01_Lectura_Escritura_txt

November 29, 2023

0.1 Acceso Local: Lectura, Escritura y ficheros TXT

En esta unidad vas a ver diferentes maneras de leer y escribir datos desde archivos locales y, en general, volcarlos a un `DataFrame`, aunque lo que aprenderás te servirá para cualquier otro procesamiento que necesites. Rara vez trabajarás únicamente con los datos que genere tu programa de Python, sino que lo normal será acudir a una fuente de datos, o leer de algún archivo.

En concreto en esta sesión veremos la forma genérica de leer y escribir archivos que tiene Python y lo aplicaremos a lo que hemos llamado ficheros de texto plano.

0.2 1. Archivos

Antes de ir a leer o escribir archivos, es importante saber exáctamente qué es un archivo. Recordando las primeras pílldoras del bootcamp: **Un archivo es un conjunto de datos almacenados en el ordenador en forma de bits.** Los datos se organizan en un formato específico, pudiendo ser un archivo de texto, un ejecutable, etc como comentamos en las introducciones de la unidad anterior, pero en el fondo todos esos archivos se traducen a nivel binario para el procesado del ordenador.

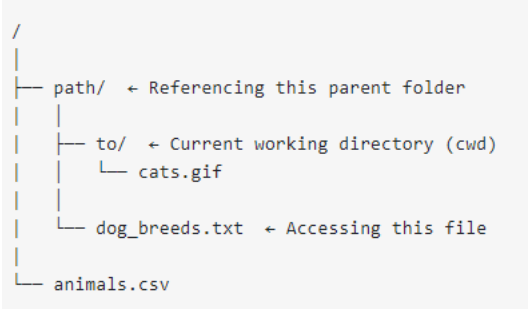
Poniéndonos formales, los archivos se componen de:

1. **Header**: metadatos del archivo (nombre, tamaño, tipo...)
2. **Data**: contenido del archivo
3. **End of file (EOF)**: caracter especial que indica el final del archivo.

Aunque tú al abrir un archivo de texto por ejemplo o un word no veras estas partes internamente el ordenador sí que las considera.

File path (a vueltas con) Hay tres elementos que tenemos que conocer cuando leamos un archivo: 1. **Folder path**: en que lugar del ordenador está el archivo. 2. **File name** 3. **Extension**: lo que va después del punto

Fíjate en la siguiente imagen:



- Si estamos trabajando en el directorio *to*, accederemos a *cats.gif* como `cats.gif`
- Si queremos leer *dog_breeds.txt*, hay que ir un directorio hacia atrás, `../dog_breeds.txt`
- Y si queremos acceder a `animals.csv`, son dos directorios hacia atrás: `../../animals.csv`

Recordarás que lo anterior son rutas relativas (no empiezan en C: o en /). Siempre podemos poner la ruta absoluta (`/animals.csv`, para el caso del ejemplo de la figura, claro) para el acceso a cada archivo, **aunque no es lo recomendable**.

Aquí te dejo una buena guía para iniciarse en la lectura/escritura de archivos con Python.

0.3 2. Abrir ficheros

Aunque veremos la forma concreta que tiene pandas para abrir archivo (una de las cuales pd.read_csv ya la has visto bastantes veces), Python tiene sus propias funciones *built-in*, para tratar con ficheros. Básicamente: `open`.

Para ello **usaremos la función `open`**, que devuelve un objeto de tipo `File`, con unos métodos y atributos propios empleados para obtener información de los archivos abiertos. `open` sigue la siguiente sitaxis:

```
file_object  = open("filename", "mode")
```

El primer argumento es el nombre del archivo, mientras que en el modo tendremos que especificar si queremos leer, o escribir. Por defecto leerá, es decir, el parámetro valdrá *r*, de read. Te dejo el enlace a la documentación para consultar el resto de modos.

Vamos a probar a leer un archivo:

[2]:
```
with open(".\data\dog_breeds.txt","r") as open_file:
    all_text = open_file.read()
    print(type(all_text))
    print(all_text)
```

```
<class 'str'>
Pug
Jack Russell Terrier
English Springer Spaniel
German Shepherd
Staffordshire Bull Terrier
Cavalier King Charles Spaniel
Golden Retriever
West Highland White Terrier
Boxer
Border Terrier
```

La sintaxis de línea que has visto es la recomendada, porque en algún momento se tiene que cerrar el archivo. Se abre, leemos, realizamos operaciones, y cuando acaba el `with open()`, se cierra el archivo.

Leer y escribir mientras los archivos están abiertos nos dará errores.

[3]:
```
with open ("./data/dog_breeds.txt","r") as open_file:
    all_text = open_file.readlines()
    print(type(all_text))
    print(all_text)
```

```
<class 'list'>
['Pug\n', 'Jack Russell Terrier\n', 'English Springer Spaniel\n', 'German
Shepherd\n', 'Staffordshire Bull Terrier\n', 'Cavalier King Charles Spaniel\n',
'Golden Retriever\n', 'West Highland White Terrier\n', 'Boxer\n', 'Border
Terrier']
```

El método `.read()` nos devuelve un string con todo el texto, que no es lo ideal para tratar luego los datos.

En el siguiente ejemplo vemos como también lo leemos, pero en este caso cada línea la guarda en una lista.

[10]:
```
from pyparsing import line


with open ("./data/dog_breeds.txt","r") as f:
    lineas = [line.replace("\n","") for line in f]

lineas
```

[10]:
```
['Pug',
 'Jack Russell Terrier',
 'English Springer Spaniel',
 'German Shepherd',
 'Staffordshire Bull Terrier',
 'Cavalier King Charles Spaniel',
 'Golden Retriever',
 'West Highland White Terrier',
 'Boxer',
 'Border Terrier']
```

[14]:
```
nuevos_datos =[perrito.replace("Terrier","").replace("Spaniel","") for perrito␣
 ↪in lineas]
```

[15]:
```
nuevos_datos
```

[15]:
```
['Pug',
 'Jack Russell ',
 'English Springer ',
 'German Shepherd',
 'Staffordshire Bull ',
 'Cavalier King Charles ',
 'Golden Retriever',
 'West Highland White ',
 'Boxer',
 'Border ']
```

Aunque a mí el estilo que me gusta es: (no por ello es mejor)

[ ]:
```

```

Con el texto ya almacenado en una variabe o en las que queramos podemos procesarlo. Hagamos un procesamiento simple y escribámoslo.

[ ]:
```

```

Y ahora para escribir sólo tenemos que cambiar el modo a ”w” (sobreescribiremos lo que haya) o ”a” (añadiremos al final)

[17]:
```
with open("./data/new_breed.txt", "w") as g:
    for linea in nuevos_datos:
        g.write(linea+ "\n")

with open("./data/new_breed.txt", "r") as f:
    for lineas in f:
        print(linea, end= "")
```

```
Border Border Border Border Border Border Border Border Border Border
```

[ ]:

Hemos visto como abrir, leer y escrbir ficheros de texto, con el modificado ”b” en los argumentos del open podríamos abrir, leer y escribir ficheros binarios, es decir cualquier tipo de ficheros pero cómo interpretar lo leído depende de cada fichero, así que por ahora trabaja con ficheros ”.txt”, hasta que nos adentremos en csv, json y xml en las píldoras siguientes.

