PRO UT3-A4: Operaciones con ficheros

1 - Ficheros:

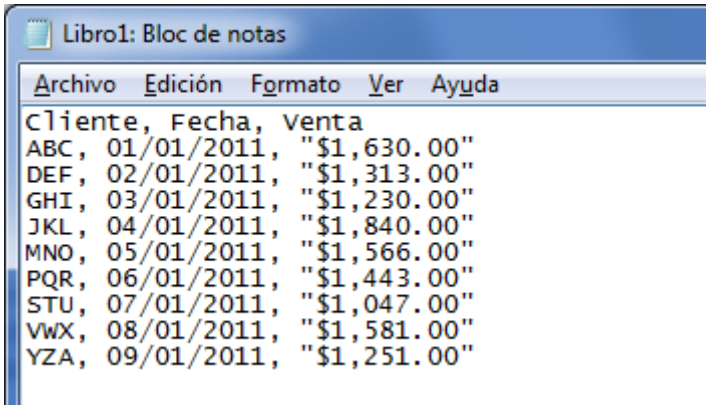
Llamamos fichero al conjunto de información relacionada entre sí. Esta información es tratada como un todo y se organiza de manera estructurada.

Con el paso del tiempo, la evolución de los clásicos ficheros pasaron del papel a una versión digital. En cuanto a la programación se refiere, podemos decir que estos ficheros se encuentran formados por registros lógicos donde están contenidos datos relativos al mismo elemento u objeto. Son altamente importantes, pues hace posible la mera exitencia de las aplicaciones en el disco duro. Además, es posible continuar con las tareas que se estaban realizando en sesiones anteriores.

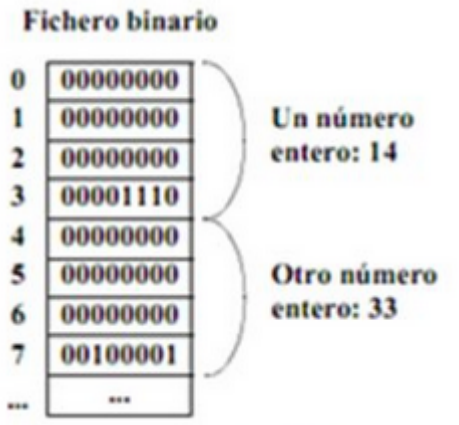
Al almacenar un fichero estamos almacenando de manera permanente. Por lo que hasta que sean borrados, estos datos permanecerán en el medio en el que han sido almacenados y continuarán existiendo, incluso, después de que el programa que los ha creado, deje de ejecutarse o el ordenador haya sido apagado. Por tanto, no son vólatiles, ya que no son almacenados en la memoria RAM.

Entre los tipos de ficheros, encontramos los siguientes dos:

- **Ficheros de texto:** Los datos que recogen están representados con caracteres alfanuméricos. Pueden ser leídos y modificados mediante un editor de texto.



- **Ficheros binarios:** Contienen una representación binaria de ceros y unos, y no se pueden editar.



2 - Apertura de ficheros:

En python los ficheros los abrimos con la función *open*. El primer parámetro hace alusión al nombre del fichero, y el segundo parámetro el modo con el que abrimos el mismo. Mostramos los siguientes ejemplos:

- Abrir fichero en modo lectura:

```
f = open("mi_archivo.txt", "r")
```

- Abrir fichero en modo escritura:

```
f = open("mi_archivo.txt", "w")
```

- Abrir fichero en modo lectura / escritura:

```
f = open("mi_archivo.txt", "r+")
```

- Abrir fichero binario en modo lectura:

```
f = open("mi_archivo.txt", "rb")
```

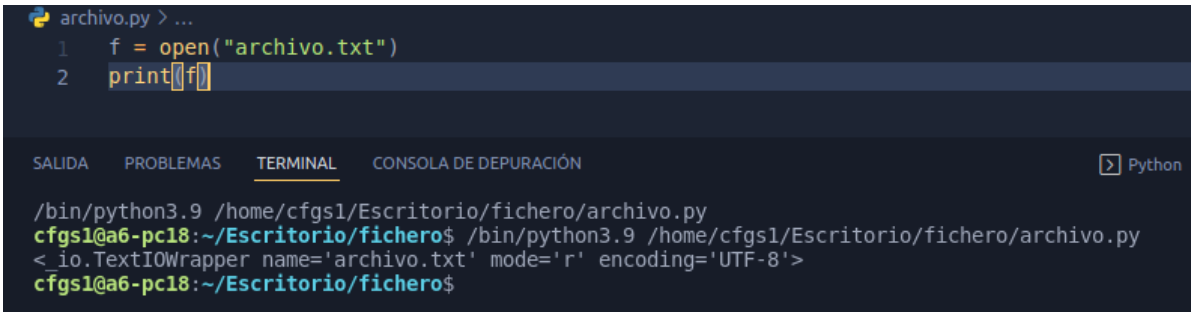
- Abrir fichero binario en modo escritura:

```
f = open("mi_archivo.txt", "wb")
```

- Abrir fichero para añadir al final:

```
f = open("mi_archivo.txt", "a")
```

El parámetro encoding se usa para la especificación de la utilización de un archivo, permitiendo cualquier codificación que codifique y decodifique desde bytes y los tipos de datos admitidos por los métodos de archivos que depeden del códec utilizado. Al usar el método encode() se devuelve una versión codificada de acuerdo a la codificación especificada.



