📚 Clase 2: Manejo de Archivos CSV en Python

En esta guía aprenderás a leer, escribir, actualizar y añadir nueva información en archivos CSV usando Python. Los archivos CSV son ideales para almacenar datos tabulares, como inventarios de productos, y son comúnmente utilizados en ciencia de datos y análisis.

🚀 Introducción al Formato CSV

Un archivo CSV (Comma Separated Values) es un formato simple que permite almacenar datos en tablas, donde cada fila representa un registro y cada columna un atributo. Se utiliza ampliamente en aplicaciones de análisis de datos debido a su simplicidad y capacidad para manejar grandes volúmenes de datos de manera eficiente.

Ejemplo de archivo CSV:

Nombre	Precio	Cantidad	Marca	Categoría	Fecha de entrada
Laptop	1200	5	Dell	Electrónica	2023-08-01
Teclado	25	20	Logitech	Accesorios	2023-08-02

🛠️ Manipulación de Archivos CSV en Python

Para trabajar con archivos CSV en Python, utilizamos el módulo `csv` que está incluido por defecto. Este módulo nos permite realizar operaciones como lectura, escritura y actualización

de archivos CSV.

🔧 Importar la Librería CSV

```
import csv
```

1️⃣ Leer un Archivo CSV

El primer paso para manipular un archivo CSV es leerlo. Para esto, abrimos el archivo en modo lectura (`'r'`).

```
with open('productos.csv', mode='r') as archivo:
    lector_csv = csv.DictReader(archivo)
    for fila in lector_csv:
        print(fila)
```

Explicación:

- `DictReader` lee el archivo y lo convierte en un diccionario, donde las claves son los nombres de las columnas y los valores son los datos correspondientes.

Salida:

```
{'Nombre': 'Laptop', 'Precio': '1200', 'Cantidad': '5', 'Marca': 'Del
l', 'Categoría': 'Electrónica', 'Fecha de entrada': '2023-08-01'}
{'Nombre': 'Teclado', 'Precio': '25', 'Cantidad': '20', 'Marca': 'Logit
ech', 'Categoría': 'Accesorios', 'Fecha de entrada': '2023-08-02'}
```

2️⃣ Escribir en un Archivo CSV

Si queremos añadir información nueva a nuestro archivo CSV, lo abrimos en modo `'a'` (append).

```
nuevo_producto = {
    'Nombre': 'Cargador Inalámbrico',
    'Precio': '75',
    'Cantidad': '10',
    'Marca': 'Xiaomi',
    'Categoría': 'Accesorios',
    'Fecha de entrada': '2023-09-01'
}

with open('productos.csv', mode='a', newline='') as archivo:
```

```
    campos = ['Nombre', 'Precio', 'Cantidad', 'Marca', 'Categoría', 'Fe
cha de entrada']
    escritor_csv = csv.DictWriter(archivo, fieldnames=campos)
    escritor_csv.writerow(nuevo_producto)
```

Explicación:

- `DictWriter` permite escribir diccionarios en un archivo CSV.
- `fieldnames` define las columnas del archivo.
- `writerow` agrega una nueva fila con los datos del diccionario `nuevo_producto`.

3️⃣ Actualizar un Archivo CSV

Para actualizar información en un archivo CSV, es recomendable leer los datos actuales, realizar los cambios y luego escribir los datos modificados en un nuevo archivo.

```
with open('productos.csv', mode='r') as archivo:
    lector_csv = csv.DictReader(archivo)
    productos = list(lector_csv)

# Actualizar la cantidad de un producto
for producto in productos:
    if producto['Nombre'] == 'Laptop':
        producto['Cantidad'] = '8'

with open('productos_actualizados.csv', mode='w', newline='') as archiv
o:
    campos = ['Nombre', 'Precio', 'Cantidad', 'Marca', 'Categoría', 'Fe
cha de entrada']
    escritor_csv = csv.DictWriter(archivo, fieldnames=campos)
    escritor_csv.writeheader()
    escritor_csv.writerows(productos)
```

Explicación:

- Se lee el archivo original, se realiza una modificación en la cantidad del producto y se escribe la información en un nuevo archivo.

💡 Buenas Prácticas al Manipular Archivos CSV

1. **No sobrescribir los archivos originales**: Es recomendable crear una copia del archivo antes de realizar modificaciones. Esto asegura que los datos originales no se pierdan en

caso de errores.