02_Ficheros_CSV

November 29, 2023

ETL Y DATOS

0.1 ACCESO LOCAL: Ficheros .csv

Los ficheros csv (*Comma Separated Values*) son el estándar de la industria que se utiliza para leer/escribir datos en formato tabla, en dos dimensiones. Se llaman *Comma Separeted Values* ya que todos los valores de las columnas van separados por comas, y las filas por saltos de línea. **Su extension de archivo es `.csv`**. Además, el 99% de las veces llevan la cabecera de columnas en la primera línea. Aunque no siempre se dará el caso, depende de la manera en la que se haya generado el CSV.

Es el archivo más común utilizado para guardar datos tabulares, puesto que ocupa muy poco espacio ya que es simplemente un archivo de texto plano, con todos los datos separados por el caracter coma. Y además, sencillo de entender, los datos no van en un árbol json o xml... Si lo abrimos como texto plano, son los datos separados por coma, tal cual.

Por supuesto, tenemos el otro gran protagonista en cuanto a almacenamiento de datos en formato tabla, **el Excel**. A ver, son cosas diferentes. El Excel tiene sus formatos (.xlsx, .xls), que encima son muy eficientes ya que el dato va comprimido, pero no deja de ser un software de pago para tratar los datos, mientras que **el CSV es un formato estándar que se utiliza en todos los sistemas operativos para el exportado/importado de datos**.

Como decíamos al principio, los CSVs se llaman *Comma Separated Values* porque todos los valores van separados por comas... bueno, esto no es del todo cierto ya que **puede haber otro caracter que no sea la coma**, como por ejemplo el punto y coma. ¿Por qué? Simplemente porque si tenemos datos decimales, separados por comas, no vamos a saber distinguir cuando una coma es de un decimal, o es el separador de columnas.

0.1.1 Fichero csv como fichero de texto

Como es un fichero de texto plano podemos leer los ficheros csv como hemos visto en la sesión anterior:

[2]:
```
# Leamos uno de los ficheros csv que hemos visto en sesiones anteriores
with open("./data/df_liga_2019.csv", "r", encoding = "utf8") as f: # cuaudo␣
 ↪falle algun fichero de texto incluye al final incoding = utf8 o latin1)
    datos =[linea.replace("\n", "") for linea in f]


# Veamos las primeras 10 filas
datos[0:10]
```

[2]:
```
['id_partido,equipo_local,equipo_visitante,Division,Temporada,fecha_dt,goles_loc
al,goles_visitante,arbitro,estadio,odd_1,odd_x,odd_2,Informe_Tarjetas',
 '214023,Celta Vigo,Real Madrid,1,2019,2019-08-17 17:00:00,1,3,Javier
Estrada,Abanca-Balaídos,4.75,4.2,1.65,Hubo 01 tajetas rojas al equipo
visitante;Hubo 0 rojas a jugadores del equipo local;Hubo 2 tarjetas amarillas
para el equipo visitantes;Hubo 6 amarillas para jugadores del equipo local',
 '214403,Racing Santander,Malaga,2,2019,2019-08-17 18:00:00,0,1,Aitor
Gorostegui,Campos de Sport de El Sardinero,2.87,3.1,2.55,Hubo 03 amarillas
mostradas al equipo local;Hubo 2  tarjetas amarillas de jugadores
visitantes;Hubo 00 rojas a jugadores visitantes;Hubo 1 rojas a jugadores del
equipo local',
 '214024,Valencia,Real Sociedad,1,2019,2019-08-17 19:00:00,1,1,Jesús Gil,Estadio
de Mestalla,1.66,3.75,5.5,Hubo 4 amarillas mostradas al equipo local;Hubo 1
tarjetas rojas sobre el equipo local;Hubo 4 tarjetas amarillas para el equipo
visitantes;Hubo 0 tajetas rojas al equipo visitante',
 '214404,Almeria,Albacete,2,2019,2019-08-17 19:00:00,3,0,Saúl Ais,Estadio de los
Juegos Mediterráneos,2.37,3.1,3.1,Hubo 00 rojas a jugadores visitantes;Hubo 01
amarillas mostradas al equipo local;Hubo 1  tarjetas amarillas de jugadores
visitantes;Hubo 0 tarjetas rojas sobre el equipo local',
 '214026,Villarreal,Granada CF,1,2019,2019-08-17 21:00:00,4,4,Adrián
Cordero,Estadio de la Cerámica,1.6,3.8,6.5,Hubo 01  tarjetas amarillas de
jugadores visitantes;Hubo 00 tajetas rojas al equipo visitante;Hubo 03 amarillas
para jugadores del equipo local;Hubo 00 rojas a jugadores del equipo local',
 '214025,Leganes,Osasuna,1,2019,2019-08-17 21:00:00,0,1,Javier Alberola,Estadio
Municipal de Butarque,2.0,3.2,4.2,Hubo 01 amarillas mostradas al equipo
local;Hubo 1 rojas a jugadores del equipo local;Hubo 4  tarjetas amarillas de
jugadores visitantes;Hubo 0 rojas a jugadores visitantes',
 '214406,Zaragoza,Tenerife,2,2019,2019-08-17 21:00:00,2,0,Dámaso
Arcediano,Estadio de la Romareda,2.0,3.3,3.8,Hubo 3 tarjetas amarillas para el
equipo visitantes;Hubo 01 amarillas para jugadores del equipo local;Hubo 0
tarjetas rojas sobre el equipo local;Hubo 1 tajetas rojas al equipo visitante',
 '214405,Rayo Vallecano,Mirandes,2,2019,2019-08-17 21:00:00,2,2,Juan
Pulido,Estadio de Vallecas,1.53,3.75,7.0,Hubo 1 rojas a jugadores
visitantes;Hubo 01 tarjetas rojas sobre el equipo local;Hubo 00 amarillas para
```

```
jugadores del equipo local;Hubo 4  tarjetas amarillas de jugadores visitantes',
 '214027,Alaves,Levante,1,2019,2019-08-18 17:00:00,1,0,César Soto,Estadio de
Mendizorroza,2.15,3.2,3.6,Hubo 01  tarjetas amarillas de jugadores
visitantes;Hubo 0 rojas a jugadores visitantes;Hubo 2 amarillas para jugadores
del equipo local;Hubo 0 tarjetas rojas sobre el equipo local']
```

Se ve que cada campo está separado por comas y además que la primera línea es el nombre de las columnas... Está a h... tiro para leerlo con Pandas, que es lo que haremos con este tipo de ficheros, no pasaremos por la lectura con `open`

0.1.2 Pandas y csv: Lectura

¿Cómo podemos leer un CSV en Python con Pandas? Aquí tienes la documentación

Pero ya lo hemos hecho tantas veces que te será familiar, usando el método `read_csv`:

[5]:
```
import pandas as pd

df = pd.read_csv("./data/df_liga_2019.csv")
df.head()
```

[5]:

	id_partido	equipo_local	equipo_visitante	Division	Temporada	fecha_dt	goles_local	goles_visitante	arbitro	estadio	odd_1	odd_x	odd_2	Informe_Tarjetas
0	214023	Celta Vigo	Real Madrid	1	2019	2019-08-17 17:00:00	1	3	Javier Estrada	Abanca-Balaídos	4.75	4.20	1.65	Hubo 01 tajetas rojas al equipo visitante;Hubo…
1	214403	Racing Santander	Malaga	2	2019	2019-08-17 18:00:00	0	1	Aitor Gorostegui	Campos de Sport de El Sardinero	2.87	3.10	2.55	Hubo 03 amarillas mostradas al equipo local;Hu…
2	214024	Valencia	Real Sociedad	1	2019	2019-08-17 19:00:00	1	1	Jesús Gil	Estadio de Mestalla	1.66	3.75	5.50	Hubo 4 amarillas mostradas al equipo local;Hub…
3	214404	Almeria	Albacete	2	2019	2019-08-17 19:00:00	3	0	Saúl Ais	Estadio de los Juegos Mediterráneos	2.37	3.10	3.10	Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
4	214026	Villarreal	Granada CF	1	2019	2019-08-17 21:00:00	4	4	Adrián Cordero	Estadio de la Cerámica	1.60	3.80	6.50	Hubo 01 tarjetas amarillas de jugadores visit…

