## SESIÓN OBTENCIÓN DE DATOS DESDE ARCHIVOS

**CONTENIDOS:**

- CSV: leyendo un archivo CSV.
- Escribiendo un archivo CSV.
- Archivos Excel: librería XLRD.
- Leyendo un archivo Excel.
- Escribiendo un archivo Excel.
- Leer tablas web con pandas (contenido opcional)

### CSV: LEYENDO UN ARCHIVO CSV

En ciencia de datos, la obtención y manipulación de datos es uno de los pasos más críticos en el flujo de trabajo. Los datos pueden provenir de diversas fuentes, como archivos CSV, hojas de cálculo de Excel, bases de datos o incluso tablas en páginas web.

A continuación, veremos cómo utilizar la librería Pandas para leer y escribir archivos CSV y Excel, así como a extraer datos de tablas web. Estos conocimientos te permitirán trabajar con datos almacenados en diferentes formatos y prepararlos para su análisis de manera eficiente.

#### ¿Qué es un archivo CSV?

Un archivo CSV (*Comma-Separated Values*) es un formato de texto plano que almacena datos en una estructura tabular, donde cada fila representa un registro y cada columna un campo.

Características principales de los archivos CSV:

- Formato ligero y ampliamente compatible con hojas de cálculo y bases de datos.
- Cada fila se separa con una nueva línea (\n).
- Los valores dentro de una fila se separan por comas “,” o por otro delimitador (como ; o |).
- No admite estilos ni fórmulas, solo datos en texto.

**Ejemplo de archivo CSV:**

```
Nombre,Edad,Puntaje
Juan,25,85
María,30,92
Pedro,27,78
```

*Ilustración 1 Ejemplo de archivo CSV*

**Leyendo un Archivo CSV con Pandas**

La función **read_csv()** de Pandas permite cargar datos desde un archivo CSV en un DataFrame, facilitando la manipulación y análisis de los datos.

**Ejemplo básico de lectura de un CSV:**

```python
import pandas as pd

# Leer el archivo CSV
df = pd.read_csv("datos.csv")

# Mostrar las primeras filas
print(df.head())
```

*Ilustración 2 Ejemplo de lectura de un CSV:*

**Parámetros Comunes de read_csv():**

- sep=",": Especifica el delimitador (por defecto es “,” pero puede ser “;”, “|”, etc.).
- header=0: Define la fila que se usará como encabezado (por defecto es la primera fila).
- index_col=0: Define qué columna se usará como índice del DataFrame.
- na_values=["NA", "NULL"]: Especifica valores que deben ser tratados como NaN (valores nulos).

**Ejemplo con parámetros personalizados:**

```
df = pd.read_csv("datos.csv", sep=";", index_col=0, na_values=["?", "NULL"])
```

*Ilustración 3 Ejemplo con parámetros personalizados*

**ESCRIBIENDO UN ARCHIVO CSV**

Una vez modificado un DataFrame en Pandas, se puede exportar nuevamente a un archivo CSV usando **to_csv().**

Ejemplo básico de escritura de un CSV:

```
df.to_csv('datos_guardados.csv', index=False)
```

*Ilustración 4 Ejemplo de escritura de un CSV:*

**Opciones útiles en to_csv():**

- index=False: Evita que Pandas guarde el índice del DataFrame en el archivo.
- sep=";": Cambia el delimitador a ; en lugar de la coma ,.
- na_rep="Desconocido": Define cómo se guardarán los valores nulos.

Ejemplo con más parámetros:

```
df.to_csv("nuevo_archivo.csv", index=False, sep=";", na_rep="Desconocido")
```

*Ilustración 5 Ejemplo con más parámetros*

## ARCHIVOS EXCEL: LIBRERÍA XLRD

Excel es uno de los formatos más utilizados en análisis de datos. Pandas facilita la lectura y escritura de archivos Excel sin necesidad de abrirlos manualmente.

### ¿Qué es XLRD?

XLRD es una librería de Python que permite leer archivos de Excel con extensión .xls (formato antiguo de Excel). Sin embargo, desde Pandas 1.2 en adelante, se recomienda usar openpyxl en lugar de XLRD para archivos .xlsx.

### Instalación de openpyxl (para leer archivos .xlsx)

```
pip install openpyxl
```

*Ilustración 6 Instalación de openpyxl*

## LEYENDO UN ARCHIVO EXCEL

La función **read_excel()** permite cargar datos desde una hoja de cálculo de Excel en un DataFrame.

Ejemplo básico de lectura de Excel:

```
df = pd.read_excel("datos.xlsx")
print(df.head())
```

*Ilustración 7 Ejemplo de lectura de Excel*

**Parámetros Comunes de read_excel()**

- sheet_name="Hoja1": Especifica la hoja a leer (por defecto es la primera).
- index_col=0: Define qué columna se usará como índice del DataFrame.
- usecols=["Nombre", "Edad"]: Permite seleccionar solo ciertas columnas.

Ejemplo con parámetros personalizados:

```
df = pd.read_excel("datos.xlsx", sheet_name="Ventas", usecols=["Cliente", "Total"], index_col="Cliente")
```

*Ilustración 8 Ejemplo con parámetros personalizados*

**ESCRIBIENDO UN ARCHIVO EXCEL**

Para exportar datos de Pandas a Excel, se usa **to_excel().**

Ejemplo básico de escritura en Excel:

```
df.to_excel("nuevo_archivo.xlsx", index=False)
```

*Ilustración 9 Ejemplo de escritura en Excel:*

**Opciones útiles en to_excel():**

- sheet_name="Resultados": Define el nombre de la hoja donde se guardarán los datos.
- na_rep="Sin datos": Define cómo se guardarán los valores nulos.

Ejemplo con parámetros personalizados:

```
df.to_excel("resultados.xlsx", sheet_name="Datos Analizados", index=False, na_rep="Sin datos")
```

*Ilustración 10 Ejemplo con parámetros personalizados*

## LEER TABLAS WEB CON PANDAS (CONTENIDO OPCIONAL)

Pandas también permite extraer datos de tablas HTML en páginas web usando **read_html(),** lo que facilita la recopilación de información desde la web.

### ¿Cómo funciona read_html()?

La función busca automáticamente todas las tablas de una página web y las convierte en una lista de DataFrames.

Ejemplo de extracción de datos web:

```
url = "https://es.wikipedia.org/wiki/Anexo:Países_y_territorios_dependientes_por_población"
tablas = pd.read_html(url)
df_poblacion = tablas [0] # Selecciona la primera tabla

# Mostrar la tabla encontrada
print(df_poblacion.head())
```

*Ilustración 11 Ejemplo de extracción de datos web*

La capacidad de leer y escribir datos desde diferentes fuentes es fundamental en ciencia de datos. Esta habilidad permite obtener datos de diversas fuentes y prepararlos para su análisis, lo que es esencial en cualquier proyecto de ciencia de datos.