En la imagen anterior se puede ver el resultado del siguiente código al ser ejecutado:

```
f = open("archivo.txt")
print(f)
```

Tal y como se muestra podemos apreciar la dirección en memoria, el nombre del archivo, los permisos y el tipo de codificación que tiene el archivo.

Cierre de ficheros:

Son diversos los motivos por los que es necesario el cierre de un archivo. Podría darse alguna de las siguientes situaciones: determinados sistemas operativos donde los archivos sólo puedan ser abiertos por un programa a la vez; perdida de la información por no guardarla al cerrar el archivo; el límite de la cantidad de archivos que podemos abrir es bajo...

¿Cómo cerramos, entonces, un archivo en python? De la siguiente manera:

```
fichero.close()
```

Adicionalmente, se pueden abrir o cerrar ficheros de otras dos maneras.

- **Try ... finally:**

```
f = open("archivo.txt") # Al no tener especificados los permisos de escritura, se abre sólo como un archivo de lectura

try:
    f.write() # Intentará escribir en el fichero, pero no tiene los permisos

except:
    print("No tienes permisos de escritura") # Así que se lanza el mensaje de error en el "except"

finally: # "Finally" siempre se ejecuta.
    f.close() # Finalmente, se cerrará el archivo.
```

- **With ... as ...:**

```
with open("archivo.txt") as f: # Usamos with para abrir el archivo y después del "as" nombramos cómo queremos llamarlo

   # Ahora se ejecutaría el bloque de código y después se cerraría sólo.

   print("With... as... ")
```

Escritura en fichero. El método write():

Tal y como sugiere el título, se utiliza para escribir el contenido en un fichero:

```
f = open("archivo.txt", "w") # crea y abre un fichero nuevo

f.write("tres tristes tigres comen trigo en un trigal") # se escribe el contenido, en este caso, un texto

f.close() # cerramos el fichero
```

La función write no añade saltos de línea automáticos, pues al escribir se queda en la última posición de donde se escribió.

```
f = open("archivo.txt", "w") # crea y abre un fichero nuevo

alumno = input("Introduce un nombre: ")
edad = int(input(f"Edad de {alumno}: "))

f.write(alumno, edad)# Añadimos el valor de las variabes anteriores al documento
".txt"
```

Los modos de apertura que se pueden utilizar para escribir en el fichero son:

- **write:** Para escribir en el documento.
- **append:** Añadimos el contenido al final del documento.

Lectura de ficheros. El método read():

Su función es obtener una única cadena de caracteres unión de todas las líneas, incluyendo caracteres de nueva línea. Al usarse abre todo el contenido a la vez, si el archivo es grande producirá problemas de memoria, por ende para asegurar que el tamaño es adecuado y se lea cada vez hasta el final de llamada final, se debe pasar un parámetro de tamaño en el método read() y luego repetir llamando read repetidamente.

Podemos mostrar en pantalla el número de caracteres hasta la longitud que se le indica mediante el número que le pasamos. Por ejemplo:

```
Hola, mundo!
```

```
archivo = open("hola.txt", "r")
contenido = archivo.read(2)
print(contenido)
```

Resultado:

```
la
```

```
with open("archivo.txt", "r") as f:
    print(f.read())
    print(f.read())
```

Método tell():

El método tell() devuelve la posición actual del puntero (dirección en memoria) de un archivo.

Si el contenido de archivo.txt es:

```
Este es un fichero de texto de prueba
```

De este programa:

```
with open("archivo.txt", "r") as f:
    print(f.read(10))
    print(f.tell())
    print(f.read(5))
    print(f.tell())
    print(f.read())
    print(f.tell())
```

Resultado:

```
este es un
10
fich
15
ero de texto de prueba
37
```

El **primer** print() lo que hace es mostrar el texto del archivo hasta la longitud que se le indicó, con los espacios incluidos.
El **segundo** print() muestra el número que se le pasó en el anterior.
El tercero muestra lo mismo que el el primer solo que lo hace desde donde se quedó ese primer print() y lo continúa hasta la longitud que se le ha indicado (en este caso 5).
En el **cuarto** lo que hace es mostrar la suma de los read() anteriores, siendo 15 en total.
En el **quinto** como no se le pone el número de caracteres a mostrar lo que hace es mostrar todos los caracteres que faltaban a partir de donde se quedó el read() anterior.
En este **último** print() lo que hace es sumar el total del anterior tell() y el resto del read() último, danto un total de 37.

