USO DE FICHEROS EN PYTHON

Concepto de fichero

Un fichero es un conjunto de datos relacionados entre sí, que son almacenados de forma **permanente** en dispositivos tales como discos duros, memorias *flash*, etc. y que pueden ser tratados, hasta cierto punto, como una unidad. Esto marca la diferencia con los datos que son provisionalmente almacenados en la memoria RAM o la **memoria volátil** del ordenador, los cuales no *sobreviven* al programa que los crea y mucho menos al desconectar el computador de la red eléctrica.

Los ficheros son muy importantes en programación. No solo debido al hecho evidente de que brindan el soporte que permite la mera existencia de las aplicaciones en el disco duro. Si no existieran los ficheros, por ejemplo, toda la interacción del usuario con una aplicación concreta, como la introducción y la obtención de resultados parciales, tendría que ser recreada cada vez que se ejecutara el programa. La existencia del fichero hace posible que, al terminar una sesión de trabajo, se almacenen los datos y resultados en el disco, para poder retomar la tarea en una sesión posterior.

Por otra parte, los ficheros son útiles también como vehículos para el intercambio de información entre diferentes aplicaciones: los datos resultantes *escritos* en un fichero por el programa **A** podrán servir de *entrada* para un programa **B**

En la mayor parte de las aplicaciones reales se debe leer y escribir datos en ficheros. Al utilizar ficheros para guardar los datos estos perdurarán tras la ejecución del programa, pudiendo ser consultados o utilizados más tarde.

Los ficheros en Python son objetos de tipo **file** creados mediante la función **open** (abrir).

Las operaciones más habituales con ficheros son:

- Abrir un fichero. → **fichero = open(‘nombreArchivo.txt’, ‘w’)**
- Escribir datos en un fichero. → **fichero.write(“contenido del archivo”)**
- Leer datos de un fichero. → **fichero.read()**
- Lectura y escritura
- Agregar → a (append)
- Borrar un fichero.

En Python para nosotros poder leer y crear archivos, debemos hacer uso del la f**unción *open***. La función recibe cómo argumento la ruta del archivo a utilizar. Opcionalmente, como segundo argumento podemos colocar el modo con el cual trabajaremos el archivo. Por default será lectura ('**r'**). Se le puede agregar **encoding = ‘utf-8** para que acepte los caracteres especiales, es opcional.

Abrir un fichero

Para crear un fichero nuevo se utiliza la instrucción ***open('nombre_archivo.txt', 'w')***: Crea el fichero con el nombre '*nombre_archivo*' y la ruta especifica, lo abre en modo escritura (el argumento '**w**' significa write) y devuelve un objeto que lo referencia. Si el fichero indicado por la ruta ya existe en el sistema, se reemplazará por el nuevo.

file = open('Bienvenida.txt', 'w')

```
file = open('Bienvenida.txt', 'w')
```

Simplemente se crea el archivo indicado.

Si queremos que cree el archivo con algún contenido podemos usar la siguiente estructura.

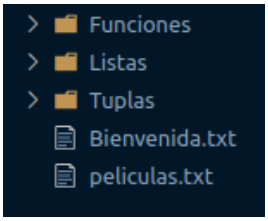
```
file = open('Bienvenida.txt', 'w')
file.write("Hoal, bienvenido al tema de ficheros en python")

print(file.read())

file.close()
```

Se creará el archivo con el mensaje determinado. El **.close()** se usa para que el archivo no quede abierto, siempre es recomendable cerrar los archivos después de abiertos.

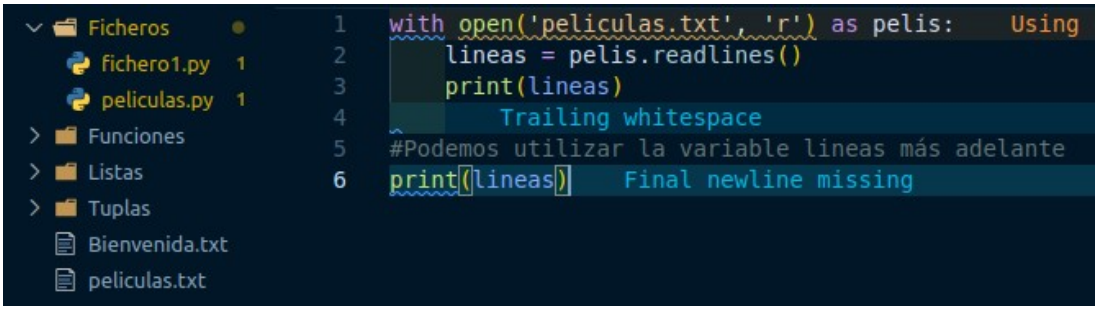
Empleando la palabra clave with

La palabra reservada **with** nos permite también crear o abrir un fichero y tiene la particularidad que lo cierra automáticamente sin necesidad de usar el .close()

