Tema 10. Manejo de Archivos en Python

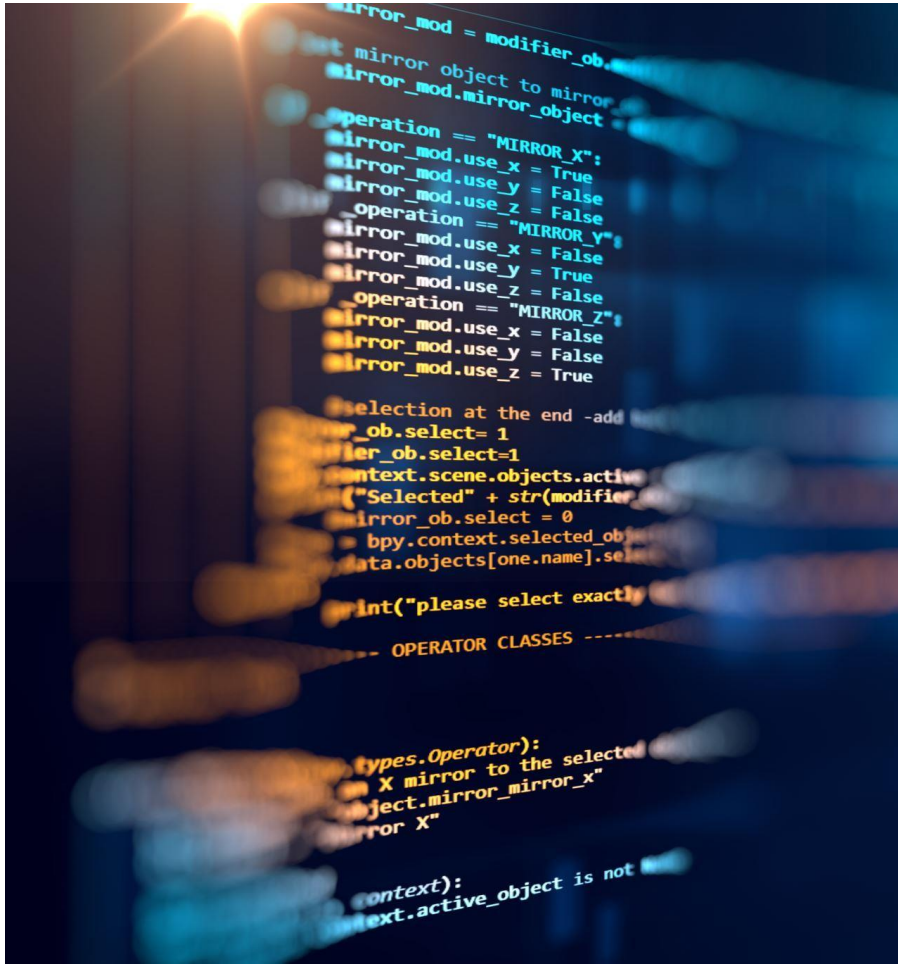
Visión general

Manejo básico de archivos

El manejo de archivos es una habilidad esencial para cualquier programador de Python. Aprenderemos a abrir, leer y escribir en archivos utilizando Python.