Método seek():

Este método es usado para mover el puntero de lectura del archivo a la posición especificada.

```
Este es un fichero de texto de prueba
```

```
with open("archivo.txt") as f: # Abrimos el archivo ".txt"
    print(f.read(4)) # Indicamos que queremos leer los 4 primeros caracteres
    print(f.tell()) # Mostramos la posición y nos devuelve 4
    print(f.seek(0)) # Devolvemos el puntero a la posición 0
    print(f.tell()) # Mostramos que el puntero sí está en la posición 0
```

Uso de bucle for para leer fichero:

Se utiliza para iterear con el contenido de un fichero. No podemos hacerlo directamente, pues nos mostraría el contenido directamente. Para poder hacerlo, debemos utilizar el método read(). De esta manera, sí estaríamos iterando con el contenido.

Ejemplo:

```
Tomates, huevos, aguacates, plátanos
```

```
with open("archivo.txt") as archivo:
    lista = archivo.read().split(", ") # Llamamos al archivo en modo lectura y le aplicamos un split para almacenarlo en una lista

    print(lista) # Imprimimos la lista, para combrobar que está dentro

    for producto in lista:
        print(producto) # Mostramos por consola cada item dentro de la lista
```

Resultado:

```
Tomates
huevos
aguacates
platanos
```

Métodos `readline()` y `readlines()`:

Método `readline()`:

Podemos leer un determinado número de líneas sin tener que leer todo el fichero, para ello, utilizamos el método readline(). Aquí un ejemplo:

Tenemos el fichero con el siguiente contenido:

```
Las plantas del instituto las tratan bien, sólo falta plantar unos girasoles. Las gafas se me empañan mucho.
Necesito descansar unas cinco horas más.
El aguacate es de lo mejorcito.
```

```
with open("otro_texto.txt", "r") as archivo_texto_2:
    print(archivo_texto_2.readline())
    print(archivo_texto_2.readline()) #si se le llama de nuevo, leerá otra línea porque ahora apunta a la siguiente
```

Resultado:

```
Las plantas del instituto las tratan bien, sólo falta plantar unos girasoles. Las gafas se me empañan mucho.
Necesito descansar unas cinco horas más.
```

Sintaxis: `archivo.readline(nºcaracteres)`
También podemos leer una cierta cantidad de caracteres, por ello en la sintaxis vemos que podemos añadir un argumento, el nº de caracteres que se deseen leer, ejemplo:

```
with open("otro_texto.txt", "r") as archivo_texto_2:
    print(archivo_texto_2.readline(10)) # Se leen los 10 priemros caracteres
```

Resultado: `Las planta`

Método `readlines()`:

Este método consiste en devolver una lista con el contenido, donde cada elemento es una línea del fichero.

```
with open("otro_texto.txt", "r") as archivo_texto_2:
    print(archivo_texto_2.readlines())
```

Resultado:

```
['Las plantas del instituto las tratan bien, sólo falta plantar unos girasoles.
Las gafas se me empañan mucho.\n', 'Necesito descansar unas cinco horas más.\n',
'El aguacate es de lo mejorcito.']
```

Esa lista también la podemos usar para iterar y mostrarlas línea a línea:

```
archivo_texto_2 = open("otro_texto.txt", "r")
lineas = archivo_texto_2.readlines()

for linea in lineas:
    print(linea)
```

Método `writelines()`:

Como explicamos antes, usamos la función write() para escribir y guardar contenido en el fichero. Con el caso de writelines, podemos escribir y guardar una lista:

```
lista_frutas = ["plátano", "fresa", "manzana", "kiwi", "ciruela", "granada",
"pera", "melón", "sandía"]

with open("fichero-tres.txt","w") as fichero_texto_3:
    fichero_texto_3.writelines(lista_frutas)
```

Resultado:

```
plátanofresamanzanakiwiciruelagranadaperamelónsandía
```

Si no queremos que se nos junte el contenido de la lista al guardar, tenemos que añadirle un salto de línea en cada elemento de la lista.

Fuentes:

Concepto de fichero y utilidad
chuwiki | Apertura de ficheros en python
Uniwebsidad - Cerrar archivos en python
With ... as ...
Programador clic, método read()
Pythones.net, método tel()
Programador clic, método seek()
Freecodecamp, uso del bucle for para leer fichero
Fuente: el libro de python