Crearemos el archivo **peliculas.txt**



Ahora lo abriremos desde un archivo **python** haciendo uso de **with**



La palabra **with** abre el fichero utilizando un **alias** que en este caso estamos usando **pelis.**

La variable **lineas** se puede seguir usando más adelante en el código, las veces que sea necesario. Al imprimir veremos que nos imprime dos veces el listado de películas.

```
[Running] python -u "/home/juang/SENA_2025/TRIMESTRE_3/GRUPO_3002085/Ejercicios/Ficheros/peliculas.py"
['Corazón Valiente\n', 'La Vida es Bella\n', 'En Busca de la Felicidad\n', 'El Señor de los Anillos\n']
['Corazón Valiente\n', 'La Vida es Bella\n', 'En Busca de la Felicidad\n', 'El Señor de los Anillos\n']
```

Se puede ver que al final de cada elemento se coloca un **\n** lo que significa que al final se imprime un salto de línea. Para evitar que nos imprima el \n al final podemos usar la siguiente expresión.

```
with open('frutas.txt', 'r') as archivo_frutas:
    contenido = archivo_frutas.read()
    line = contenido.split('\n')
    print(line)    Final newline missing
```

Usando el **método split** se puede suprimir ese salto de línea al final

```
[Running] python -u "/home/juang/SENA_2025/TRIMESTRE_3/GRUPO_3002085/Ejercicios/Ficheros/frutas.py"
['Manzana', 'Naranja', 'Mandarina', 'Pera', 'Uvas']
```

Recorriendo el fichero con un for

```
for elemento in line:
    print(elemento)
```

```
Manzana
Naranja
Mandarina
Pera
Uvas
```

En siguiente ejemplo podemos ver que al imprimir los elementos del archivo también se imprime un salto de línea

```
with open('lenguajes.txt', 'r') as tiposLenguajes:
        for linea in tiposLenguajes:
                print(linea)
```

```
Python

Javascript

PHP

Java

C++

C#
```

Cuando se imprime el fichero automáticamente se imprime un salto de línea al final de cada línea, para quitar o evitar que se imprima el salto de línea podemos emplear la siguiente expresión, reemplazando el salto de línea por un espacio en blanco.

```
for l in lineas:
        print(l.replace('\n', ''))
```

```
Python
Javascript
PHP
Java
C++
C#
```

Escribir datos en un fichero

Una vez creado el fichero, para escribir datos en él se utiliza el método ***fichero.write(c)*** : Escribe la cadena c en el fichero referenciado por fichero.

```
fichero2.write('¡Bienvenido a Python!')
```

```
fichero2 = open('fichero2.txt', 'w')
fichero2.write("Bienvenido al manejo de Ficheros en Python")
fichero2.close()
```

Escribir varias cadenas de texto al tiempo

Para escribir varias cadenas al tiempo en un archivo se debe emplear el método **writelines().** Se crea una lista de elementos de cadena y luego cada cadena se agrega al archivo de texto.

```
pais = open('paises.txt', 'w')
pais.write('Suecia, ')
pais.writelines(['Inglaterra, ', 'Francia, ', 'Colombia \n', 'Brasil'])
pais.close()
```

Leer Archivo

Para leer el contenido de un archivo podemos hacerlo con los métodos del tipo objeto **file** como **read(), readline() y readlines()**

`read()` lee todo el contenido como una única cadena (o un número de caracteres especificado), **`readline()`** lee y devuelve una sola línea a la vez, y **`readlines()`** lee todas las líneas de una vez y las devuelve como una lista de cadenas, donde cada elemento es una línea del archivo.

```
file = open("nombres.txt", "r")
nombres_apellidos = file.readlines()
file.close()
```

Añadir datos a un fichero

Si en lugar de crear un fichero nuevo queremos añadir datos a un fichero existente se debe utilizar la instrucción ***open('ruta del fichero', 'a')***: Abre el fichero con la ruta ***del fichero especificada*** en modo añadir (el argumento 'a' significa ***append***) y devuelve un objeto que lo referencia.

Una vez abierto el fichero, se utiliza el método de escritura anterior y los datos se añaden al final del fichero.

```
with open('agregarDatos.txt', 'a') as file:
    file.write("Primer línea de texto.\n")
    file.write("Segunda línea de texto.\n ")
    file.write("Tercera línea de texto.\n")
```

Añadir datos de una lista

Para añadir los elementos de una lista a un archivo se debe iterar sobre la lista y añadir cada elemento.

```
#Añadir elementos de una lista
paises = ['Colombia', 'España', 'Suiza', 'Escocia']

with open('agregarDatos.txt', 'a') as file:    Using
    for pais in paises:
        file.write(pais + "\n")
    file.write("\n")    Final newline missing
```