Lectura y escritura de ficheros de textos

Función open()

La función open() se utiliza normalmente con dos parámetros (fichero con el que vamos a trabajar y modo de acceso) y nos devuelve un objeto de tipo fichero.

```
>>> f = open("ejemplo.txt","w")
>>> type(f)
<class '_io.TextIOWrapper'>
>>> f.close()
```

Modos de acceso

Los modos que podemos indicar son los siguientes:

Añadido en modo binario. Crea si éste no existe

Modo	Comportamiento	Puntero
r	Solo lectura	Al inicio del archivo
rb	Solo lectura en modo binario	
r+	Lectura y escritura	Al inicio del archivo
rb+	Lectura y escritura binario	Al inicio del archivo
w	Solo escritura. Sobreescribe si existe. Crea el archivo si no existe.	Al inicio del archivo
wb	Solo escritura en modo binario. Sobreescribe si existe. Crea el archivo si no existe.	Al inicio del archivo
w+	Escritura y lectura. Sobreescribe si existe. Crea el archivo si no existe.	Al inicio del archivo
wb+	Escritura y lectura binaria. Sobreescribe si existe. Crea el archivo si no existe.	Al inicio del archivo
a	Añadido (agregar contenido). Crea el archivo si no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo.
ab		Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo.
a+	Añadido y lectura. Crea el archivo si no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo.
ab+	Añadido y lectura en binario. Crea el archivo si no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo.

Como podemos comprobar podemos trabajar con ficheros binarios y con ficheros de textos.

Codificación de caracteres

Si trabajamos con fichero de textos podemos indicar también el parámetro `encoding` que será la codificación de caracteres utilizadas al trabajar con el fichero, por defecto se usa la indicada en el sistema:

```
>>> import locale
>>> locale.getpreferredencoding()
'UTF-8'
```

Y por último también podemos indicar el parámetro `errors` que controla el comportamiento cuando se encuentra con algún error al codificar o decodificar caracteres.

Objeto fichero

Al abrir un fichero con un determinado modo de acceso con la función `open()` se nos devuelve un objeto fichero. El fichero abierto siempre hay que cerrarlo con el método `close()`:

```
>>> f = open("ejemplo.txt","w")
>>> type(f)
<class '_io.TextIOWrapper'>
>>> f.close()
```

Se pueden acceder a las siguientes propiedades del objeto file:

- `closed` : retorna `True` si el archivo se ha cerrado. De lo contrario, `False`.
- `mode` : retorna el modo de apertura.
- `name` : retorna el nombre del archivo
- `encoding` : retorna la codificación de caracteres de un archivo de texto

Podemos abrirlo y cerrarlo en la misma instrucción con la siguiente estructura:

```
>>> with open("ejemplo.txt", "r") as archivo:
...    contenido = archivo.read()
>>> archivo.closed
True
```

Métodos principales

Métodos de lectura

```
>>> f = open("ejemplo.txt","r")
>>> f.read()
'Hola que tal\n'

>>> f = open("ejemplo.txt","r")
>>> f.read(4)
'Hola'
>>> f.read(4)
' que'
>>> f.tell()
8
>>> f.seek(0)
>>> f.read()
'Hola que tal\n'

>>> f = open("ejemplo2.txt","r")
>>> f.readline()
'Línea 1\n'
>>> f.readline()
'Línea 2\n'
>>> f.seek(0)
0
>>> f.readlines()
['Línea 1\n', 'Línea 2\n']
```

Métodos de escritura

```
>>> f = open("ejemplo3.txt","w")
>>> f.write("Prueba 1\n")
9
>>> print("Prueba 2\n",file=f)
>>> f.writelines(["Prueba 3","Prueba 4"])
>>> f.close()
>>> f = open("ejemplo3.txt","r")
>>> f.read()
'Prueba 1\nPrueba 2\n\nPrueba 3Prueba 4'
```

Recorrido de ficheros

```
>>> with open("ejemplo3.txt","r") as fichero:
...    for linea in fichero:
...        print(linea)
```