2. **Validar los datos**: Asegúrate de que los datos que estás agregando o modificando tengan el formato correcto para evitar errores en la manipulación de los archivos.
3. **Usar el parámetro** `newline=''`: Al abrir archivos CSV, usa este parámetro para evitar que se inserten líneas en blanco adicionales al escribir en el archivo.

📝 Ejercicios Prácticos

🛒 Ejercicio 1: Gestión de Inventario

Imagina que trabajas en una tienda y necesitas gestionar el inventario de productos. Tienes un archivo CSV que almacena el nombre, precio y cantidad de cada producto. Crea un programa en Python que realice las siguientes operaciones:

1. Leer el archivo CSV y mostrar todos los productos en la tienda.
2. Agregar un nuevo producto al inventario.
3. Actualizar el precio de un producto existente.
4. Mostrar solo los productos cuyo precio sea mayor a 100.

💻 Solución

```
import csv

# Leer productos
def leer_productos():
    with open('productos.csv', mode='r') as archivo:
        lector_csv = csv.DictReader(archivo)
        for fila in lector_csv:
            print(f"Producto: {fila['Nombre']}, Precio: {fila['Preci
o']}, Cantidad: {fila['Cantidad']}")

# Agregar un nuevo producto
def agregar_producto(nombre, precio, cantidad, marca, categoria, fech
a):
    nuevo_producto = {
        'Nombre': nombre,
        'Precio': precio,
        'Cantidad': cantidad,
        'Marca': marca,
        'Categoría': categoria,
```

```
        'Fecha de entrada': fecha
    }
    with open('productos.csv', mode='a', newline='') as archivo:
        campos = ['Nombre', 'Precio', 'Cantidad', 'Marca', 'Categoría',
'Fecha de entrada']
        escritor_csv = csv.DictWriter(archivo, fieldnames=campos)
        escritor_csv.writerow(nuevo_producto)

# Actualizar el precio de un producto
def actualizar_precio(nombre_producto, nuevo_precio):
    with open('productos.csv', mode='r') as archivo:
        productos = list(csv.DictReader(archivo))

    for producto in productos:
        if producto['Nombre'] == nombre_producto:
            producto['Precio'] = nuevo_precio

    with open('productos.csv', mode='w', newline='') as archivo:
        campos = ['Nombre', 'Precio', 'Cantidad', 'Marca', 'Categoría',
'Fecha de entrada']
        escritor_csv = csv.DictWriter(archivo, fieldnames=campos)
        escritor_csv.writeheader()
        escritor_csv.writerows(productos)

# Mostrar productos con precio mayor a 100
def productos_precio_mayor_100():
    with open('productos.csv', mode='r') as archivo:
        lector_csv = csv.DictReader(archivo)
        for fila in lector_csv:
            if float(fila['Precio']) > 100:
                print(f"Producto: {fila['Nombre']}, Precio: {fila['Prec
io']}")

# Llamadas de ejemplo
leer_productos()
agregar_producto('Mouse', '15', '50', 'Logitech', 'Accesorios', '2024-0
9-15')
actualizar_precio('Laptop', '1300')
productos_precio_mayor_100()
```

🧱 Ejercicio 2: Cálculo del Valor Total del Inventario

Crea una columna adicional en el archivo CSV que calcule el valor total de cada producto (precio * cantidad).

💻 Solución

```
with open('productos.csv', mode='r') as archivo:
    lector_csv = csv.DictReader(archivo)
    productos = list(lector_csv)

# Añadir columna "Valor Total"
for producto in productos:
    producto['Valor Total'] = float(producto['Precio']) * int(producto
['Cantidad'])

with open('productos_con_valor_total.csv', mode='w', newline='') as arc
hivo:
    campos = ['Nombre', 'Precio', 'Cantidad', 'Marca', 'Categoría', 'Fe
cha de entrada', 'Valor Total']
    escritor_csv = csv.DictWriter(archivo, fieldnames=campos)
    escritor_csv.writeheader()
    escritor_csv.writerows(productos)
```

🎯 Conclusión

El manejo de archivos CSV en Python es una habilidad esencial, especialmente cuando trabajas con datos estructurados. Esta guía te proporciona las bases para leer, escribir y modificar archivos CSV, con prácticas comunes en el mundo real, como la gestión de inventarios y el análisis de datos. Además, dominar estas técnicas te permitirá integrar datos de diversas fuentes y facilitar la toma de decisiones informadas en proyectos y análisis.