

MANEJO DE ARCHIVOS CON PYTHON



■ Guía Paso a Paso: Implementación de un Context Manager para Manejo de Archivos en Python

✦ Introducción

En esta lección aprenderemos cómo crear una clase personalizada en Python para manejar archivos utilizando un **Context Manager**. Esta técnica nos permite abrir y cerrar archivos de manera automática, asegurando una correcta gestión de los recursos. Además, veremos cómo utilizar esta clase junto con la palabra clave `with` para simplificar nuestro código y hacerlo más seguro y legible.

Acompáñame en este recorrido donde implementaremos la clase, sus métodos principales y probaremos su funcionamiento. 🚀

◆ Paso 1: Crear la clase `ManejoArchivos`

👉 Vamos a crear el archivo `ManejoArchivos.py` dentro de nuestro proyecto. Este archivo contendrá la definición de la clase.

📁 **Ruta del archivo:** `ManejoArchivos.py`

📄 Descripción breve:

Aquí creamos la clase `ManejoArchivos` que implementa los métodos `__enter__` y `__exit__` necesarios para funcionar como un **Context Manager**. También incluimos un constructor `__init__` para recibir el nombre del archivo a manejar.

💻 Código:

```
class ManejoArchivos:
    def __init__(self, nombre):
        self.nombre = nombre

    def __enter__(self):
        print('Obtenemos el recurso'.center(50, '-'))
        self.nombre = open(self.nombre, 'r', encoding='utf8')
        return self.nombre


    def __exit__(self, tipo_excepcion, valor_excepcion, traza_error):
        print('Cerramos el recurso'.center(50, '-'))
        if self.nombre:
            self.nombre.close()
```

📄 Explicación:

- ✅ En el método `__init__` guardamos el nombre del archivo.
- ✅ En `__enter__` abrimos el archivo y mostramos un mensaje indicando que hemos obtenido el recurso.
- ✅ En `__exit__` cerramos el archivo y mostramos otro mensaje indicando que cerramos el recurso.

◆ Paso 2: Utilizar la clase `ManejoArchivos` con `with`

👉 Ahora creamos el archivo `archivos_con_with.py` donde utilizaremos la clase que definimos para abrir y leer un archivo usando la sintaxis `with`.

 **Ruta del archivo:** `archivos_con_with.py`

Descripción breve:

Este archivo importa la clase `ManejoArchivos` y la usa para abrir el archivo `prueba.txt` dentro de un bloque `with`, garantizando que el archivo se cierre automáticamente al finalizar el bloque.

Código:


```
from ManejoArchivos import ManejoArchivos

with ManejoArchivos('prueba.txt') as archivo:
    print(archivo.read())
```

Explicación:

- ✓ Usamos `with ManejoArchivos('prueba.txt') as archivo:` para abrir el archivo.
 - ✓ Al entrar al bloque `with`, se ejecuta automáticamente el método `__enter__`.
 - ✓ Dentro del bloque, leemos y mostramos el contenido del archivo.
 - ✓ Al salir del bloque, se ejecuta automáticamente el método `__exit__`, cerrando el archivo.
-

◆ Paso 3: Crear el archivo de prueba `prueba.txt`

 Este archivo contiene el texto que vamos a leer.

 **Ruta del archivo:** `prueba.txt`

Contenido del archivo:

```
Agregamos información al archivo
Adiós
Última línea
```

Explicación:

- ✓ Este archivo es necesario para probar nuestra clase. Contiene algunas líneas de texto que vamos a recuperar y mostrar usando nuestro **Context Manager**.
-

Cómo ejecutar el proyecto

✅ Ejecuta el archivo `archivos_con_with.py` con clic derecho > **Run** o con el comando:

👉 Al ejecutarlo, deberías ver en la consola algo similar a esto:

```
-----Obtenemos el recurso-----  
Agregamos información al archivo  
Adiós  
Última línea  
-----Cerramos el recurso-----
```

🎉 ¡El archivo se abrió y cerró correctamente de manera automática!

✅ Conclusión

En esta lección hemos aprendido a:

- 🎯 Implementar un **Context Manager** personalizado en Python mediante los métodos `__enter__` y `__exit__`.
- 🎯 Manejar correctamente la apertura y cierre de archivos sin necesidad de bloques `try-finally`.
- 🎯 Usar la sintaxis `with` para simplificar el manejo de recursos.

Este patrón no solo es útil para archivos, sino también para otros recursos como conexiones a bases de datos.



Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 🙌

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)