# E/S de datos. Ficheros

## Programación de Inteligencia Artificial


Junta de Andalucía

FPA
FORMACIÓN
PROFESIONAL
ANDALUZA

Fondo Social Europeo

# Índice | Ficheros/Texto plano | CSV | JSON

## Ficheros/Texto plano:

- Introducción.
- Abrir fichero. Modos.
- Atributos y métodos.

## CSV:

- Introducción.
- Primeros pasos.
- Dialectos.
- Lectura de datos.
- Escritura de datos.

## JSON:

- Introducción.
- Cargar un JSON.
- Escribir en un fichero JSON.
- Tabla de equivalencias entre Python y JSON.

- En un principio, Python estaba pensado para hacer operaciones de sistema, dando lugar a multitud de herramientas.
- Podemos trabajar con distintos archivos. Veremos como hacerlo con archivos de texto plano, CSV y JSON.
- En este enlace podéis encontrar más información sobre la librería.

- Para abrir ficheros se usa la función *open(nombre_fichero, tipo_acceso)*. El segundo parámetro no es necesario ponerlo si se abre en modo lectura.

| Sim | Modo | Información adicional |
|---|---|---|
| r | Lectura. | Puntero: inicio.<br>+: lectura y escritura de los datos. |
| w | Escritura de un nuevo archivo. | Sobrescribe si existe. Si no lo crea.<br>Puntero: inicio. |
| a | Escritura después de los datos existentes. | Si no existe el archivo lo crea.<br>Puntero: al final de los datos. |
| t | Modo texto. | Se suele omitir. |
| b | Modo binario. | |
| + | Escritura. | |

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

- f será un objeto del tipo *_io.TextIOWrapper*.

- closed: devuelve *verdadero* si un archivo se ha cerrado.
- mode: indica el modo en el que se está accediendo al archivo.
- name: devuelve el nombre del archivo.
- close(): cierra el flujo de datos con el archivo.
- read(): lee todo el contenido de un archivo.
- readable(): devuelve *true* si el archivo se puede leer.
- readline(): devuelve una cadena de caracteres con la línea leída.
- readlines(): devuelve una lista con las distintas líneas leídas de un archivo.
- seek(posicion): coloca el puntero en la posición indicada en el parámetro.
- tell(): devuelve la posición actual del puntero.
- write(texto): escribe el texto pasado por parámetro.
- writable(): devuelve *true* si se puede escribir en el archivo.
- writelines(lista): escribe las líneas contenidas en la lista.

```
f = open("fichero.txt", "w")
f.write("Hola fichero\n")
f.close()

if(f.closed):
    f = open("fichero.txt")
    print(f.read())
    f.close()
```

- Se puede abrir un archivo mediante el uso de *with* y este se cerrará automáticamente cuando se salga de este bloque.

```
with open("fichero.txt", "a+") as f:
    f.write("Hola universo\n")

if(f.closed):
    with open("fichero.txt") as f:
        print(f.read())
```

- Se trata de ficheros en texto plano que guardan información.
- Por regla general, los datos están separados por comas. De ahí viene su nombre: Comma Separated Values.
- Cada línea guarda una serie de información relacionada entre sí.
- En ocasiones, la primera línea se utiliza para identificar los distintos campos (como si se tratase del encabezado de una tabla de datos).

```
Nombre, Apellidos, Nacimiento, Fallecimiento
Almudena, Grandes, 1960, 2021
Elvira, Lindo, 1962, -
María, Dueñas, 1964, -
Julia, Navarro, 1953, -
Nieves, Concostrina, 1961, -
Ana María, Matute, 1925, 2014
Carme, de Burgos, 1867, 1932
```

- Python ya dispone de un módulo que permite trabajar con archivos CSV, el cual es necesario importar.

```
import csv
```

- Lo primero para poder trabajar con un archivo CSV es abrirlo, tal como hemos visto con anterioridad.

```
import csv
# Opcion 1
f = open("D:\\datos.csv")
datos = csv.reader(f)
print(datos) # Obtenemos información sobre el objeto datos
print(list(datos))
f.close()

# Opción 2. Al usar el with el archivo se cierra automáticamente cuando se sale del “with”
with open("D:\\datos.csv") as f:
    datos = list(csv.reader(f))
print(datos)
```

- En el ejemplo se ve como abrir el archivo de dos formas distintas y pasar su contenido a una lista.
- También se puede especificar la codificación en la que está creado el archivo.

```
with open("D:\\datos.csv", encoding='utf-8') as f:
    ...
    ...
```

- No existe un formato o estándar bien definido para CSV, es decir, los parámetros de formateo de un archivo de datos. Al conjunto de estos parámetros se le llama *dialecto*. En estos enlaces (e1, e2) se pueden ver todos los parámetros de un dialecto.
- El módulo tiene por defecto los dialectos *excel, excel-tab* y *unix*.
- Podemos leer un dialecto mediante la función *sniffer* de la siguiente manera.

```
import csv
with open("D:\\datos.csv", encoding='utf-8') as f:
    dialecto = csv.Sniffer().sniff(f.readline())
    datos = list(csv.reader(f, dialect = dialecto))

print(datos)
```

- Tenemos que tener en cuenta que, tal cual se ha planteado el ejemplo, el puntero de lectura estaría en la segunda línea y la primera no la almacenaría en la lista.
- Podemos crearnos nuestros propios dialectos y añadirlos a la lista de dialectos mediante la función *register_dialect* del módulo de *csv*.