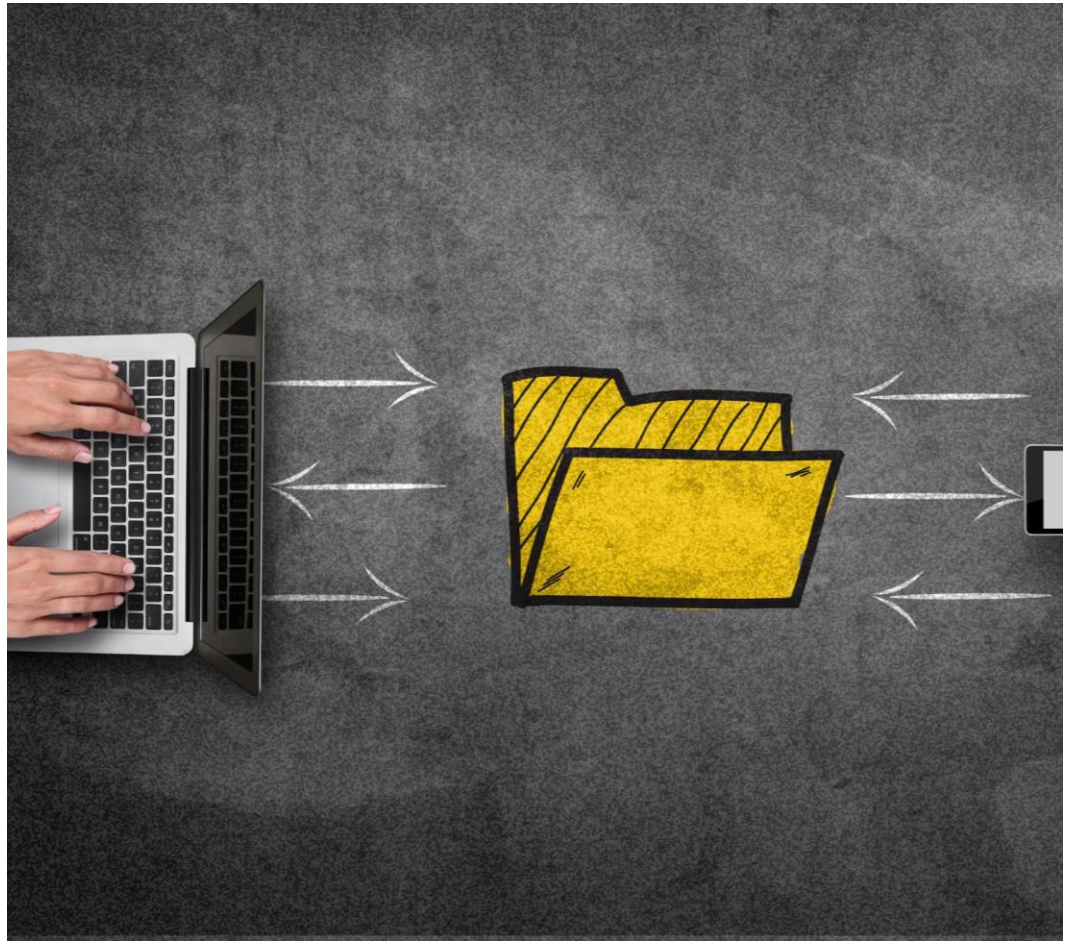
Archivos CSV

Los archivos CSV son una forma común de almacenar datos estructurados. Veremos cómo trabajar con archivos CSV en Python y cómo manipular los datos almacenados en ellos.



Conceptos básicos de los archivos de Python

Python proporciona una variedad de funciones y métodos integrados para trabajar con archivos, lo que hace que la manipulación de archivos sea una tarea muy fácil en Python. Exploraremos los conceptos básicos de cómo trabajar con archivos en Python.



Abrir un archivo

Para trabajar con un archivo en Python, es necesario abrirlo primero. Aprenderemos a abrir un archivo para lectura, escritura y apendizado.

Abrir Archivos

Para abrir un archivo es necesario la función **open()** que retorna un objeto archivo (POO).

fo = open (“nombredelarchivo”,”modo”)

Creamos un objeto fo (fo file object) que recibe el objeto de la función open

El modo, es el permiso que tendrá el programa sobre el archivo.

Modo de lectura

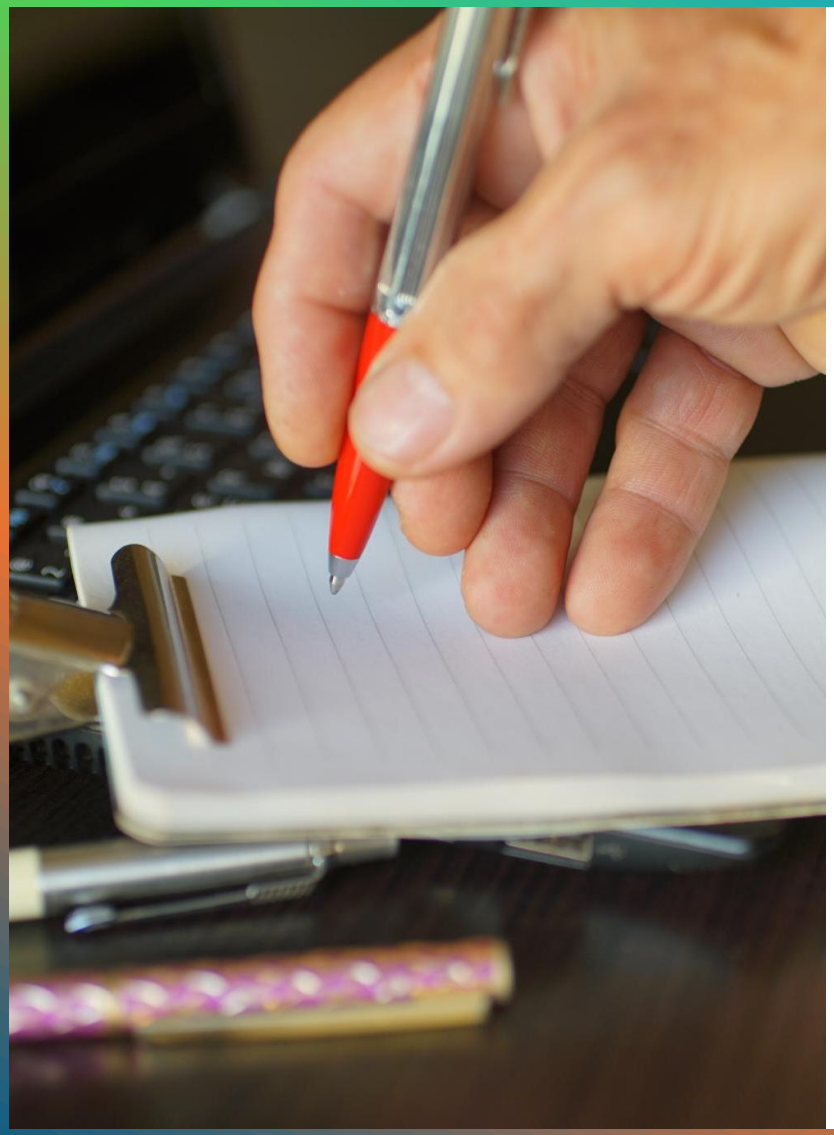
r	Read. Permiso de solo lectura.
w	Write. Permite leer y escribir nueva información sobre el archivo •Al guardar la información si existe otro archivo con el mismo nombre, será borrado y reemplazado.
a	Appending. Se usa para agregar datos al archivo. •Toda la nueva información se escribe al final del archivo
r+	Lectura y escritura especial. Invoca ambas acciones cuando se trabaja con un archivo.

