“Instituto Tecnológico Superior de Lerdo”

Alumna: Aranza del Carmen Aranda Gonzalez

Núm. Control:232310674

Docente: Jesús Salas Marín

Carrera: ingeniería informática.

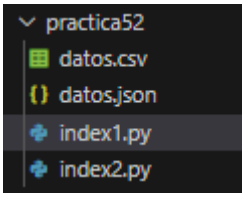
Documento: practica 5-2

Fecha: 3 de junio 2025

Lerdo, Dgo

“Desarrollo de la actividad”

Carpetas y archivos:



```python
import csv
import os

csv_file_path = os.path.join('practica52', 'datos.csv')

print(f"--- Manipulación de datos.csv ---")

try:
    with open(csv_file_path, mode='r', newline='', encoding='utf-8') as file:
        reader = csv.reader(file)
        header = next(reader)
        data_csv = list(reader)

    print(f"Encabezado del CSV: {header}")
    print(f"Primeras 5 filas del CSV:")
    for row in data_csv[:5]:
        print(row)


    try:

        columna_a_filtrar_nombre = 'Estado'

        if columna_a_filtrar_nombre in header:
            idx_columna_filtro = header.index(columna_a_filtrar_nombre)
            valor_a_filtrar = 'Activo'
            print(f"\nFiltrando filas donde la columna '{columna_a_filtrar_nombre}' es '{valor_a_filtrar}':")
            filas_filtradas = [row for row in data_csv if len(row) > idx_columna_filtro and row[idx_columna_filtro] == valor_a_filtrar]

            if filas_filtradas:
                for row in filas_filtradas[:5]:
                    print(row)
                print(f"... y {len(filas_filtradas)} filas en total cumplen el criterio.")
            else:
                print(f"No se encontraron filas con '{columna_a_filtrar_nombre}' igual a '{valor_a_filtrar}'.")
        else:
            print(f"Columna '{columna_a_filtrar_nombre}' no encontrada en el encabezado del CSV. No se pudo filtrar.")

    except IndexError:
        print("Error: El índice de columna para filtrar está fuera de rango. Revisa tu CSV.")
    except Exception as e:
        print(f"Ocurrió un error inesperado durante el filtrado: {e}")

except FileNotFoundError:
    print(f"Error: El archivo '{csv_file_path}' no se encontró. Asegúrate de que esté en la carpeta 'practica52'.")
except Exception as e:
    print(f"Ocurrió un error al leer el archivo CSV: {e}")

print(f"\nBeneficios observados para CSV en este escenario:")
print(f"- **Facilidad de Lectura/Escritura**: El módulo `csv` de Python facilita la lectura y escritura de datos tabulares de forma sencilla y directa.")
print(f"- **Eficiencia en Datos Tabulares**: Es muy eficiente para almacenar y procesar grandes volúmenes de datos donde cada registro tiene la misma estructura de columnas.")
print(f"- **Compatibilidad**: Es universalmente compatible con hojas de cálculo y muchas herramientas de análisis de datos.")
print(f"")
```

Que hace el primer código:

os.path.join('practica52', 'datos.csv'): Construye una ruta de archivo segura y compatible con el sistema operativo para datos.csv dentro de la carpeta practica52.

open(csv_file_path, mode='r', newline='', encoding='utf-8'): Abre el archivo CSV especificado en modo de lectura, asegurando el manejo correcto de líneas y codificación de caracteres, y garantizando que el archivo se cierre automáticamente.

csv.reader(file): Crea un objeto lector que interpreta las líneas del CSV como filas y los valores separados por coma como columnas, permitiendo iterar sobre el archivo.

next(reader): Lee la primera fila del CSV (generalmente el encabezado) y avanza el lector a la siguiente fila.

list(reader): Convierte el resto de las filas del CSV en una lista completa de listas, cargando todos los datos en memoria.

 header.index(columna_a_filtrar_nombre): Encuentra la posición (índice) de una columna específica dentro del encabezado del CSV.

