**Manejo de Archivos**

Imaginemos que estamos desarrollando un sistema de gestión de una biblioteca digital. En este sistema, almacenaremos información sobre libros, autores y préstamos en archivos de texto. Usaremos estos archivos para almacenar, leer y manipular la información, manteniendo la persistencia de los datos.

**## Apertura y Cierre de Archivos**

Para interactuar con un archivo, primero debemos **\*\*abrirlo\*\*** y, una vez que hayamos terminado con él, **\*\*cerrarlo\*\***. Python ofrece la función integrada `open()` para abrir archivos y `close()` para cerrarlos. Existen diferentes modos para abrir un archivo:

- `"r"`: Modo de lectura (read).

- `"w"`: Modo de escritura (write).

- `"a"`: Modo de agregar (append).

- `"r+"`: Modo de lectura y escritura.

**### Ejemplo: Abrir y cerrar un archivo de texto**

En este ejemplo, abriremos un archivo llamado `libros.txt` que contiene una lista de libros en la biblioteca. Después de abrirlo, lo cerraremos.

# Apertura de un archivo en modo lectura

archivo = open("libros.txt", "r")

# Leer el contenido del archivo

contenido = archivo.read()

# Cerrar el archivo

archivo.close()

# Mostrar el contenido leído

print(contenido)

**## Uso de la declaración with**

Es una práctica común utilizar la declaración with al abrir archivos, ya que asegura que el archivo se cierre automáticamente, incluso si ocurre una excepción.

# Apertura de archivo con 'with'

with open("libros.txt", "r") as archivo:

    contenido = archivo.read()

# No es necesario cerrar el archivo manualmente

print(contenido)

**## Lectura y Escritura de Archivos**

Python nos permite leer y escribir archivos de varias formas. A continuación, veremos cómo leer todo el contenido de un archivo, cómo leer línea por línea, y cómo escribir en un archivo.

**### Lectura de un archivo completo**

Para leer todo el contenido de un archivo a la vez, usamos el método read().

with open("libros.txt", "r") as archivo:

    contenido = archivo.read()

    print(contenido)

**### Lectura línea por línea**

Podemos leer un archivo línea por línea utilizando el método readline() o iterando sobre el archivo directamente.

# Leer línea por línea con readline()

with open("libros.txt", "r") as archivo:

    linea = archivo.readline()

    while linea:

        print(linea.strip())  # Eliminamos los saltos de línea

        linea = archivo.readline()

**## Escritura en archivos**

El modo "w" se utiliza para escribir en archivos. Ten en cuenta que si el archivo ya existe, se sobreescribirá. Si no existe, se creará uno nuevo.

# Escritura en un archivo

with open("nuevos\_libros.txt", "w") as archivo:

    archivo.write("El Señor de los Anillos\n")

    archivo.write("1984\n")

    archivo.write("Cien Años de Soledad\n")

**### Agregar contenido a un archivo**

Para agregar contenido a un archivo sin sobrescribir lo que ya existe, utilizamos el modo "a" (append).

# Agregar nuevos libros al archivo

with open("nuevos\_libros.txt", "a") as archivo:

    archivo.write("Don Quijote de la Mancha\n")

    archivo.write("La Divina Comedia\n")

**## Operaciones con Archivos**

Además de leer y escribir, podemos realizar otras operaciones útiles con archivos, como contar líneas, buscar palabras, o trabajar con archivos CSV que almacenan datos en forma tabular.

**### Contar el número de líneas en un archivo**

def contar\_lineas(nombre\_archivo):

    with open(nombre\_archivo, "r") as archivo:

        lineas = archivo.readlines()

        return len(lineas)

# Contar líneas en 'libros.txt'

num\_lineas = contar\_lineas("libros.txt")

print(f"El archivo tiene {num\_lineas} líneas.")

**### Buscar una palabra en un archivo**

Supongamos que queremos buscar si un libro en particular está disponible en la biblioteca. Podemos buscar una palabra específica dentro del archivo.

def buscar\_palabra(nombre\_archivo, palabra):

    with open(nombre\_archivo, "r") as archivo:

        for linea in archivo:

            if palabra in linea:

                print(f"'{palabra}' encontrado en: {linea.strip()}")

                return True

    print(f"'{palabra}' no se encontró en el archivo.")

    return False

# Buscar un libro en el archivo

buscar\_palabra("libros.txt", "1984")

**### Leer y escribir archivos CSV**

Los archivos CSV son comunes para almacenar datos en formato tabular. Python ofrece el módulo csv para facilitar el trabajo con este tipo de archivos.

import csv

# Leer un archivo CSV de libros

with open("libros.csv", "r") as archivo\_csv:

    lector\_csv = csv.reader(archivo\_csv)

    for fila in lector\_csv:

        print(fila)

Podemos escribir en un archivo CSV de manera similar.

# Escribir en un archivo CSV

with open("nuevos\_libros.csv", "w", newline="") as archivo\_csv:

    escritor\_csv = csv.writer(archivo\_csv)

    escritor\_csv.writerow(["Título", "Autor", "Año"])

    escritor\_csv.writerow(["El Hobbit", "J.R.R. Tolkien", "1937"])

    escritor\_csv.writerow(["Fahrenheit 451", "Ray Bradbury", "1953"])

**### Copiar el contenido de un archivo a otro**

Podemos copiar el contenido de un archivo a otro utilizando la lectura y escritura.

def copiar\_archivo(archivo\_origen, archivo\_destino):

    with open(archivo\_origen, "r") as origen:

        contenido = origen.read()

    with open(archivo\_destino, "w") as destino:

        destino.write(contenido)

# Copiar contenido de 'libros.txt' a 'copias\_libros.txt'

copiar\_archivo("libros.txt", "copias\_libros.txt")

**## Desafíos**

**### Desafío 1: Crear un archivo de préstamos de libros**

Desarrolla un programa que cree un archivo prestamos.txt y permita al usuario agregar el registro de un préstamo. El registro debe incluir el nombre del libro, el nombre del prestatario y la fecha del préstamo. Asegúrate de no sobrescribir el archivo cada vez que se agrega un nuevo préstamo.

**### Desafío 2: Buscar un libro por autor en un archivo de texto**

Dado un archivo libros.txt que contiene una lista de libros y sus autores, implementa una función que busque todos los libros escritos por un autor específico y los muestre. Si el autor no tiene libros en la lista, debe mostrar un mensaje indicando que no hay coincidencias.

**### Desafío 3: Actualizar la cantidad de libros disponibles en un archivo CSV**

Tienes un archivo inventario.csv que contiene una lista de libros y el número de copias disponibles. Escribe un programa que permita actualizar la cantidad de copias de un libro específico. El programa debe leer el archivo, modificar el número de copias y volver a escribir el archivo.

**### Desafío 4: Eliminar registros de un archivo de préstamos**

Escribe un programa que permita eliminar el registro de un préstamo de un archivo prestamos.txt. El programa debe mostrar los registros actuales, permitir al usuario seleccionar cuál eliminar, y luego actualizar el archivo sin el registro eliminado.

**### Desafío 5: Contar las palabras más comunes en un archivo de texto**

Desarrolla una función que lea un archivo de texto grande, como libros.txt, y cuente cuántas veces aparece cada palabra. Luego, muestra las 5 palabras más comunes y cuántas veces aparecen.