Vamos a ver algunos **parámetros interesantes del `read_csv()`** 1. `index_col`: indica cual de las columnas queremos que sea el índice (necesitamos que tenga cabecera claro, ver más abajo) 2. `names`: sirve para indicar el nombre de las columnas, por si no queremos el que venga en el fichero (ojo, es posicional, la primera columna se llamará como el primer elemento del argumento names, y así sucesivamente) 2. `sep`: el separador de los datos, por defecto es coma, pero podría ser otro como veremos en ejemplos posteriores. 3. `header`: dónde se encuentran los nombre de columnas. Por defecto es en la primera línea. [En general se usa para indicar que no viene con cabecera, entonces tendremos que usar names para dar el nombre de las columnas]

Probemos a leer el CSV desde otra ruta del ordenador

index_col

[7]:
```
df = pd.read_csv("./data/df_liga_2019.csv", index_col ="id_partido")
df
```

[7]:

id_partido	equipo_local	equipo_visitante	Division	Temporada	fecha_dt	goles_local	goles_visitante	arbitro	estadio	odd_1
214023	Celta Vigo	Real Madrid	1	2019	2019-08-17 17:00:00	1	3	Javier Estrada	Abanca-Balaídos	4.75
214403	Racing Santander	Malaga	2	2019	2019-08-17 18:00:00	0	1	Aitor Gorostegui	Campos de Sport de El Sardinero	2.87
214024	Valencia	Real Sociedad	1	2019	2019-08-17 19:00:00	1	1	Jesús Gil	Estadio de Mestalla	1.66
214404	Almeria	Albacete	2	2019	2019-08-17 19:00:00	3	0			
214026	Villarreal	Granada CF	1	2019	2019-08-17 21:00:00	4	4			
…	…	…	…	…	…	…	…			
214853	Alcorcon	Girona	2	2019	2020-07-20 21:00:00	2	0			
214863	Zaragoza	Ponferradina	2	2019	2020-07-20 21:00:00	2	1			
214854	Almeria	Malaga	2	2019	2020-07-20 21:00:00	0	0			
214862	Sporting Gijon	Huesca	2	2019	2020-07-20 21:00:00	0	1			
214856	Deportivo La Coruna	Fuenlabrada	2	2019	2020-08-07 20:00:00	2	1			

```
214404                Saúl Ais  Estadio de los Juegos Mediterráneos   2.37
214026          Adrián Cordero               Estadio de la Cerámica   1.60
…                          …                                    …    …
214853            Juan Pulido                Estadio Santo Domingo   2.37
214863       Dámaso Arcediano               Estadio de la Romareda   2.10
214854                Saúl Ais  Estadio de los Juegos Mediterráneos   2.10
214862           Gorka Sagues         Estadio Municipal El Molinón   3.30
214856     Isidro Díaz de Mera                Estadio Abanca-Riazor   2.10

            odd_x  odd_2                                Informe_Tarjetas
id_partido
214023       4.20   1.65  Hubo 01 tajetas rojas al equipo visitante;Hubo…
214403       3.10   2.55  Hubo 03 amarillas mostradas al equipo local;Hu…
214024       3.75   5.50  Hubo 4 amarillas mostradas al equipo local;Hub…
214404       3.10   3.10  Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
214026       3.80   6.50  Hubo 01  tarjetas amarillas de jugadores visit…
…             …    …                                                  …
214853       2.87   3.40  Hubo 0 tajetas rojas al equipo visitante;Hubo …
214863       3.30   3.50  Hubo 00  tarjetas amarillas de jugadores visit…
214854       3.20   3.60  Hubo 2  tarjetas amarillas de jugadores visita…
214862       3.10   2.15  Hubo 2 amarillas para jugadores del equipo loc…
214856       3.20   3.60  Hubo 02 amarillas para jugadores del equipo lo…

[592 rows x 13 columns]
```

names También es posible aplicarle nombres de columnas en la lectura de los datos

[8]:
```
df.columns
```

[8]:
```
Index(['equipo_local', 'equipo_visitante', 'Division', 'Temporada', 'fecha_dt',
       'goles_local', 'goles_visitante', 'arbitro', 'estadio', 'odd_1',
       'odd_x', 'odd_2', 'Informe_Tarjetas'],
      dtype='object')
```

[9]:
```
df = pd.read_csv("./data/df_liga_2019.csv",
                 names =␣
 ↪["id_fixture","home_team","away_team","division","season","date_dt",
                         ␣
 ↪"goals_home","goals_away","referee","stadium","odd_1","odd_draw","odd_2",
                          "Card_report"])
df
```

[9]:
```
     id_fixture           home_team         away_team  division     season  \
0    id_partido        equipo_local  equipo_visitante  Division  Temporada
1        214023          Celta Vigo       Real Madrid         1       2019
2        214403    Racing Santander            Malaga         2       2019
3        214024             Valencia     Real Sociedad         1       2019
```

```
4        214404              Almeria          Albacete         2       2019
..          …                   …                …       …        …
588      214853             Alcorcon            Girona         2       2019
589      214863             Zaragoza      Ponferradina         2       2019
590      214854              Almeria            Malaga         2       2019
591      214862       Sporting Gijon            Huesca         2       2019
592      214856  Deportivo La Coruna       Fuenlabrada         2       2019

                 date_dt   goals_home       goals_away              referee  \
0              fecha_dt  goles_local  goles_visitante              arbitro
1    2019-08-17 17:00:00            1                3       Javier Estrada
2    2019-08-17 18:00:00            0                1     Aitor Gorostegui
3    2019-08-17 19:00:00            1                1            Jesús Gil
4    2019-08-17 19:00:00            3                0             Saúl Ais
..                   …          …              …                   …
588  2020-07-20 21:00:00            2                0          Juan Pulido
589  2020-07-20 21:00:00            2                1     Dámaso Arcediano
590  2020-07-20 21:00:00            0                0             Saúl Ais
591  2020-07-20 21:00:00            0                1         Gorka Sagues
592  2020-08-07 20:00:00            2                1  Isidro Díaz de Mera

                                stadium  odd_1 odd_draw  odd_2  \
0                               estadio  odd_1    odd_x  odd_2
1                         Abanca-Balaídos   4.75      4.2   1.65
2          Campos de Sport de El Sardinero   2.87      3.1   2.55
3                      Estadio de Mestalla   1.66     3.75    5.5
4    Estadio de los Juegos Mediterráneos   2.37      3.1    3.1
..                                     …    …       …    …
588                Estadio Santo Domingo   2.37     2.87    3.4
589                Estadio de la Romareda    2.1      3.3    3.5
590  Estadio de los Juegos Mediterráneos    2.1      3.2    3.6
591          Estadio Municipal El Molinón    3.3      3.1   2.15
592                 Estadio Abanca-Riazor    2.1      3.2    3.6

                                       Card_report
0                                   Informe_Tarjetas
1    Hubo 01 tajetas rojas al equipo visitante;Hubo…
2    Hubo 03 amarillas mostradas al equipo local;Hu…
3    Hubo 4 amarillas mostradas al equipo local;Hub…
4    Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
..                                                 …
588  Hubo 0 tajetas rojas al equipo visitante;Hubo …
589  Hubo 00  tarjetas amarillas de jugadores visit…
590  Hubo 2  tarjetas amarillas de jugadores visita…
591  Hubo 2 amarillas para jugadores del equipo loc…
592  Hubo 02 amarillas para jugadores del equipo lo…
```

```
[593 rows x 14 columns]
```

sep El argumento **sep** nos permite leer un archivo CSV que no esté separado por comas.