[row for row in data_csv if ...] (List Comprehension): Filtra eficientemente las filas de los datos CSV basándose en una condición específica, creando una nueva lista solo con las filas que cumplen dicha condición.

```python
import json
import os
json_file_path = os.path.join('practica52', 'datos.json')

print(f" Manipulación de datos.json ")
try:
    with open(json_file_path, mode='r', encoding='utf-8') as file:
        data_json = json.load(file)

    print(f"Tipo de datos cargados del JSON: {type(data_json)}")

    if isinstance(data_json, list) and data_json:
        print(f"Primer elemento del JSON (si es una lista):")
        print(json.dumps(data_json[0], indent=2))


        if 'id' in data_json[0]:
            print(f"\nAccediendo al 'id' del primer elemento: {data_json[0]['id']}")
        if 'nombre' in data_json[0]:
            print(f"Accediendo al 'nombre' del primer elemento: {data_json[0]['nombre']}")


        if 'contacto' in data_json[0] and isinstance(data_json[0]['contacto'], dict):
            if 'email' in data_json[0]['contacto']:
                print(f"Accediendo al email anidado: {data_json[0]['contacto']['email']}")

    elif isinstance(data_json, dict):
        print(f"Contenido completo del JSON (si es un diccionario):")
        print(json.dumps(data_json, indent=2)) # Muestra el diccionario completo formateado

        if 'configuracion' in data_json:
            print(f"\nAccediendo a la clave 'configuracion':")
            print(json.dumps(data_json['configuracion'], indent=2))

    else:
        print("El contenido del JSON no es ni una lista ni un diccionario (o está vacío).")

except FileNotFoundError:
    print(f"Error: El archivo '{json_file_path}' no se encontró. Asegúrate de que esté en la carpeta 'practica52'.")
except json.JSONDecodeError:
    print(f"Error: El archivo '{json_file_path}' no es un JSON válido.")
except Exception as e:
    print(f"Ocurrió un error al leer el archivo JSON: {e}")

print(f"\nBeneficios observados para JSON en este escenario:")
print(f"- **Representación de Datos Complejos**: JSON es excelente para datos jerárquicos y anidados, como objetos con propiedades complejas y listas dentro de objetos.")
print(f"- **Intercambio de Datos en Web**: Es el formato estándar para APIs web, lo que facilita la integración con aplicaciones frontend (JavaScript) y backend.")
print(f"- **Legibilidad Humana**: A pesar de su complejidad, su estructura de clave-valor lo hace relativamente fácil de leer y entender para los desarrolladores.")
print(f"")
```

“código 2 de json”

os.path.join('practica52', 'datos.json'): Construye una ruta de archivo compatible con el sistema operativo para datos.json dentro de la carpeta practica52.

open(json_file_path, mode='r', encoding='utf-8'): Abre el archivo JSON en modo de lectura con codificación UTF-8 para su correcto manejo de caracteres.

json.load(file): Lee el contenido del archivo JSON y lo convierte automáticamente en un objeto de Python (diccionario si es un objeto JSON, o lista si es un array JSON), facilitando su manipulación en código.

isinstance(data_json, list) o isinstance(data_json, dict): Comprueba si el contenido JSON cargado es una lista o un diccionario de Python, lo que ayuda a determinar cómo procesar la estructura de datos.

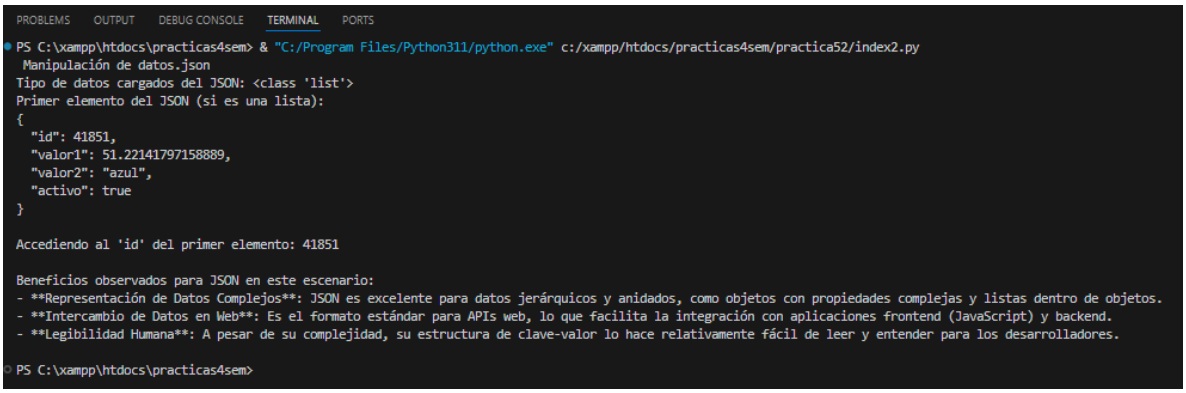
json.dumps(objeto, indent=2): Convierte un objeto de Python (diccionario o lista) de nuevo a una cadena de texto en formato JSON, con la opción de añadir indentación (indent=2) para mejorar la legibilidad.

objeto['clave'] o objeto['clave_padre']['clave_hija']: Permite acceder a los datos dentro del objeto Python (diccionario o lista) resultante de la carga JSON, incluyendo datos anidados, usando la sintaxis de corchetes.

“Resultados:”

Codigo1:

Del código 2:



“conclusiones”

Es vital saber manejar este tipo de archivos de datos: porque mas que nada se debe saber **Adaptarse y ser Versátiles**: Elegir el formato adecuado optimiza el intercambio de datos entre diferentes sistemas y aplicaciones. **Procesar Datos Eficientemente**: CSV es mejor para datos tabulares simples y grandes volúmenes, mientras que JSON es ideal para datos complejos y jerárquicos, lo que mejora el rendimiento del sistema.

Facilitar el Desarrollo: Ambos formatos son fáciles de manipular con código, lo que agiliza el desarrollo y reduce errores.

Integrarse con Tecnologías Modernas: JSON es estándar para APIs web, y CSV sigue siendo clave para hojas de cálculo y análisis, haciendo la integración más fluida.