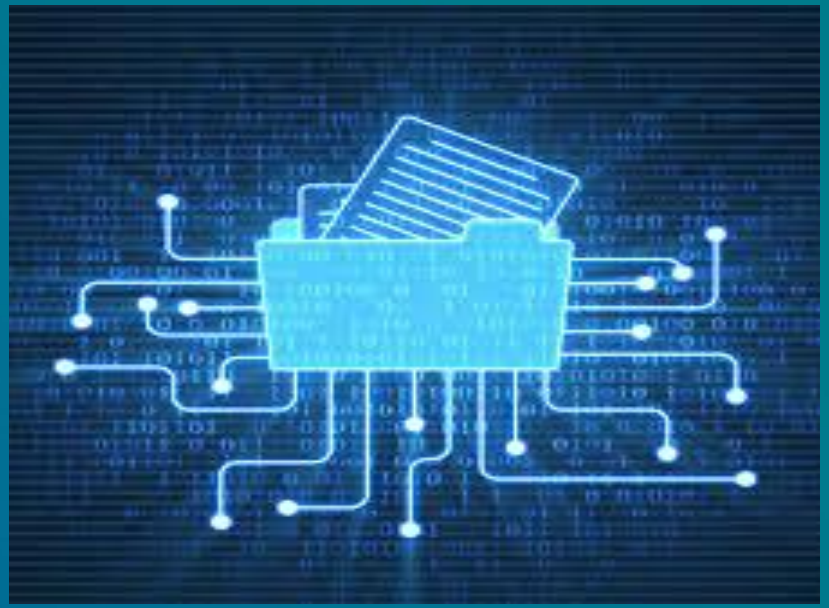
Manejo de archivos



Archivos

Un archivo es una **colección de datos almacenados** en un sistema de almacenamiento (DH, SSD, USB, etc). Permiten a los usuario y a los programas **guardar** y **recuperar** información de manera organizada y persistente. Los archivos pueden contener diversos tipos de datos, como texto, imágenes, audio, video, código fuente, etc.

Los **archivos de texto** contienen caracteres legibles, es posible no solo leer dicho contenido sino también modificarlo usando un editor de texto.

Un **archivo binario** es cualquier archivo que no se pueda interpretar en forma de texto: una imagen, un sonido o incluso un archivo comprimido.

Archivos

1. **Apertura:** Acceder a un archivo para leer, escribir o modificar su contenido.
2. **Lectura**: Extrae datos de un archivo.
3. **Escritura**: Guardar datos en un archivo, ya sea creando contenido nuevo o modificando el existente.
4. **Cierre**: terminar el acceso a un archivo y liberar cualquier recurso asociado.

Modo de apertura

utilizar el modo correspondiente para cada caso

MODO	DESCRIPCIÓN
r	Abre un archivo de texto sólo para lectura
rb	Abre un archivo binario sólo para lectura
r+	Abre un archivo de texto para escritura y lectura
w	Abre un archivo de texto sólo para escritura (si existe lo sobreescribe)
wb	Abre un archivo sólo para escritura binaria (si existe lo sobreescribe)
w+	Abre un archivo de texto para escritura y lectura (si existe lo sobreescribe)
a	Abre un archivo para anexar información

r -> abre un archivo sólo para lectura. El puntero al archivo está localizado al comienzo del archivo. Este es el modo predeterminado de la función.

r+ -> abre un archivo para escritura y lectura. El puntero del archivo está localizado al comienzo del archivo.

w -> abre un archivo solo para escritura. Sobreescribe el archivo si este ya existe. Si el archivo no existe lo crea.

w+ -> abre un archivo para escritura y lectura. Sobreescribe el archivo si este ya existe. Si el archivo no existe lo crea.

a -> abre un archivo para anexo. El puntero del archivo está al final del archivo si este existe. Es decir, el archivo está en modo anexo. Si el archivo no existe, crea un nuevo archivo para escritura.

Para abrir un archivo con Python, podemos usar la función **open**.

```
archivo = open(nombre_archivo, modo)
```

`archivo` objeto file, el cual será utilizado para llamar a otros métodos asociados con el archivo.

`nombre_archivo` string que contiene el nombre del archivo al que queremos acceder.

`modo` string que contiene el modo de apertura del archivo.

Modo de cierre

Para cerrar un archivo con Python, podemos usar la función **close**.

```
archivo.close()
```

`archivo` objeto file que fue obtenido con el método open.