Ejemplo

Observa que fo es un objeto, para manipular ese objeto necesitamos funciones.

```
fo = open ("Hola Mundo.txt","w")
## Si no existe el archivo, se creará porque el modo de abrir es
w.
print (fo)

>> <_io.TextIOWrapper name='Hola Mundo.txt' mode='w'
encoding='cp1252'>
```



Escritura en un archivo

Escribir en un archivo es una tarea común en la programación. Aprenderemos cómo escribir y agregar información adicional al final del archivo. Los archivos son útiles para almacenar información permanentemente y compartir información entre diferentes programas.

Función write()

- La función se aplica sobre el objeto.
- La función solo admite un **str**
- Debes incluir saltos de línea, espacios o tabs para tener el formato deseado.
- Observa que debemos de cerrar el archivo

```
fo = open ("Hola Mundo.txt","w")
print (fo)


fo.write ("Hola mundo!\n")
x = 10
fo.write (str(x))


fo.close()

>><_io.TextIOWrapper name='Hola Mundo.txt' mode='w'
encoding='cp1252'>
```

Instrucción close()

Cuando haya terminado de trabajar, puede usar el comando fo.close () para finalizar.

Lo que esto hace es cerrar el archivo por completo, terminando los recursos en uso y liberándolos.

Es importante comprender que cuando usa el método close (), cualquier intento adicional de usar el objeto de archivo fallará.

Esta es una buena práctica.

Instrucción with

Puedes trabajar con objetos de archivo utilizando la instrucción with.

Está diseñado para proporcionar una sintaxis mucho más **limpia** y manejar excepciones.

Es una buena práctica utilizar la instrucción **with** cuando corresponda.

Una ventaja de usar este método es que **cualquier archivo abierto se cerrará automáticamente una vez que haya terminado**. Esto deja menos de qué preocuparse durante la limpieza.

Ejemplo

- Al usar la instrucción with no es necesario la instrucción close()
- Al no usar la instrucción with, es necesario usar la instrucción close() para liberar recursos.

```
with open ("Hola Mundo.txt","r") as file:
    for line in file:
        print (line)

fo = open ("Hola Mundo.txt","r")
for line in fo:
    print (line)
fo.close()
```

Abrir desde una dirección

fo = open (**path**, “modo”)

Path es la dirección donde se encuentra el archivo

“C:/…/Holamundo.txt”

Lectura de un archivo

Abrir un archivo para lectura

Para leer el contenido de un archivo, primero debemos abrirlo para lectura. Los archivos se pueden abrir en modo de texto o binario según sea necesario.

Leer el contenido de un archivo línea por línea

La lectura de archivos línea por línea es útil cuando se trabaja con archivos de texto. Podemos usar un bucle para leer el archivo línea por línea y realizar operaciones con la información de cada línea.

Leer el contenido de un archivo en su totalidad

La lectura de archivos en su totalidad es útil cuando trabajamos con archivos binarios o archivos de texto con un tamaño pequeño. Podemos leer el archivo completo en una sola operación y manipular el contenido según sea necesario.

Ejemplo de Lectura de Archivo

```
with open ("Hola Mundo.txt","r") as
file:
    for line in file:
        print (line)
```

Ejemplo de Lectura de Archivo sin with

- Abrir Archivo de Texto

  Usar 'open' con el nombre del archivo y modo 'r' para leer

- Leer Líneas del Archivo

  Iterar sobre el archivo con un bucle 'for'

  Imprimir cada línea

- Cerrar el Archivo

  Usar 'close' para cerrar el archivo y liberar recursos

Trabajar con archivos CSV

¿Qué son los archivos CSV?

Los archivos CSV son un formato de archivo común utilizado para almacenar datos en forma de tabla. Cada línea del archivo representa una fila de la tabla y las columnas están separadas por comas.

Lectura de archivos CSV en Python

Python tiene una biblioteca incorporada para leer archivos CSV. Podemos leer archivos CSV línea por línea o leer todo el archivo a la vez y luego manipular los datos en una lista o diccionario.

Escritura de archivos CSV en Python

Python también tiene una biblioteca incorporada para escribir datos en archivos CSV. Podemos escribir datos en un archivo CSV línea por línea o escribir todos los datos en una lista o diccionario de una sola vez.