- Normalmente se almacenan los datos en una secuencia.
- Podemos obtener una lista a partir de los datos (tal como hemos visto).

```
import csv
with open("D:\\datos.csv", encoding='utf-8') as f:
    datos = list(csv.reader(f))

print(datos)
```

- Otra opción, es crear un diccionario a partir de la información con *DictReader*.

```
import csv

f = open("D:\\datos.csv", encoding='utf-8')
datos = csv.DictReader(f)

for d in datos.fieldnames:
   print(f'{d:15}', end='')
print("\n")

for d in datos:
   print(f'{d["Nombre"]:15}{d["Apellidos"]:15}{d["Nacimiento"]:15}{d["Fallecimiento"]:15}')
f.close()
```

- Si queremos usar el *with* toda la operación debe de estar dentro de él.

- Para escribir en un archivo debemos: abrir el fichero, escribir y cerrarlo.
- Se utiliza la función *writer, writerow* y *writerows*.

```
import csv
# Opción 1
f = open("D:\\datos.csv", "a", newline='')
escribir = csv.writer(f)
escribir.writerow(["Gloria", "Fuertes", "1917", "1998"])
escribir.writerows([["Rosalía", "de Castro", "1837", "1885"],
                    ["María", "Zambrano", "1904", "1991"]])
f.close()

# Opción 2
with open("D:\\Drive-IES\\IES La Puebla\\01 - PIA\\1 - Introducción a Python\\datos.csv",
"a", newline='') as f:
    escribir = csv.writer(f)
    escribir.writerow(["Gloria", "Fuertes", "1917", "1998"])
    escribir.writerows([["Rosalía", "de Castro", "1837", "1885"],
                        ["María", "Zambrano", "1904", "1991"]])
```

- Es importante poner el parámetro *newline=''*, ya que sino nos añadirá un salto de línea adicional por cada fila de información que grabemos en el fichero.

- También se puede guardar la información como un diccionario.
- Los datos deben de venir como una lista de diccionarios. Se usa el objeto *DictWriter* y la funciones que ya hemos visto: *writerow* y *writerows.*

```
import csv
with open("D:\\datos.csv", "a", newline='') as f:
    escribir = csv.DictWriter(f, ("Nombre", "Apellidos", "Nacimiento", "Fallecimiento"))
    escribir.writerows([{"Nombre":"Gloria", "Apellidos":"Fuertes",
                         "Nacimiento":"1917", "Fallecimiento":"1998"},
                        {"Nombre":"Rosalía", "Apellidos":"de Castro",
                         "Nacimiento":"1837", "Fallecimiento":"1885"},
                        {"Nombre":"María", "Apellidos":"Zambrano",
                         "Nacimiento":"1904", "Fallecimiento":"1991"}])
```

- Al igual que CSV, se trata de un archivo en texto plano que guarda información, originalmente pensado para JavaScript.
- Sus siglas significan *JavaSript Object Notation* o lo que es lo mismo, notación de objetos de JavaScript.
- Es parecido a como Python representa los diccionarios. Está formado por: claves y valores.

```
[
    {
        "Nombre": "Almudena",
        "Apellidos": "Grandes",
        "Nacimiento": "1960",
        "Fallecimiento": "2021"
    },
    {
        "Nombre": "Elvira",
        "Apellidos": "Lindo",
        "Nacimiento": "1962",
        "Fallecimiento": "-"
    }
]
```

- Python trae un módulo para trabajar con JSON.

```
import json
```

- Podemos pasar una cadena de caracteres JSON a un diccionario mediante la función *loads*.

```
import json
escritora =  """
{
    "Nombre": "Almudena",
    "Apellidos": "Grandes",
    "Nacimiento": "1960",
    "Fallecimiento": "2021",
    "Libros": ["Las edades de Lulú", "Te llamaré viernes"]
}
"""
print(type(escritora))
diccionario = json.loads(escritora)
print(type(diccionario))
print(diccionario)
```

- Para leer desde un archivo JSON, a parte de “abrirlo”, se usa el método *load*.

```
import json
with open("D:\\datos.json", encoding = 'utf-8') as f:
    datos = json.load(f)
    print(type(datos))
    print(datos)
```

- Para la escritura se usa la función *dump*, indicando como parámetro los datos y el fichero de salida. Para que no se grabe en una única línea se puede modificar la indentación mediante el parámetro *indent*.

```
import json
datos = [{"Nombre":"Gloria", "Apellidos":"Fuertes",
          "Nacimiento":"1917", "Fallecimiento":"1998"},
         {"Nombre":"Rosalía", "Apellidos":"de Castro",
          "Nacimiento":"1837", "Fallecimiento":"1885"},
         {"Nombre":"María", "Apellidos":"Zambrano",
          "Nacimiento":"1904", "Fallecimiento":"1991"}]

with open("D:\\datos.json", "w") as f:
    json.dump(datos, f, indent=4)
```

- La relación entre los tipos y objetos de JSON y Python es la siguiente tabla ( enlace a la documentación oficial).

| Python | JSON |
|---|---|
| dict | object |
| list, tuple | array |
| str | string |
| int, float, Enums derivadas de int o float | number |
| True | true |
| False | false |
| None | null |


Instituto de Educación Secundaria PUEBLA DE VICAR

Junta de Andalucía

FPA
FORMACIÓN PROFESIONAL ANDALUZA

Fondo Social Europeo

*Imágenes: Flaticon.com, IES La Puebla, Junta de Andalucía*