7. **Web Scraper simple:** Extrae datos de una página web específica (`requests` y `BeautifulSoup`).

¡Claro que sí! Con gusto te ayudaré a crear un *web scraper* simple usando Python.
Vamos a generar un único archivo de Python que utilizará las librerías requests para descargar el contenido HTML de una página web (usaremos el sitio de documentación de Python como ejemplo) y BeautifulSoup para analizar (parsear) ese HTML y extraer información específica.
Aquí tienes el código:

```
# -*- coding: utf-8 -*-
import requests
from bs4 import BeautifulSoup

# URL de ejemplo para el scraping. Usaremos un sitio seguro y con propósitos educativos.
# En este caso, la documentación oficial de Python es una fuente estable.
URL_A_SCRAPEAR = 'https://docs.python.org/3/library/random.html'

def web_scraper_simple(url):
  """
  Realiza un scraping simple: descarga la página y extrae los títulos (<h1>, <h2>)
  y los enlaces (<a>) encontrados en el contenido.

  Args:
    url (str): La URL de la página web a scrapear.
  """
  print(f"Iniciando scraping de: {url}\n")

  try:
    # 1. Realizar la petición HTTP para obtener el contenido de la página
    # El parámetro 'timeout' evita que la petición se quede colgada indefinidamente.
    response = requests.get(url, timeout=10)

    # Levantar una excepción para códigos de estado HTTP erróneos (4xx o 5xx)
    response.raise_for_status()

    # 2. Inicializar BeautifulSoup con el contenido y el parser HTML
    # El método .content asegura que manejamos bytes (útil para codificación)
    soup = BeautifulSoup(response.content, 'html.parser')

    # --- Extracción de Títulos (Encabezados) ---
    print("-" * 50)
    print("TÍTULOS ENCONTRADOS (H1 y H2):")
    print("-" * 50)

    # Buscar todos los elementos <h1> y <h2> en la página
    # La función find_all devuelve una lista de objetos Tag
    titles = soup.find_all(['h1', 'h2'])

    if titles:
      for title in titles:
        # .get_text() extrae el texto visible dentro de la etiqueta
        print(f"[{title.name.upper()}]: {title.get_text(strip=True)}")
```

```
    else:
        print("No se encontraron elementos <h1> o <h2>.")

    # --- Extracción de Enlaces ---
    print("\n" + "=" * 50)
    print("ENLACES (URLs) ENCONTRADOS:")
    print("=" * 50)

    # Buscar todos los elementos <a>
    links = soup.find_all('a')

    # Usamos un conjunto (set) para evitar enlaces duplicados
    unique_links = set()

    for link in links:
        # Extraer el atributo 'href' (la URL del enlace)
        href = link.get('href')

        # Asegurarse de que el enlace existe y no está vacío
        if href:
            # Opcional: imprimir el texto del enlace y su destino
            link_text = link.get_text(strip=True)

            # Añadir a nuestro conjunto de enlaces únicos
            unique_links.add((link_text, href))

    # Imprimir los enlaces únicos
    if unique_links:
        for text, url in sorted(list(unique_links)):
            # Limitar la longitud del texto impreso para mejor formato
            display_text = text if len(text) < 40 else text[:37] + '...'
            print(f"Texto: {display_text:<40} -> URL: {url}")
    else:
        print("No se encontraron enlaces (<a>) con atributo 'href'.")

  except requests.exceptions.RequestException as e:
    # Manejo de errores de red (DNS, conexión, timeouts, códigos 404/500, etc.)
    print(f"\n[ERROR de Conexión/Petición]: No se pudo acceder a la URL: {e}")
  except Exception as e:
    # Manejo de otros errores (p. ej., error de parsing)
    print(f"\n[ERROR General]: Ocurrió un error inesperado: {e}")
  finally:
    print("\nScraping finalizado.")

# Ejecutar el scraper
if __name__ == '__main__':
  # Necesitas instalar las librerías si aún no las tienes:
  # pip install requests beautifulsoup4
  web_scraper_simple(URL_A_SCRAPEAR)
```

Este archivo (simple_scraper.py) contiene una función que realiza los pasos fundamentales del *web scraping*:

1. **Petición:** Usa requests.get() para descargar la página de documentación del módulo random de Python.
2. **Parseo:** Inicializa BeautifulSoup para convertir el código HTML en un objeto de Python que se puede navegar fácilmente.
3. **Extracción:**
- Busca todas las etiquetas <h1> y <h2> (títulos) y extrae su texto.
- Busca todas las etiquetas <a> (enlaces) y extrae su atributo href (la URL de destino).
4. **Manejo de Errores:** Incluye bloques try...except para manejar problemas comunes como errores de conexión o que la página no exista (códigos 4xx o 5xx).
   Para ejecutarlo, asegúrate de tener las librerías instaladas:

   Bash
   pip install requests beautifulsoup4

   Y luego ejecuta el script:

   ```bash
   python simple_scraper.py

   Si quieres hacer más, házmelo saber. ¿Te gustaría intentar extraer un elemento específico, como el contenido de un *div* con una clase particular, o tal vez generar los resultados en un archivo CSV?
   ```