Primero, probemos a leer un archivo CSV, que no tiene comas como delimitador y ver qué ocurre:

[ ]:
```
df_co
```

[12]:
```
df = pd.read_csv("./data/df_liga_2019_pipe.csv")
df
```

[12]:
```
    id_partido|equipo_local|equipo_visitante|Division|Temporada|fecha_dt|goles_l
ocal|goles_visitante|arbitro|estadio|odd_1|odd_x|odd_2|Informe_Tarjetas
0    214023|Celta Vigo|Real Madrid|1|2019|2019-08-1…
1    214403|Racing Santander|Malaga|2|2019|2019-08-…
2    214024|Valencia|Real Sociedad|1|2019|2019-08-1…
3    214404|Almeria|Albacete|2|2019|2019-08-17 19:0…
4    214026|Villarreal|Granada CF|1|2019|2019-08-17…
..                                                …
587  214853|Alcorcon|Girona|2|2019|2020-07-20 21:00…
588  214863|Zaragoza|Ponferradina|2|2019|2020-07-20…
589  214854|Almeria|Malaga|2|2019|2020-07-20 21:00:…
590  214862|Sporting Gijon|Huesca|2|2019|2020-07-20…
591  214856|Deportivo La Coruna|Fuenlabrada|2|2019|…

[592 rows x 1 columns]
```

[13]:
```
df.columns
```

[13]:
```
Index(['id_partido|equipo_local|equipo_visitante|Division|Temporada|fecha_dt|gol
es_local|goles_visitante|arbitro|estadio|odd_1|odd_x|odd_2|Informe_Tarjetas'],
dtype='object')
```

Lo lee todo como una única línea ya que no encuentra comas. **Se recomienda trajar con CSVs cuyo separador sea el ";" o el "|" así evitamos problemas por los decimales**.

[16]:
```
df = pd.read_csv("./data/df_liga_2019_pipe.csv", sep= "|", index_col =␣
↪"id_partido")
df
```

[16]:
```
                  equipo_local equipo_visitante  Division  Temporada  \
id_partido
214023               Celta Vigo      Real Madrid         1       2019
214403         Racing Santander           Malaga         2       2019
214024                 Valencia    Real Sociedad         1       2019
214404                   Almeria         Albacete         2       2019
214026                Villarreal       Granada CF         1       2019
…                            …              …       …        …
```

```
214853                Alcorcon           Girona         2       2019
214863                Zaragoza     Ponferradina         2       2019
214854                 Almeria           Malaga         2       2019
214862          Sporting Gijon           Huesca         2       2019
214856      Deportivo La Coruna      Fuenlabrada         2       2019

                     fecha_dt  goles_local  goles_visitante  \
id_partido
214023      2019-08-17 17:00:00            1                3
214403      2019-08-17 18:00:00            0                1
214024      2019-08-17 19:00:00            1                1
214404      2019-08-17 19:00:00            3                0
214026      2019-08-17 21:00:00            4                4
…                           …          …              …
214853      2020-07-20 21:00:00            2                0
214863      2020-07-20 21:00:00            2                1
214854      2020-07-20 21:00:00            0                0
214862      2020-07-20 21:00:00            0                1
214856      2020-08-07 20:00:00            2                1

                      arbitro                              estadio  odd_1  \
id_partido
214023          Javier Estrada                   Abanca-Balaídos   4.75
214403        Aitor Gorostegui     Campos de Sport de El Sardinero   2.87
214024               Jesús Gil                 Estadio de Mestalla   1.66
214404                Saúl Ais  Estadio de los Juegos Mediterráneos   2.37
214026          Adrián Cordero              Estadio de la Cerámica   1.60
…                          …                                   …    …
214853             Juan Pulido              Estadio Santo Domingo   2.37
214863        Dámaso Arcediano              Estadio de la Romareda   2.10
214854                Saúl Ais  Estadio de los Juegos Mediterráneos   2.10
214862           Gorka Sagues         Estadio Municipal El Molinón   3.30
214856      Isidro Díaz de Mera              Estadio Abanca-Riazor   2.10

            odd_x  odd_2                                Informe_Tarjetas
id_partido
214023       4.20   1.65  Hubo 01 tajetas rojas al equipo visitante;Hubo…
214403       3.10   2.55  Hubo 03 amarillas mostradas al equipo local;Hu…
214024       3.75   5.50  Hubo 4 amarillas mostradas al equipo local;Hub…
214404       3.10   3.10  Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
214026       3.80   6.50  Hubo 01  tarjetas amarillas de jugadores visit…
…            …    …                                                   …
214853       2.87   3.40  Hubo 0 tajetas rojas al equipo visitante;Hubo …
214863       3.30   3.50  Hubo 00  tarjetas amarillas de jugadores visit…
214854       3.20   3.60  Hubo 2  tarjetas amarillas de jugadores visita…
214862       3.10   2.15  Hubo 2 amarillas para jugadores del equipo loc…
214856       3.20   3.60  Hubo 02 amarillas para jugadores del equipo lo…
```

```
[592 rows x 13 columns]
```

0.1.3 Pandas y csv: Escritura

Para escribir un CSV usamos el método `to_csv()`. Tienes el enlace a la documentación para ver más detalle.

[19]:
```
df.to_csv("./data/df_ejemplo_write.csv", sep ="|")
```

`sep`: indicar el separador que queramos usar, por defecto si no pones nada usará la ","
`index`: escribe la columna índice (True, por defecto) o no (False) [si escribes un dataframe con índice y este no tiene nombre luego aparecerá como Unamed cuando leas el fichero, usa el argumento `index_label` para ponerle nombre al índice en ese caso]

[21]:
```
with open("./data/df_ejemplo_write.csv", "r", encoding = "utf8") as f:
    datos = [linea.replace("\n","") for linea in f]
datos[0:12]
```

