Practica 5— “Resúmenes deportivos en consola con Ollama”

Objetivo

Desarrollar un programa que:

1. Descargue y limpie el texto de una **URL de prensa deportiva en español**.
2. Envíe ese texto a un **modelo local de Ollama** para obtener un **resumen en español**.
3. Muestre el resultado **por consola** con formato claro (títulos y viñetas).

Requisitos previos (instalación)

- Tener Python operativo en el sistema.
- Tener Ollama en ejecución (puerto 11434) y un modelo descargado (por ejemplo, “llama3.2”).
- Instalar las dependencias de scraping y del SDK de Ollama.

(No incluyas código: explica en tu README cómo verificaste cada requisito.)

Páginas sugeridas (España, deportes)

- AS (as.com)
- MARCA (marca.com)
- Mundo Deportivo (mundodeportivo.com)
- SPORT (sport.es)
- Opcionales: El País – Deportes (elpais.com/deportes/), Relevo (relevo.com)

Formato de salida esperado (en consola)

Debe verse **similar** a lo siguiente (el contenido variará según el día):

- Una línea de información previa con la URL y el tamaño del texto analizado (en caracteres).
- Un bloque con separadores arriba y abajo.
- Un título con el nombre del medio y la URL.

Estructura orientativa:

- [INFO] Resumiendo: <URL> (<N> chars)
- Separador con símbolos iguales.

- RESUMEN: <Nombre del medio> — <URL>
- Separador con símbolos iguales.
- Cuerpo con secciones (por ejemplo: “Titulares del día”, “Temas destacados”, “Otras secciones”) y viñetas.
- Separador de cierre.

Lo que debes construir (sin pegar soluciones)

1. **Descarga y limpieza de HTML**
    - Decide cabeceras HTTP (User-Agent), tiempo de espera y parser de HTML.
    - Elimina etiquetas irrelevantes (navegación, scripts, etc.).
    - Extrae texto legible del cuerpo de la página.
    - Limita la longitud del texto a un máximo razonable (y justifícalo).
2. **Diseño de prompts**
    - Crea un **system prompt** en español que pida un resumen breve en markdown e ignore navegación.
    - Crea un **user prompt** que incluya el título de la página y el texto extraído.
3. **Llamada al modelo de Ollama**
    - Envía los mensajes en formato de chat (system + user).
    - Decide si recibes la salida en streaming o completa (y justifica la elección).
    - Extrae solo el **contenido textual del resumen**.
4. **Impresión en consola**
    - Muestra el encabezado informativo.
    - Imprime un bloque con separadores, un título de resumen y el cuerpo en markdown con viñetas.
    - (Opcional) Guarda también el resumen a archivo .md con marca temporal.
5. **Ejecución con múltiples URLs**
    - Si no se reciben argumentos, usa **4 URLs por defecto** (medios deportivos españoles).
    - Si se reciben, procesa esas URLs.
6. **Manejo de errores**
    - Errores de red/HTTP.
    - Estructuras HTML no estándar o sin texto útil.

- Modelo no disponible o servidor de Ollama apagado.

Restricciones y ayudas (para evitar "copiar y pegar")

- **No se permite** entregar una sola función enorme: estructura el programa en funciones pequeñas y con nombres descriptivos.
- **Incluye comentarios breves** que expliquen cada decisión (máx. 1–2 líneas por decisión).
- **No uses librerías de "resumir páginas"** de terceros: el resumen debe venir del modelo de Ollama.
- **No hardcodees** un resumen fijo: debe generarse a partir del contenido descargado de cada URL.
- Debes **justificar** en tu README:
  - Elección del parser HTML.
  - Límite de caracteres del texto enviado al modelo.
  - Elección de streaming sí/no.
  - Lista de etiquetas eliminadas y por qué.

Entregables

1. Un archivo con tu **programa** (nombre sugerido: resumen_deportes.py).
2. Un **README.md** que incluya:
   - Instrucciones de instalación y ejecución (comandos exactos).
   - Explicación de las decisiones (parser, límite, streaming, limpieza).
   - Capturas de pantalla de **al menos 3 ejecuciones** con URLs distintas.
   - Respuestas a las **preguntas de reflexión** (ver abajo).
3. (Opcional) Una carpeta salidas/ con resúmenes .md generados.

Preguntas de reflexión (responder en el README)

1. ¿Por qué es útil **limitar** los caracteres del texto enviado al modelo? ¿Qué problema evita y qué pierdes al hacerlo?
2. ¿En qué casos preferirías **streaming** y en cuáles la **respuesta completa**?
3. ¿Qué problemas viste al limpiar HTML de estos medios y cómo los resolviste?
4. ¿Qué cambios introducirías en el prompt para mejorar la **fidelidad** o la **brevedad** del resumen?

5. ¿Cómo asegurarías que el modelo responda **siempre en español**, incluso si el texto de entrada está en otro idioma?