# TEMA V

DESARROLLO DE APLICACIONES DE PROPÓSITO GENERAL

## FICHEROS

### 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")
>>> f.close()
```

Los modos que podemos indicar son los siguientes:

| 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. Sobrescribe si existe. Crea el archivo si no existe. | Al inicio del archivo |
| wb | Solo escritura en modo binario. Sobrescribe si existe. Crea el archivo si no existe. | Al inicio del archivo |
| w+ | Escritura y lectura. Sobrescribe si existe. Crea el archivo si no existe. | Al inicio del archivo |
| wb+ | Escritura y lectura binaria. Sobrescribe 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 | Añadir en modo binario | 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. |

Podemos trabajar con ficheros binarios y con ficheros de texto.

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'
```

f=open(ruta, modo, encoding)

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

Siempre cerramos un fichero abierto con el método close()

## Lectura de un Fichero

Las principales funciones para realizar operaciones de lectura

| | |
|---|---|
| read() | lee todo el contenido |
| read(numero) | lee esa cantidad de bytes |
| readline() | lee una línea |
| readline(numero) | lee la línea indicada |
| readlines() | todas las líneas |
| tell() | retorna la posición del puntero |
| Seek(byte) | posiciona el puntero |

## Ejemplos

```
>>> f = open("ejemplo.txt","r")
>>> f.read()
```

```
'Hola que tal\n'
```

Podemos pasarle como parámetro el número de caracteres a leer

```
>>> 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']
```

En importante tener en cuenta, como se aprecia en los ejemplos, que el salto de línea forma parte de la cadena que leemos.

**Ejercicios:**

1. Leer todo el contenido de un fichero de diversas maneras

**Véase EjemplosFicheros uno.py**

## Escritura en un Fichero

| | |
|---|---|
| write() | Escribe la cadena dentro del fichero desde la posición en la que esté el cursor |
| writelines(lista) | Escribe las cadenas contenidas en la lista desde la posición en la que esté el cursor |

## Ejemplos

```
>>> 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'
```

**Véase EjemplosFicheros dos.py**

## Estructura with

La estructura with es una estructura de control cuya función es controlar que el objeto devuelto por la expresión ejecutará un código

with expresión as var

código para var

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

```
>>>with open("ejemplo1.txt", "r") as fichero:
        cotenido=fichero.read()
```

## Módulo CSV

Si necesitas manejar grandes cantidades de datos es probable utilices el formato CSV(comma-separated values). Este es un formato muy popular para almacenar datos estructurados en forma de tabla, en la que las columnas se separan por comas (o por otro carácter).

El módulo cvs nos permite trabajar con ficheros con este formato de forma sencilla

### Leer ficheros CSV

Para leer un fichero CSV utilizamos la función reader()

Csv.reader(fichero, delimitador)

```
>>> import csv
>>> fichero = open("ejemplo1.csv" ,”r”, encoding=”UTF-8”)
>>> contenido = csv.reader(fichero,delimiter=”,”)
>>> datos=list(contenido)
>>> fichero.close()
```

```
[['4/5/2015 13:34', 'Apples', '73'], ['4/5/2015 3:41',
'Cherries', '85'], ['4/6/2015 12:46', 'Pears', '14'],
['4/8/2015 8:59', 'Oranges', '52'], ['4/10/2015 2:07',
'Apples', '152'], ['4/10/2015 18:10', 'Bananas', '23'],
['4/10/2015 2:40', 'Strawberries', '98']]
```

Podemos guardar la lista obtenida en una variable y acceder a ella indicando fila y columna.

```
>>> datos = list(contenido)
>>> datos[0][0]
'4/5/2015 13:34'
>>> datos[1][1]
'Cherries'
>>> datos[2][2]
```

```
'14'
```

Por supuesto podemos recorrer el resultado:

```
>>> for row in contenido:
          print("Fila "+str(contenido.line_num)+" "+str(row))


Fila 1 ['4/5/2015 13:34', 'Apples', '73']
Fila 2 ['4/5/2015 3:41', 'Cherries', '85']
Fila 3 ['4/6/2015 12:46', 'Pears', '14']
Fila 4 ['4/8/2015 8:59', 'Oranges', '52']
```

**Véase EjemplosFicheros tres.py**

Podemos utilizar la función csv.DictReader para leer datos de un archivo CSV y convertirlos en un diccionario. Las claves de cada fila serán los datos de la primera fila, por tanto debemos tener una primera fila para dichas claves.

## Escribir ficheros CSV

Para escribir usamos la función writer()

```
>>> import csv
>>> fichero = open("ejemplo3.csv","w")
>>> contenido = csv.writer(fichero)
>>> contenido.writerow(['4/5/2015 13:34', 'Apples', '73'])
>>> contenido.writerows(['4/5/2015 3:41', 'Cherries',
'85'],['4/6/2015 12:46', 'Pears', '14'])
>>> fichero.close()

$ cat ejemplo3.csv
4/5/2015 13:34,Apples,73
4/5/2015 3:41,Cherries,85
4/6/2015 12:46,Pears,14
```

**Véase EjemplosFicheros cuatro.py**

## Módulo JSON

El módulo json nos permite gestionar ficheros con formato JSON (JavaScript Object Notation).

La correspondecia entre JSON y Python la podemos resumir en la siguiente tabla:

| JSON | Python |
|---|---|
| object | dict |
| array | list |
| string | str |
| number (int) | int |
| number (real) | float |
| true | True |
| false | False |
| null | None |

### Leer ficheros json

Desde una cadena de caracteres:

```
>>> import json
>>> datos_json='{"nombre":"carlos","edad":23}'
>>> datos = json.loads(datos_json)
>>> type(datos)
<class 'dict'>
>>> print(datos)
{'nombre': 'carlos', 'edad': 23}
```

Desde un fichero:

```
>>> with open("ejemplo1.json") as fichero:
...   datos=json.load(fichero)


>>> type(datos)
<class 'dict'>
>>> datos
{'bookstore': {'book': [{'_category': 'COOKING', 'price':
'30.00', 'author': 'Giada De Laurentiis', 'title': {'__text':
'Everyday Italian', '_lang': 'en'}, 'year': '2005'},
{'_category': 'CHILDREN', 'price': '29.99', 'author': 'J K.
Rowling', 'title': {'__text': 'Harry Potter', '_lang': 'en'},
'year': '2005'}, {'_category': 'WEB', 'price': '49.99',
```

```
'author': ['James McGovern', 'Per Bothner', 'Kurt Cagle',
'James Linn', 'Vaidyanathan Nagarajan'], 'title': {'__text':
'XQuery Kick Start', '_lang': 'en'}, 'year': '2003'},
{'_category': 'WEB', 'price': '39.95', 'author': 'Erik T.
Ray', 'title': {'__text': 'Learning XML', '_lang': 'en'},
'year': '2003'}]}}
```

### Escribir ficheros json

```
>>> datos = {'isCat': True, 'miceCaught': 0, 'name':
'Zophie','felineIQ': None}
>>> fichero = open("ejemplo2.json","w")
>>> json.dump(datos,fichero)
>>> fichero.close()
```