[21]:
```
['id_partido|equipo_local|equipo_visitante|Division|Temporada|fecha_dt|goles_loc
al|goles_visitante|arbitro|estadio|odd_1|odd_x|odd_2|Informe_Tarjetas',
 '214023|Celta Vigo|Real Madrid|1|2019|2019-08-17 17:00:00|1|3|Javier
Estrada|Abanca-Balaídos|4.75|4.2|1.65|Hubo 01 tajetas rojas al equipo
visitante;Hubo 0 rojas a jugadores del equipo local;Hubo 2 tarjetas amarillas
para el equipo visitantes;Hubo 6 amarillas para jugadores del equipo local',
 '214403|Racing Santander|Malaga|2|2019|2019-08-17 18:00:00|0|1|Aitor
Gorostegui|Campos de Sport de El Sardinero|2.87|3.1|2.55|Hubo 03 amarillas
mostradas al equipo local;Hubo 2  tarjetas amarillas de jugadores
visitantes;Hubo 00 rojas a jugadores visitantes;Hubo 1 rojas a jugadores del
equipo local',
 '214024|Valencia|Real Sociedad|1|2019|2019-08-17 19:00:00|1|1|Jesús Gil|Estadio
de Mestalla|1.66|3.75|5.5|Hubo 4 amarillas mostradas al equipo local;Hubo 1
tarjetas rojas sobre el equipo local;Hubo 4 tarjetas amarillas para el equipo
visitantes;Hubo 0 tajetas rojas al equipo visitante',
 '214404|Almeria|Albacete|2|2019|2019-08-17 19:00:00|3|0|Saúl Ais|Estadio de los
Juegos Mediterráneos|2.37|3.1|3.1|Hubo 00 rojas a jugadores visitantes;Hubo 01
amarillas mostradas al equipo local;Hubo 1  tarjetas amarillas de jugadores
visitantes;Hubo 0 tarjetas rojas sobre el equipo local',
 '214026|Villarreal|Granada CF|1|2019|2019-08-17 21:00:00|4|4|Adrián
Cordero|Estadio de la Cerámica|1.6|3.8|6.5|Hubo 01  tarjetas amarillas de
jugadores visitantes;Hubo 00 tajetas rojas al equipo visitante;Hubo 03 amarillas
para jugadores del equipo local;Hubo 00 rojas a jugadores del equipo local',
 '214025|Leganes|Osasuna|1|2019|2019-08-17 21:00:00|0|1|Javier Alberola|Estadio
Municipal de Butarque|2.0|3.2|4.2|Hubo 01 amarillas mostradas al equipo
local;Hubo 1 rojas a jugadores del equipo local;Hubo 4  tarjetas amarillas de
jugadores visitantes;Hubo 0 rojas a jugadores visitantes',
 '214406|Zaragoza|Tenerife|2|2019|2019-08-17 21:00:00|2|0|Dámaso
Arcediano|Estadio de la Romareda|2.0|3.3|3.8|Hubo 3 tarjetas amarillas para el
```

```
equipo visitantes;Hubo 01 amarillas para jugadores del equipo local;Hubo 0
tarjetas rojas sobre el equipo local;Hubo 1 tajetas rojas al equipo visitante',
 '214405|Rayo Vallecano|Mirandes|2|2019|2019-08-17 21:00:00|2|2|Juan
Pulido|Estadio de Vallecas|1.53|3.75|7.0|Hubo 1 rojas a jugadores
visitantes;Hubo 01 tarjetas rojas sobre el equipo local;Hubo 00 amarillas para
jugadores del equipo local;Hubo 4  tarjetas amarillas de jugadores visitantes',
 '214027|Alaves|Levante|1|2019|2019-08-18 17:00:00|1|0|César Soto|Estadio de
Mendizorroza|2.15|3.2|3.6|Hubo 01  tarjetas amarillas de jugadores
visitantes;Hubo 0 rojas a jugadores visitantes;Hubo 2 amarillas para jugadores
del equipo local;Hubo 0 tarjetas rojas sobre el equipo local',
 '214408|Numancia|Alcorcon|2|2019|2019-08-18 18:00:00|0|1|Alejandro Muñiz|Nuevo
Estadio Los Pajaritos|1.95|3.3|4.0|Hubo 0 tajetas rojas al equipo visitante;Hubo
2 amarillas mostradas al equipo local;Hubo 06 tarjetas amarillas para el equipo
visitantes;Hubo 00 rojas a jugadores del equipo local',
 '214407|Deportivo La Coruna|Oviedo|2|2019|2019-08-18 18:00:00|3|2|Daniel
Ocón|Estadio Abanca-Riazor|1.75|3.3|5.25|Hubo 00 tarjetas rojas sobre el equipo
local;Hubo 03  tarjetas amarillas de jugadores visitantes;Hubo 00 amarillas
mostradas al equipo local;Hubo 0 rojas a jugadores visitantes']
```

[23]:
```
df = pd.read_csv("./data/df_ejemplo_write.csv", sep = "|")
df
```

[23]:

	id_partido	equipo_local	equipo_visitante	Division	Temporada	\
0	214023	Celta Vigo	Real Madrid	1	2019	
1	214403	Racing Santander	Malaga	2	2019	
2	214024	Valencia	Real Sociedad	1	2019	
3	214404	Almeria	Albacete	2	2019	
4	214026	Villarreal	Granada CF	1	2019	
..	…	…	…	…	…	
587	214853	Alcorcon	Girona	2	2019	
588	214863	Zaragoza	Ponferradina	2	2019	
589	214854	Almeria	Malaga	2	2019	
590	214862	Sporting Gijon	Huesca	2	2019	
591	214856	Deportivo La Coruna	Fuenlabrada	2	2019	

	fecha_dt	goles_local	goles_visitante	arbitro	\
0	2019-08-17 17:00:00	1	3	Javier Estrada	
1	2019-08-17 18:00:00	0	1	Aitor Gorostegui	
2	2019-08-17 19:00:00	1	1	Jesús Gil	
3	2019-08-17 19:00:00	3	0	Saúl Ais	
4	2019-08-17 21:00:00	4	4	Adrián Cordero	
..	…	…	…	…	
587	2020-07-20 21:00:00	2	0	Juan Pulido	
588	2020-07-20 21:00:00	2	1	Dámaso Arcediano	
589	2020-07-20 21:00:00	0	0	Saúl Ais	
590	2020-07-20 21:00:00	0	1	Gorka Sagues	
591	2020-08-07 20:00:00	2	1	Isidro Díaz de Mera	

```
                                estadio  odd_1  odd_x  odd_2  \
0                        Abanca-Balaídos   4.75   4.20   1.65
1         Campos de Sport de El Sardinero   2.87   3.10   2.55
2                      Estadio de Mestalla   1.66   3.75   5.50
3    Estadio de los Juegos Mediterráneos   2.37   3.10   3.10
4                  Estadio de la Cerámica   1.60   3.80   6.50
..                                    …    …    …    …
587                Estadio Santo Domingo   2.37   2.87   3.40
588                Estadio de la Romareda   2.10   3.30   3.50
589  Estadio de los Juegos Mediterráneos   2.10   3.20   3.60
590         Estadio Municipal El Molinón   3.30   3.10   2.15
591                Estadio Abanca-Riazor   2.10   3.20   3.60

                                      Informe_Tarjetas
0    Hubo 01 tajetas rojas al equipo visitante;Hubo…
1    Hubo 03 amarillas mostradas al equipo local;Hu…
2    Hubo 4 amarillas mostradas al equipo local;Hub…
3    Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
4    Hubo 01  tarjetas amarillas de jugadores visit…
..                                                 …
587  Hubo 0 tajetas rojas al equipo visitante;Hubo …
588  Hubo 00  tarjetas amarillas de jugadores visit…
589  Hubo 2  tarjetas amarillas de jugadores visita…
590  Hubo 2 amarillas para jugadores del equipo loc…
591  Hubo 02 amarillas para jugadores del equipo lo…

[592 rows x 14 columns]
```

03_Ficheros_Excel

November 29, 2023

ETL Y DATOS

0.1 ACCESO LOCAL: Ficheros Excel

¿Qué empresa no trabaja con Excel? **Nos vamos a encontrar los formatos de datos de Excel en cualquier sitio**. Las extensiones de archivo más habituales son `.xslx` y `.xls`. Por suerte, **pandas tiene una función para leer los formatos de archivo de Excel y un método para escribirlos**.

