting humano, ToS, datos personales, copyright. Lo último que debe hacer al scrapear es tirar abajo el sitio o meterse en problemas.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Parsear HTML con BeautifulSoup(html, 'html.parser').
2. Encontrar elementos con find, find_all, select (CSS selectors).
3. Extraer texto y atributos (.text, ['href']).
4. Respetar robots.txt y rate limit (delay entre requests).
5. Identificar cuándo scraping es buena idea vs cuándo buscar otra fuente (API, dataset público).

Temas

#	Tema	Por qué importa
1	HTTP → HTML → parser tree	Cómo funciona scraping.
2	BeautifulSoup: find vs select	Selectores CSS son más potentes.
3	Extracción de texto y atributos	.text, .get_text(strip=True), ['href'].
4	Páginas dinámicas (JS) — requests no las r	Para eso: Playwright/Selenium.
5	robots.txt — qué dice y por qué respetar	User-agent, Disallow, Crawl-delay.
6	Ética: ToS, rate limiting, datos personale	Lo que sí, lo que no.

Definiciones y características

Web scraping

: Extracción programática de datos de páginas HTML cuando no hay API. Pipeline típico: descargar HTML con requests, parsear con BeautifulSoup, extraer con selectores.

BeautifulSoup

: Parser HTML tolerante a errores. soup = BeautifulSoup(html, 'html.parser'). Permite navegar el árbol y buscar por tag, atributo o selector CSS.

CSS selector

: Sintaxis para localizar elementos: '.class', '#id', 'tag', 'parent > child', 'tag[attr=val]'. Usados con soup.select(...). Más expresivos que find_all.

DOM (Document Object Model)

: Representación en árbol del HTML. Cada tag es un nodo; tiene padre, hermanos, hijos. BeautifulSoup navega este árbol con .parent, .next_sibling, .find_all, etc.

robots.txt

: Archivo en raíz del dominio (/robots.txt) que declara qué paths pueden crawlear los bots. Estándar de facto; respetarlo es legalmente importante (variable por jurisdicción) y éticamente siempre.

JS rendering

: Páginas SPA (React, Vue) cargan contenido vía JavaScript tras el HTML inicial. requests solo trae el HTML inicial — JS no se ejecuta. Solución: Playwright o Selenium (navegador headless).

Dataset / recursos

Página HTML simple servida desde un string en el notebook (sin tocar internet). Ejercicios opcionales con https://quotes.toscrape.com (sitio diseñado para practicar).

Ejercicios

1. Parsea HTML local. Crea un HTML con 3 productos ( ). Extrae nombres y precios con find_all.

2. Selectores CSS. Lo mismo con soup.select('.product .price').

3. Tabla a DataFrame. pd.read_html(url) para una tabla HTML — bonus: requests + BeautifulSoup para tablas custom.

4. Scrape ético. Scrapea quotes.toscrape.com (público, diseñado para esto). Respeta Crawl-delay. 3 páginas con time.sleep(1) entre cada una.

5. Inspeccionar robots.txt. Lee https://quotes.toscrape.com/robots.txt con requests. Identifica qué paths están Disallow.

Homework verificable

Notebook: (a) HTML local con 5 productos, extrae nombre/precio/url; (b) scrape quotes.toscrape.com (3 páginas, con delay); (c) consulta robots.txt y razona; (d) listado de 3 escenarios cuando scrapear es buena idea y 3 cuando no.

Criterio de aceptación: Scraping respeta delays. Análisis de robots.txt correcto.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
soup.find('div') devuelve None aunque hay	Buscas un atributo específico que no match
Scrapeo y recibo HTML distinto al que veo	El sitio renderiza con JavaScript. request
HTTP 403 Forbidden	El sitio detecta tu bot (User-Agent vacío
Site funciona en navegador pero scraper de	Anti-bot agresivo (Cloudflare, reCAPTCHA).
Encoding raro (acentos Ã¡)	Pandas/requests dedujo encoding mal. Fix:

Preguntas frecuentes

¿Scraping es legal?

Depende: jurisdicción, ToS del sitio, naturaleza del dato. Datos personales = casi siempre regulado (GDPR/LGPD). Datos públicos sin ToS prohibitivo = generalmente OK. Consulta abogado para casos serios.

¿find_all o select?

select (CSS selectors) — más potente, sintaxis estándar web, más legible. find_all para casos simples o cuando integras con código heredado.

¿Cómo descargo imágenes?

r = requests.get(url_imagen); open('img.jpg', 'wb').write(r.content). Para muchas, usa Session + thread pool.

¿Scrapy vs BeautifulSoup?

BeautifulSoup: librería de parsing. Una página, una request. Scrapy: framework completo (crawler, pipelines, throttling). Para proyectos serios (miles de páginas), Scrapy.

¿Y si el sitio me bloquea?

Respeta. Aumentar agresividad (proxies, rotating User-Agents) puede ser ilegal en algunas jurisdicciones (CFAA en USA). Considera: ¿realmente vale la pena? ¿hay otra fuente?

Referencias

- Mitchell, Web Scraping with Python 2e, caps. 1-3.
- BeautifulSoup docs
- quotes.toscrape.com (sitio para practicar)
- Google — robots.txt

Siguiente clase

Clase 049 — async / httpx / aiohttp para data scientists

Apéndice: notebook (primer bloque)

bash pip install beautifulsoup4 lxml

```
import requests
import time
from bs4 import BeautifulSoup
print(f'bs4 OK')
```

Archivos complementarios

- notebook.ipynb