Clase 13 – Archivos de texto en Python

Fundamentación

Hasta ahora los programas que hemos creado funcionaban mientras estaban en ejecución, pero al cerrarlos se perdía toda la información.
El **manejo de archivos** permite que los programas **guarden datos de manera persistente**.
Trabajar con archivos de texto (.txt) es el primer acercamiento al almacenamiento, y sienta las bases para comprender cómo funcionan luego las **bases de datos**.

Objetivos

- Comprender cómo abrir, leer y escribir archivos en Python.
- Diferenciar los modos de apertura:
    - "r" lectura
    - "w" escritura (sobrescribe)
    - "a" agregar (append).
- Utilizar correctamente open() y with open().
- Crear un programa que guarde y recupere información en un archivo de texto.

¿Por qué aprender esto?

Hasta ahora, los programas que escribimos en Python **olvidaban la información al cerrarse**.
Con los **archivos de texto (.txt)** podemos **guardar información de manera permanente** y volver a usarla después.

PYTHON INICIAL

Funciones básicas

 # Abrir un archivo
archivo = open("archivo.txt", "modo")

 # Escribir en el archivo
archivo.write("Texto\n")

 # Leer el archivo
contenido = archivo.read()

 # Cerrar el archivo
archivo.close()

Modos de apertura:

- `"r"` → lectura (read)
- `"w"` → escritura (write) → ⚠️ sobrescribe lo que ya había
- `"a"` → agregar (append) → suma al final del archivo

Primer contacto: escritura en archivos

Escribir en un archivo

```
archivo = open("ejemplo.txt", "w")

archivo.write("Hola, soy tu primer archivo en Python.\n")

archivo.close()
```

- ✔ Verificar en la carpeta que se creó el archivo.
- ✔ Abrirlo con un editor de texto (Bloc de notas o similar).

Leer un archivo

```
archivo = open("ejemplo.txt", "r")

contenido = archivo.read()

print(contenido)

archivo.close()
```

Agregar texto al final

```
archivo = open("ejemplo.txt", "a")
```

```
archivo.write("Nueva línea agregada.\n")

archivo.close()
```

Forma recomendada (con with)

```
with open("ejemplo.txt", "a") as f:
    f.write("Texto agregado de forma segura.\n")
```

Funciones de lectura en archivos

Cuando abrimos un archivo en **modo lectura ("r")**, podemos usar distintas funciones para recuperar su contenido:

1. read()

- **Lee todo el archivo completo** y lo devuelve como **una sola cadena de texto**.
- Si el archivo es muy grande, carga todo en memoria.

Ejemplo:

```
with open("ejemplo.txt", "r") as f:
    contenido = f.read()
    print(contenido)
```

Si ejemplo.txt tiene:

```
Primera línea
Segunda línea
Tercera línea
```

La salida será:

```
Primera línea
Segunda línea
Tercera línea
```

➔ Todo junto en un único string.

2. readline()

- **Lee una sola línea** del archivo cada vez que se llama.
- Incluye el salto de línea \n al final.
- Si llamamos varias veces, avanza línea por línea.

Ejemplo:

```
with open("ejemplo.txt", "r") as f:
    print(f.readline())  # Lee la primera línea
    print(f.readline())  # Lee la segunda línea
```

Salida:

Primera línea

Segunda línea

➔ Observa que aparece una línea en blanco porque readline() incluye el salto \n.

3. readlines()

- Lee **todas las líneas del archivo**, pero las devuelve en una **lista**, donde cada elemento es una línea.
- Útil para recorrer con un for.

Ejemplo:

```
with open("ejemplo.txt", "r") as f:
    lineas = f.readlines()
    print(lineas)
```

Salida:

['Primera línea\n', 'Segunda línea\n', 'Tercera línea']

➔ Cada elemento de la lista es una línea del archivo.

Comparación

Función	Devuelve	Uso típico
read()	Un **string** con todo el archivo	Cuando necesito todo el texto junto
readline()	Un **string** con **una sola línea**	Cuando leo línea por línea manualmente
readlines()	Una **lista** con todas las líneas	Cuando quiero recorrer el archivo con un bucle

✔ Consejo: en la práctica, muchas veces se usa directamente:

```
with open("ejemplo.txt", "r") as f:
    for linea in f:
        print(linea.strip())   # strip() quita el salto de línea
```

- ✔ Esta forma es más eficiente que readlines().

Actividades prácticas

Actividad 1 – Mi primera escritura

1. Crea un archivo llamado **mi_texto.txt**.
2. Escribe dentro la frase:

   "Hoy aprendí a guardar información con Python."
3. Abre el archivo con un editor de texto y verifica que esté la frase.

Actividad 2 – Leer el archivo

1. Abre **mi_texto.txt** en modo lectura.
2. Muestra el contenido en pantalla con print().

Actividad 3 – Agregar frases

1. Abre el mismo archivo en modo **"a"**.
2. Escribe **dos frases nuevas**.
3. Lee el archivo completo y verifica que ahora tiene 3 frases.

Actividad 4 – with open

1. Vuelve a agregar una línea, pero usando la forma segura:

```
with open("mi_texto.txt", "a") as f:
    f.write("Estoy aprendiendo a programar.\n")
```

2.Revisa que el archivo se haya actualizado.

5. Actividad final – Agenda simple

Crea un programa que funcione como una **agenda de notas**.

Debe permitir:

1. **Ingresar una nota** y guardarla en un archivo llamado agenda.txt.
2. **Leer y mostrar** todas las notas guardadas.

```
--- Agenda ---
1. Agregar nota
2. Ver notas
3. Salir
Elige una opción: 1
Escribe una nota: Comprar útiles

Elige una opción: 2
Tus notas:
- Comprar útiles
```

```
def guardar_nota():
    nota = input("Escribe una nota: ")
    with open("agenda.txt", "a") as f:
        f.write(nota + "\n")
    print("Nota guardada con éxito.")

def mostrar_notas():
    print("\n Tus notas:")
    with open("agenda.txt", "r") as f:
        for linea in f:
            print("- " + linea.strip())

# Programa principal
while True:
    print("\n--- Agenda ---")
    print("1. Agregar nota")
    print("2. Ver notas")
    print("3. Salir")
    opcion = input("Elige una opción: ")

    if opcion == "1":
        guardar_nota()
    elif opcion == "2":
        mostrar_notas()
    elif opcion == "3":
        print(" Adiós!")
        break
    else:
        print("Opción inválida. Intenta de nuevo.")
```

PYTHON INICIAL

Tarea Obligatoria:

Crea un archivo llamado recetas.txt y guarda tus 3 comidas favoritas. Luego haz un programa que las muestre en pantalla.