El problema que presenta este tipo de lectura de datos es que **no es un formato tan cerrado como el CSV**. En el CSV tenemos una estructura compacta, con todos los datos separados por comas y con una línea de cabecera en la primera fila. El Excel permite tener datos en un formato mucho más flexible, con tablas en cualquier sitio de las hojas, información en varias hojas y demás.

Teniendo esto en cuenta, y sabiendo bien el formato del Excel en cuestión, podremos leerlo sin problemas con `pandas`, debido a la cantidad de argumentos que tiene la función `read_excel` que nos van a permitir adaptarnos hasta cierto punto a esa flexibilidad. En la documentación tienes todo el detalle.

0.1.1 Pandas y Excel: Lectura

El método para leer archivos excel tiene una lógica simiar al `read_csv` pero se llama `read_excel`. Leamos nuestro archivo de resultados de futbol de 2019, pero esta vez en formato excel:

[1]:
```
import pandas as pd
df= pd.read_excel("./data/df_liga_2019.xlsx")

df
```

[1]:

	id_partido	equipo_local	equipo_visitante	Division	Temporada
0	214023	Celta Vigo	Real Madrid	1	2019
1	214403	Racing Santander	Malaga	2	2019
2	214024	Valencia	Real Sociedad	1	2019
3	214404	Almeria	Albacete	2	2019
4	214026	Villarreal	Granada CF	1	2019
..	…	…	…	…	…
587	214853	Alcorcon	Girona	2	2019
588	214863	Zaragoza	Ponferradina	2	2019
589	214854	Almeria	Malaga	2	2019
590	214862	Sporting Gijon	Huesca	2	2019
591	214856	Deportivo La Coruna	Fuenlabrada	2	2019

	fecha_dt	goles_local	goles_visitante	arbitro
0	2019-08-17 17:00:00	1	3	Javier Estrada
1	2019-08-17 18:00:00	0	1	Aitor Gorostegui
2	2019-08-17 19:00:00	1	1	Jesús Gil
3	2019-08-17 19:00:00	3	0	Saúl Ais
4	2019-08-17 21:00:00	4	4	Adrián Cordero
..	…	…	…	…
587	2020-07-20 21:00:00	2	0	Juan Pulido
588	2020-07-20 21:00:00	2	1	Dámaso Arcediano
589	2020-07-20 21:00:00	0	0	Saúl Ais
590	2020-07-20 21:00:00	0	1	Gorka Sagues
591	2020-08-07 20:00:00	2	1	Isidro Díaz de Mera

	estadio	odd_1	odd_x	odd_2
0	Abanca-Balaídos	4.75	4.20	1.65
1	Campos de Sport de El Sardinero	2.87	3.10	2.55
2	Estadio de Mestalla	1.66	3.75	5.50
3	Estadio de los Juegos Mediterráneos	2.37	3.10	3.10
4	Estadio de la Cerámica	1.60	3.80	6.50
..	…	…	…	…
587	Estadio Santo Domingo	2.37	2.87	3.40
588	Estadio de la Romareda	2.10	3.30	3.50
589	Estadio de los Juegos Mediterráneos	2.10	3.20	3.60
590	Estadio Municipal El Molinón	3.30	3.10	2.15
591	Estadio Abanca-Riazor	2.10	3.20	3.60

	Informe_Tarjetas
0	Hubo 01 tajetas rojas al equipo visitante;Hubo…
1	Hubo 03 amarillas mostradas al equipo local;Hu…
2	Hubo 4 amarillas mostradas al equipo local;Hub…
3	Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
4	Hubo 01 tarjetas amarillas de jugadores visit…
..	…
587	Hubo 0 tajetas rojas al equipo visitante;Hubo …

```
588  Hubo 00  tarjetas amarillas de jugadores visit…
589  Hubo 2  tarjetas amarillas de jugadores visita…
590  Hubo 2 amarillas para jugadores del equipo loc…
591  Hubo 02 amarillas para jugadores del equipo lo…

[592 rows x 14 columns]
```

Al igual que con `read_csv`, vamos a ver algunos de los argumentos más interesantes del método y que nos ayudarán cuando la hoja no esté tan ordenadita como la que acabamos de leer:

sheet_name Dado que un "libro" excel (yo siempre lo he llamado hoja, pero puede llevar a confusión) puede tener varias hojas, este argumento permite indicar las hojas queremos leer usando su nombre o posición en el "libro". Si no se pone nada lee la primera según esté ordenado el fichero excel.

[3]:
```
df = pd.read_excel("./data/df_liga_2019.xlsx", sheet_name = "futbol_2")

df
```

[3]:

	id_fixture	home_team	away_team	Division	Season	date_dt	goals_home
0	'goals_away', 'refer…	Celta Vigo	Real Madrid	1	2019	2019-08-17 17:00:00	1
1	214403	Racing Santander	Malaga	2	2019	2019-08-17 18:00:00	0
2	214024	Valencia	Real Sociedad	1	2019	2019-08-17 19:00:00	1
3	214404	Almeria	Albacete	2	2019	2019-08-17 19:00:00	3
4	214026	Villarreal	Granada CF	1	2019	2019-08-17 21:00:00	4
5	214025	Leganes	Osasuna	1	2019	2019-08-17 21:00:00	0
6	214406	Zaragoza	Tenerife	2	2019	2019-08-17 21:00:00	2
7	214405	Rayo Vallecano	Mirandes	2	2019	2019-08-17 21:00:00	2
8	214027	Alaves	Levante	1	2019	2019-08-18 17:00:00	1
9	214408	Numancia	Alcorcon	2	2019	2019-08-18 18:00:00	0
10	214407	Deportivo La Coruna					
11	214411	Girona					
12	214028	Espanyol					
13	214410	Las Palmas					
14	214029	Atletico Madrid					
15	214034	Osasuna					

```
10          Oviedo          2     2019  2019-08-18 18:00:00           3
11  Sporting Gijon          2     2019  2019-08-18 18:30:00           1
12         Sevilla          1     2019  2019-08-18 19:00:00           0
13          Huesca          2     2019  2019-08-18 20:00:00           0
14          Getafe          1     2019  2019-08-18 22:00:00           1
15           Eibar          1     2019  2019-08-24 17:00:00           0
```

[4]:
```
#leer la hoja por posiciion
df =pd.read_excel("./data/df_liga_2019.xlsx", sheet_name = 0)
df
```

