Archivos

Archivos

Si al correr un programa generamos datos, ya sea ingresados por teclado o generados como resultado de transformaciones, y deseamos acceder a ellos una vez terminado el programa, o desde otro programa, necesitaremos guardarlos en una estructura capaz de sobrevivir al fin de la ejecución del proceso activo. De esta manera, se pueden rescatar posteriormente y seguir trabajando con ellos.

La manera de conservarlos para más tarde es grabarlos como **archivo**, con un nombre, una extensión, y una organización que nos sea conveniente.

Cada tipo de archivo tiene sus particularidades y requiere formas específicas en su manejo, y puede requerir algoritmos especiales para su uso y manipulación.

Según la Arquitectura Von Neumann, para que los datos se puedan procesar, deben estar alojados en la Memoria Interna (MI). Y para que se puedan copiar datos hacia un dispositivo externo (disco, red, etc.), también debe pasar por la MI. Es decir, para leer un archivo necesitaremos copiarlo a la MI, y para guardar las modificaciones, se debe copiar la versión que existe en la MI al archivo.

En ocasiones los archivos contienen grandes volúmenes de datos, por ello no es eficiente copiar el archivo completo en la MI, así que se copia por partes. A medida que necesitamos información la copiamos en la MI, y a medida que se modifica, se va guardando en el archivo.

El sistema operativo reserva zonas de memoria para la comunicación entre el programa y los archivos. Genéricamente, estos espacios de intercambio se denominan **buffers**. El SO implementa un sistema de lectura y escritura en bloques, es decir, si el programa necesita un dato, se trae un bloque de información dentro del cual se incluye ese dato, esto debido a que sería ineficiente traer un solo dato, y porque es probable que también se vayan a necesitar datos cercanos.

Archivos en Python

Para abrir un archivo en Python disponemos de la función `open()`. Esta función le solicita al SO establecer el vínculo de trabajo y nos devuelve la dirección de inicio del buffer del archivo. Necesitaremos guardarla en una variable, ya que será la manera con la que nos vamos a referir al archivo desde ese momento.

Necesitaremos indicarle a la función open, la **ruta** (*path*) al archivo, que puede ser absoluta o relativa, el nombre del archivo, y la extensión.

Por ejemplo, para abrir un archivo `texto.txt` que está en el mismo directorio (carpeta) que nuestro programa:

```
archivo = open("texto.txt")
```

Una vez que terminamos de usar el archivo, es importante **cerrarlo**, ya que esta operación libera el buffer y completa el grabado del mismo. Al igual que la lectura, el grabado de archivos también se realiza en bloques, y cerrar un archivo obliga al SO a grabar cualquier dato que haya quedado pendiente. Para cerrar un archivo, se utiliza el método `close()`.

Modos de apertura

Según lo que vayamos a hacer con el archivo, existen diferentes modos de apertura, que se pasan como segundo parámetro a la función `open()`. El modo por defecto es `r` (leer).

Modo	Lee	Escribe	Existencia de Archivo	Posición Inicial de Lectura	Posición Inicial de Escritura
r	✅	-	Si el archivo no existe, da error.	Inicio	-
w	-	✅	Si el archivo no existe, lo crea. Si existe, le borra todo el contenido.	-	Inicio
a	-	✅	Si el archivo no existe, lo crea.	-	Final
r+	✅	✅	Si el archivo no existe, da error.	Inicio	Depende
w+	✅	✅	Si el archivo no existe, lo crea. Si existe, le borra todo el contenido.	Inicio	Depende

Métodos de Lectura

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

- **read**: Lee todo el archivo de una vez y lo devuelve como un string.

- **readlines**: Lee todo el archivo y devuelve una lista de líneas. Cada línea contiene el carácter `\n` que indica el salto de línea.
- **readline**: Lee línea por línea, cada vez que se llama a la función.

Métodos de Escritura

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

- **write**: Escribe un string en el archivo.
- **writelines**: Escribe una lista de strings en el archivo.

Nota: Los métodos write y writelines no añaden saltos de línea al final. Si se desea cambiar de línea, debemos agregar manualmente un `\n`.

Modo append (agregar)

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

Cuando se abre un archivo en modo append, este se abre para escritura, pero con el puntero de posición al final del contenido del mismo. Esto significa que al utilizar *write* o *writelines*, estaremos añadiendo contenido al final del archivo, sin sobreescribir lo existente.

Modos `r+` y `w+`

El modo `r+` se comporta de la misma manera que `r` al leer contenido del archivo, y al escribir, sobrescribe lo que ya existe.

- Es importante notar que cuando leemos el archivo, el puntero de posición se sitúa al final de lo leído. Por lo tanto, si leemos y luego escribimos en el mismo archivo, el texto se añadirá al final (sin sobreescribir lo que ya teníamos).

El modo `w+` se comporta igual que `w` (si el archivo no existe, lo crea, y si ya existe, borra **todo** su contenido). Es decir, si leemos un archivo abierto como `w+`, no nos devolverá nada a menos que escribamos algo antes.

En todos los modos, si es necesario, se puede mover manualmente el puntero usando `.seek(n)`, donde `n` es la posición donde queremos colocarlo (empezando desde 0 para el inicio del archivo).

Tipos de archivo

Texto plano (.txt)

Un archivo de texto plano o texto simple, es un archivo que almacena texto sin utilizar ningún formato o estructura definido.

Comma-separated values (.csv)

Los archivos CSV se utilizan generalmente para guardar datos con cierta estructura y relación entre sí. Por ejemplo, si tenemos estos datos (representados en una tabla o en una hoja de cálculo):

Nombre	Apellido	DNI	Provincia	Teléfono
Juan	Pérez	12345678	Buenos Aires	1122334455
María	Gómez	23456789	Córdoba	2233445566
Pedro	Martínez	34567890	Santa Fe	3344556677
Ana	Lopez	45678901	Mendoza	4455667788
Lucía	Fernández	56789012	Entre Ríos	5566778899

Estos datos tendrían la siguiente estructura en un archivo CSV:

```
nombre,apellido,dni,provincia,teléfono
Juan,Pérez,12345678,Buenos Aires,1122334455
María,Gómez,23456789,Córdoba,2233445566
Pedro,Martínez,34567890,Santa Fe,3344556677
Ana,Lopez,45678901,Mendoza,4455667788
Lucía,Fernández,56789012,Entre Ríos,5566778899
```

En cada línea del archivo, los valores (columnas) se separan mediante un carácter, generalmente la coma (`,`), aunque algunos programas usan el punto y coma (`;`).