Podemos usar la sentencia **with** para abrir archivos en Python y dejar que el intérprete se encargue de cerrar el mismo.

```
with open('archivo.txt','r+') as archivo:
    for linea in archivo:
        print(linea, end="")
```

Modo de escritura

Write : Escribe una cadena de caracteres dentro del archivo.

```
archivo = open('archivo.txt', 'w')
archivo.write('Primer linea de texto\n')
archivo.write('segunda linea\n')
archivo.write('tercera linea\n')
archivo.close()
```

Writelines : Escribe una lista de cadena de caracteres dentro del archivo. El parámetro que recibe el método podrá ser cualquier iterable.

```
archivo = open('archivo.txt', 'w')
lineas_texto = ['Primer linea de texto\n',
                'segunda linea\n',
                'tercera linea\n']
archivo.writelines(lineas_texto)
archivo.close()
```

Modo de lectura

read: permite extraer un string que contenga todos los caracteres del archivo(Es posible limitar al método read a que lea cierta cantidad de bytes **read**(cantidad)).

```
# Abrimos el archivo en modo lectura y escritura
archivo = open('archivo.txt', 'r+')
texto = archivo.read()
print('El contenido del archivo es: ' + texto)
# Cerramos el archivo
archivo.close()
```

readlines: Permite obtener una lista con todas las líneas que contiene el archivo.

```
archivo = open('archivo.txt', 'r+')
lista_lineas = archivo.readlines()
for linea in lista_lineas:
    print(linea, end="")
# Cerramos el archivo
archivo.close()
```

Archivos CSV

CSV: (Comma Separated Values) es un formato de archivo de texto que almacena datos en tablas, donde cada línea representa una fila y los valores dentro de cada fila están separados por comas.

```
Nombre,Edad,Ciudad
Juan,28,Madrid
Ana,22,Barcelona
Luis,35,Sevilla
```

Lectura:

Podemos leer un csv recorriendo sus líneas y obteniendo uno por uno los valores que se encuentran entre las comas.

leer csv.py > ...

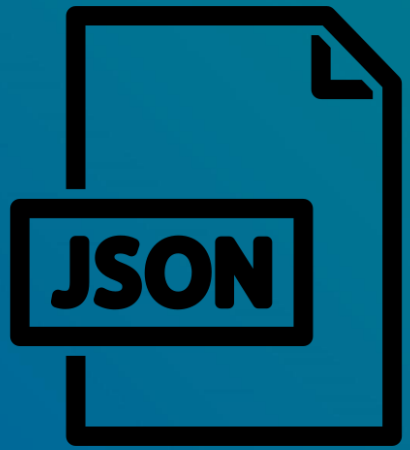
```python
with open("archivo.csv", "r") as archivo:
    matriz = []
    nombres_columnas = archivo.readline().strip().split(",")

    for linea in archivo:

        linea = linea.rstrip("\n")
        fila = []
        valores = linea.split(",")

        for valor in valores:

            if valor.isdigit():
                fila.append(int(valor))
            else:
                fila.append(valor)

        matriz.append(fila)

print(nombres_columnas)

for fila in matriz:
    print(fila)
```

Escritura:

Para guardar un csv debemos realizar el procedimiento inverso. Debemos transformar cada fila de la matriz en una línea de texto en la que los valores se separan por comas. Y cuando la fila finaliza, realizar un salto de línea.

```python
nombres_columnas = ["Nombre", "Edad", "Ciudad"]
matriz = [["Pedro", 24, "París"], ["José", 25, "toronto"]]

with open("archivo.csv", "w") as archivo:
    archivo.write(",".join(nombres_columnas) + "\n")

    for fila in matriz:
        linea = ""
        for i in range(len(fila)):

            linea += str(fila[i])

            if i < (len(fila) - 1):
                linea += ","

        archivo.write(linea + "\n")
```

Archivos JSON



Json: (JavaScript Object Notation) es un formato de texto utilizado para almacenar e intercambiar datos estructurados. Organiza los datos en pares de clave-valor, similar a un diccionario en Python.

```
{
    "nombre": "Juan",
    "edad": 28,
    "ciudad": "Madrid",
    "amigos": [
      {"nombre": "Ana", "edad": 22},
      {"nombre": "Luis", "edad": 35}
    ]
}
```

Lectura: El módulo json en Python permite trabajar con datos en formato JSON. La función json.load() lee un archivo JSON y lo convierte en un objeto de Python (como un diccionario o una lista).

```python
import json
# Leer archivo JSON
with open('archivo.json', 'r') as archivo_json:
    datos = json.load(archivo_json)  # Cargar el contenido en un diccionario
    print(datos)
```

Escritura: La función json.dump() convierte un objeto de Python (como un diccionario) a formato JSON y lo escribe en un archivo.

```python
import json
# Escribir en un archivo JSON
datos = {
    "nombre": "Juan",
    "edad": 28,
    "ciudad": "Madrid"
}
with open('archivo.json', 'w') as archivo_json:
    json.dump(datos, archivo_json, indent=4)  # El parámetro indent para que sea más legible
```