[4]:
```
     id_partido         equipo_local equipo_visitante  Division  Temporada  \
0        214023           Celta Vigo      Real Madrid         1       2019
1        214403     Racing Santander           Malaga         2       2019
2        214024              Valencia    Real Sociedad         1       2019
3        214404               Almeria         Albacete         2       2019
4        214026            Villarreal       Granada CF         1       2019
..          …                    …               …       …         …
587      214853              Alcorcon           Girona         2       2019
588      214863              Zaragoza     Ponferradina         2       2019
589      214854               Almeria           Malaga         2       2019
590      214862       Sporting Gijon           Huesca         2       2019
591      214856  Deportivo La Coruna      Fuenlabrada         2       2019

                fecha_dt  goles_local  goles_visitante              arbitro  \
0    2019-08-17 17:00:00            1                3       Javier Estrada
1    2019-08-17 18:00:00            0                1     Aitor Gorostegui
2    2019-08-17 19:00:00            1                1            Jesús Gil
3    2019-08-17 19:00:00            3                0             Saúl Ais
4    2019-08-17 21:00:00            4                4       Adrián Cordero
..                   …          …              …                    …
587  2020-07-20 21:00:00            2                0          Juan Pulido
588  2020-07-20 21:00:00            2                1     Dámaso Arcediano
589  2020-07-20 21:00:00            0                0             Saúl Ais
590  2020-07-20 21:00:00            0                1         Gorka Sagues
591  2020-08-07 20:00:00            2                1  Isidro Díaz de Mera

                                estadio  odd_1  odd_x  odd_2  \
0                        Abanca-Balaídos   4.75   4.20   1.65
1          Campos de Sport de El Sardinero   2.87   3.10   2.55
2                     Estadio de Mestalla   1.66   3.75   5.50
3    Estadio de los Juegos Mediterráneos   2.37   3.10   3.10
4                  Estadio de la Cerámica   1.60   3.80   6.50
..                                    …    …    …    …
587                Estadio Santo Domingo   2.37   2.87   3.40
588                Estadio de la Romareda   2.10   3.30   3.50
589  Estadio de los Juegos Mediterráneos   2.10   3.20   3.60
```

```
590         Estadio Municipal El Molinón   3.30   3.10   2.15
591                Estadio Abanca-Riazor   2.10   3.20   3.60

                                      Informe_Tarjetas
0    Hubo 01 tajetas rojas al equipo visitante;Hubo…
1    Hubo 03 amarillas mostradas al equipo local;Hu…
2    Hubo 4 amarillas mostradas al equipo local;Hub…
3    Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
4    Hubo 01  tarjetas amarillas de jugadores visit…
..                                                 …
587  Hubo 0 tajetas rojas al equipo visitante;Hubo …
588  Hubo 00  tarjetas amarillas de jugadores visit…
589  Hubo 2  tarjetas amarillas de jugadores visita…
590  Hubo 2 amarillas para jugadores del equipo loc…
591  Hubo 02 amarillas para jugadores del equipo lo…

[592 rows x 14 columns]
```

Puedes indicarle una lista con las posiciones o los nombres de varias hojas, e incluso `None` si quieres cargar todas. En estos casos la función devuelve un diccionario. Prueba con alguna de tus hojas excel en local o con la que usamos aquí

index_col Tiene el mismo uso que en `read_csv`, sirve para indicar la columna que funciona de índice (si lees varias hojas a la vez puedes pasarle una lista con los índices para cada una pero eso no lo veremos en esta sesión)

[5]:
```
df = pd.read_excel("./data/df_liga_2019.xlsx", index_col= "id_partido")
df
```

[5]:
```
                  equipo_local equipo_visitante  Division  Temporada  \
id_partido
214023               Celta Vigo      Real Madrid         1       2019
214403         Racing Santander           Malaga         2       2019
214024                 Valencia    Real Sociedad         1       2019
214404                  Almeria         Albacete         2       2019
214026                Villarreal       Granada CF         1       2019
…                            …                …       …         …
214853                 Alcorcon           Girona         2       2019
214863                 Zaragoza     Ponferradina         2       2019
214854                  Almeria           Malaga         2       2019
214862           Sporting Gijon           Huesca         2       2019
214856      Deportivo La Coruna      Fuenlabrada         2       2019

                        fecha_dt  goles_local  goles_visitante  \
id_partido
214023      2019-08-17 17:00:00            1                3
214403      2019-08-17 18:00:00            0                1
214024      2019-08-17 19:00:00            1                1
```

```
214404      2019-08-17 19:00:00            3                0
214026      2019-08-17 21:00:00            4                4
…                       …           …                …
214853      2020-07-20 21:00:00            2                0
214863      2020-07-20 21:00:00            2                1
214854      2020-07-20 21:00:00            0                0
214862      2020-07-20 21:00:00            0                1
214856      2020-08-07 20:00:00            2                1

                        arbitro                                estadio  odd_1  \
id_partido
214023           Javier Estrada                     Abanca-Balaídos   4.75
214403         Aitor Gorostegui     Campos de Sport de El Sardinero   2.87
214024                Jesús Gil                  Estadio de Mestalla   1.66
214404                Saúl Ais  Estadio de los Juegos Mediterráneos   2.37
214026           Adrián Cordero               Estadio de la Cerámica   1.60
…                           …                                   …    …
214853              Juan Pulido                Estadio Santo Domingo   2.37
214863         Dámaso Arcediano               Estadio de la Romareda   2.10
214854                Saúl Ais  Estadio de los Juegos Mediterráneos   2.10
214862             Gorka Sagues        Estadio Municipal El Molinón   3.30
214856      Isidro Díaz de Mera               Estadio Abanca-Riazor   2.10

            odd_x  odd_2                                Informe_Tarjetas
id_partido
214023       4.20   1.65  Hubo 01 tajetas rojas al equipo visitante;Hubo…
214403       3.10   2.55  Hubo 03 amarillas mostradas al equipo local;Hu…
214024       3.75   5.50  Hubo 4 amarillas mostradas al equipo local;Hub…
214404       3.10   3.10  Hubo 00 rojas a jugadores visitantes;Hubo 01 a…
214026       3.80   6.50  Hubo 01  tarjetas amarillas de jugadores visit…
…             …    …                                                  …
214853       2.87   3.40  Hubo 0 tajetas rojas al equipo visitante;Hubo …
214863       3.30   3.50  Hubo 00  tarjetas amarillas de jugadores visit…
214854       3.20   3.60  Hubo 2  tarjetas amarillas de jugadores visita…
214862       3.10   2.15  Hubo 2 amarillas para jugadores del equipo loc…
214856       3.20   3.60  Hubo 02 amarillas para jugadores del equipo lo…

[592 rows x 13 columns]
```

usecols y skiprows Si los datos no empiezan en la columna A y en la fila 1, leeremos mal estos si hacemos uso de la función tal y como hasta ahora:

[6]:
```
df = pd.read_excel("./data/df_liga_2019.xlsx", sheet_name="futbol_3")

df
```

[6]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	id_partido	equipo_local	equipo_visitante
3	NaN	NaN	214023	Celta Vigo	Real Madrid
4	NaN	NaN	214403	Racing Santander	Malaga
5	NaN	NaN	214024	Valencia	Real Sociedad
6	NaN	NaN	214404	Almeria	Albacete
7	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	214026	Villarreal	Granada CF
9	NaN	NaN	214025	Leganes	Osasuna
10	NaN	NaN	214406	Zaragoza	Tenerife
11	NaN	NaN	214405	Rayo Vallecano	Mirandes
12	NaN	NaN	214027	Alaves	Levante
13	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	214408	Numancia	Alcorcon
15	NaN	NaN	214407	Deportivo La Coruna	Oviedo
16	NaN	NaN	214411	Girona	Sporting Gijon
17	NaN	NaN	214028	Espanyol	Sevilla
18	NaN	NaN	214410	Las Palmas	Huesca
19	NaN	NaN	214029	Atletico Madrid	Getafe
20	NaN	NaN	214034	Osasuna	Eibar

