



Clase 1: Guía de Estudio: Manejo de Archivos **.TXT** en Python

1. Introducción al Manejo de Archivos en Python

En Python, se puede trabajar no solo con datos almacenados en variables, sino también con archivos externos, como archivos `.txt`, `.csv` o `.json`. Trabajar con archivos implica realizar varias operaciones, tales como:

- Leer archivos (read)
- Escribir archivos (write)
- Modificar archivos (append)

Estas operaciones permiten procesar la información contenida en un archivo, línea por línea o en su totalidad.



Tipos de Operaciones Comunes:

- **Lectura (Read):** Permite abrir un archivo y leer su contenido.
- **Escritura (Write):** Sobrescribe el contenido de un archivo.

- **Añadir (Append):** Añade contenido al final de un archivo sin modificar lo anterior.

2. Abrir y Leer Archivos

Para trabajar con archivos, se utiliza el método `open()`, que permite abrir un archivo en varios modos. Para leer un archivo:

```
archivo = open("caperucita.txt", "r") # "r" es el modo de lectura
contenido = archivo.read() # Lee todo el contenido del archivo
print(contenido)
archivo.close() # Es importante cerrar el archivo después de usarlo
```

Explicación del código:

- `"r"` indica que el archivo se abrirá en modo lectura.
- `read()` lee todo el contenido del archivo como un solo string.
- `close()` se usa para cerrar el archivo cuando ya no es necesario.

Ejemplo Práctico:

Supongamos que tienes un archivo `cuento.txt` que contiene el texto del cuento de "Caperucita Roja". El código anterior leería y mostraría todo el contenido de ese archivo.

3. Leer el Archivo Línea por Línea

Si se necesita procesar el archivo línea por línea, se puede hacer así:

```
archivo = open("caperucita.txt", "r")
for linea in archivo:
    linea_limpia = linea.strip() # Elimina espacios y saltos de línea
```

```
print(linea_limpia)
archivo.close()
```

Explicación del código:

- `for linea in archivo:` itera a través de cada línea del archivo.
- `strip()` elimina espacios y saltos de línea innecesarios.

Ejercicio Práctico:

Abre un archivo de texto que contenga la lista de compras y lee cada línea para mostrarla de manera ordenada:

```
archivo = open("lista_compras.txt", "r")
for item in archivo:
    print(f"Producto: {item.strip()}")
archivo.close()
```

4. Leer y Almacenar en una Lista

Para leer todas las líneas de un archivo y guardarlas en una lista:

```
archivo = open("caperucita.txt", "r")
lineas = archivo.readlines() # Almacena cada línea como un e
                              # lemento de una lista
print(lineas)
archivo.close()
```

Explicación del código:

- `readlines()` lee todas las líneas del archivo y las almacena en una lista, donde cada línea es un elemento.

Ejercicio Práctico:

Lee un archivo que contenga nombres de estudiantes y guárdalos en una lista:

```
archivo = open("nombres.txt", "r")
nombres = archivo.readlines()
print("Lista de estudiantes:", nombres)
archivo.close()
```

5. Escribir en un Archivo

El modo `"w"` permite escribir en un archivo, sobrescribiendo cualquier contenido existente:

```
archivo = open("nuevo_cuento.txt", "w")
archivo.write("Este es un nuevo cuento.\n")
archivo.close()
```

Explicación del código:

- `"w"` abre el archivo en modo escritura, y si el archivo no existe, lo crea.
- `write()` escribe una cadena de texto en el archivo. Se debe añadir un salto de línea `\n` si se quiere que el texto esté en varias líneas.

! Precaución:

El modo `"w"` **sobrescribe** todo el contenido del archivo.

6. Añadir Información a un Archivo +

Si deseas **añadir** contenido sin sobrescribir el archivo, se usa el modo `"a"`:

```
archivo = open("caperucita.txt", "a")
archivo.write("Fin del cuento. Escrito por ChatGPT.\n")
archivo.close()
```

Explicación del código:

- `"a"` abre el archivo en modo agregar (append), lo que permite añadir contenido al final del archivo sin borrar lo anterior.

7. Manipulación Avanzada de Archivos: `with`

Usar `with` para abrir archivos garantiza que el archivo se cierre automáticamente al terminar:

```
with open("caperucita.txt", "r") as archivo:
    contenido = archivo.read()
    print(contenido)
# No es necesario usar archivo.close(), ya que with lo hace automáticamente.
```

Ventaja:

- Es más seguro, ya que evita olvidarse de cerrar el archivo, lo que puede causar errores.

8. Ejercicios Prácticos Basados en la Vida Real

8.1. Contar Palabras en un Archivo de Texto

Este ejercicio consiste en leer un archivo de texto y contar cuántas palabras contiene.

```
def contar_palabras(archivo_nombre):
    with open(archivo_nombre, 'r') as archivo:
        contenido = archivo.read()
        palabras = contenido.split() # Divide el texto en palabras
    print(f"El archivo tiene {len(palabras)} palabras.")

contar_palabras("caperucita.txt")
```


8.2. Registro de Actividades de un Usuario en un Archivo de Log

Crea un archivo `log.txt` donde se registran las acciones de un usuario en un sistema:

```
def registrar_accion(accion):  
    with open("log.txt", "a") as archivo:  
        archivo.write(f"{accion}\\n")  
  
registrar_accion("Usuario inició sesión")  
registrar_accion("Usuario subió un archivo")  
registrar_accion("Usuario cerró sesión")
```

Conclusión

Este resumen detalla las operaciones básicas con archivos en Python, desde la lectura y escritura hasta el manejo de archivos en contextos más complejos. Estas herramientas te permitirán automatizar muchas tareas cotidianas relacionadas con la manipulación de archivos.

 **Clave:** Recuerda siempre cerrar los archivos o utilizar `with` para evitar problemas futuros.