	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	Division	Temporada	fecha_dt	goles_local	goles_visitante
3	1	2019	2019-08-17 17:00:00	1	3
4	2	2019	2019-08-17 18:00:00	0	1
5	1	2019	2019-08-17 19:00:00	1	1
6	2	2019	2019-08-17 19:00:00	3	0
7	NaN	NaN	NaN	NaN	NaN
8	1	2019	2019-08-17 21:00:00	4	4
9	1	2019	2019-08-17 21:00:00	0	1
10	2	2019	2019-08-17 21:00:00	2	0
11	2	2019	2019-08-17 21:00:00	2	2
12	1	2019	2019-08-18 17:00:00	1	0
13	NaN	NaN	NaN	NaN	NaN
14	2	2019	2019-08-18 18:00:00	0	1
15	2	2019	2019-08-18 18:00:00	3	2
16	2	2019	2019-08-18 18:30:00	1	1
17	1	2019	2019-08-18 19:00:00	0	2
18	2	2019	2019-08-18 20:00:00	0	1
19	1	2019	2019-08-18 22:00:00	1	0
20	1	2019	2019-08-24 17:00:00	0	0

	Unnamed: 10	Unnamed: 11	Unnamed: 12

0	NaN		NaN	NaN
1	NaN		NaN	NaN
2	arbitro		estadio	odd_1
3	Javier Estrada		Abanca-Balaídos	4.75
4	Aitor Gorostegui	Campos de Sport de El Sardinero		2.87
5	Jesús Gil		Estadio de Mestalla	1.66
6	Saúl Ais	Estadio de los Juegos Mediterráneos		2.37
7	NaN		NaN	NaN
8	Adrián Cordero		Estadio de la Cerámica	1.6
9	Javier Alberola	Estadio Municipal de Butarque		2
10	Dámaso Arcediano		Estadio de la Romareda	2
11	Juan Pulido		Estadio de Vallecas	1.53
12	César Soto		Estadio de Mendizorroza	2.15
13	NaN		NaN	NaN
14	Alejandro Muñiz	Nuevo Estadio Los Pajaritos		1.95
15	Daniel Ocón		Estadio Abanca-Riazor	1.75
16	Daniel Trujillo	Estadi Municipal de Montilivi		2.2
17	Juan Martínez		RCDE Stadium	3.2
18	Jorge Figueroa		Estadio de Gran Canaria	2.25
19	Guillermo Cuadra	Estadio Wanda Metropolitano		1.44
20	David Medié		Estadio El Sadar	2.5

	Unnamed: 13
0	NaN
1	NaN
2	odd_x
3	4.2
4	3.1
5	3.75
6	3.1
7	NaN
8	3.8
9	3.2
10	3.3
11	3.75
12	3.2
13	NaN
14	3.3
15	3.3
16	3.1
17	3.3
18	3.25
19	4.33
20	3.2

Si haces unos cálculos, verás que nuestros datos realmente van de la columna "C" a la "N" y, si te fijas además empiezan en la cuarta fila y que las filas 8 y 14 (ojo que en este caso empezamos en 1 a contar) no son buenas. La función nos permite tener en cuenta esto con los parámetros `usecols`

con el que le diremos las columnas que queremos usar y `skiprows` que nos permite decir que líneas no incluir.

[11]:
```
df= pd.read_excel("./data/df_liga_2019.xlsx", sheet_name = "futbol_3", usecol=␣
 ↪"C:N", skiprows=[0,1,2,8,14])
df
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING␣
 ↪5\Acceso_local_y_Tratamiento_de_Datos\unit␣
 ↪2_Acceso_y_Procesado_de_Datos_Internos__Ficheros\03_Ficheros_Excel.ipynb Celd␣
 ↪19 line 1
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/03_Ficheros_Excel.
 ↪ipynb#X24sZmlsZQ%3D%3D?line=0'>1</a> df= pd.read_excel("./data/df_liga_2019.
 ↪xlsx", sheet_name = "futbol_3", usecol= "C:N", skiprows=[0,1,2,8,14])
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/03_Ficheros_Excel.
 ↪ipynb#X24sZmlsZQ%3D%3D?line=1'>2</a> df

TypeError: read_excel() got an unexpected keyword argument 'usecol'
```

0.1.2 Pandas y excel: Escritura

Al igual que con el CSV, tenemos el método `to_excel()`, para escribir el `DataFame` en un archivo Excel.

Recuerda poner la extensión del Excel (.xlsx) en el nombre del archivo. Tienes el enlace a la documentación para ver más detalle.

Aquí no usaremos el parámetro sep, pero sí el index si queremos que incluya o no el índice, con las mismas consideraciones que para `read_csv` (index está a True por defecto, si el `DataFrame` no tiene un índice explícito se creará una columna en la excel llamada Unname:0 con los valores del índice implícito)

[10]:
```
df.to_excel("./data/df_ejemplo_excel.xlsx", sheet_name = "Test")
df_2 = pd.read_excel("./data/df_ejemplo_excel.xlsx")
df_2
```

[10]:

	Unnamed: 0.1	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3
0	0	NaN	NaN	NaN	NaN
1	1	NaN	NaN	NaN	NaN
2	2	NaN	NaN	id_partido	equipo_local
3	3	NaN	NaN	214023	Celta Vigo
4	4	NaN	NaN	214403	Racing Santander
5	5	NaN	NaN	214024	Valencia
6	6	NaN	NaN	214404	Almeria

```
7              7         NaN         NaN         NaN                  NaN
8              8         NaN         NaN      214026            Villarreal
9              9         NaN         NaN      214025               Leganes
10            10         NaN         NaN      214406              Zaragoza
11            11         NaN         NaN      214405        Rayo Vallecano
12            12         NaN         NaN      214027                Alaves
13            13         NaN         NaN         NaN                  NaN
14            14         NaN         NaN      214408              Numancia
15            15         NaN         NaN      214407   Deportivo La Coruna
16            16         NaN         NaN      214411                Girona
17            17         NaN         NaN      214028              Espanyol
18            18         NaN         NaN      214410           Las Palmas
19            19         NaN         NaN      214029      Atletico Madrid
20            20         NaN         NaN      214034               Osasuna

         Unnamed: 4 Unnamed: 5 Unnamed: 6           Unnamed: 7   Unnamed: 8  \
0               NaN        NaN        NaN                  NaN          NaN
1               NaN        NaN        NaN                  NaN          NaN
2   equipo_visitante   Division  Temporada            fecha_dt  goles_local
3        Real Madrid          1       2019  2019-08-17 17:00:00            1
4             Malaga          2       2019  2019-08-17 18:00:00            0
5      Real Sociedad          1       2019  2019-08-17 19:00:00            1
6           Albacete          2       2019  2019-08-17 19:00:00            3
7               NaN        NaN        NaN                  NaN          NaN
8         Granada CF          1       2019  2019-08-17 21:00:00            4
9            Osasuna          1       2019  2019-08-17 21:00:00            0
10          Tenerife          2       2019  2019-08-17 21:00:00            2
11          Mirandes          2       2019  2019-08-17 21:00:00            2
12           Levante          1       2019  2019-08-18 17:00:00            1
13              NaN        NaN        NaN                  NaN          NaN
14          Alcorcon          2       2019  2019-08-18 18:00:00            0
15            Oviedo          2       2019  2019-08-18 18:00:00            3
16    Sporting Gijon          2       2019  2019-08-18 18:30:00            1
17           Sevilla          1       2019  2019-08-18 19:00:00            0
18            Huesca          2       2019  2019-08-18 20:00:00            0
19            Getafe          1       2019  2019-08-18 22:00:00            1
20             Eibar          1       2019  2019-08-24 17:00:00            0

        Unnamed: 9       Unnamed: 10                          Unnamed: 11  \
0              NaN               NaN                                  NaN
1              NaN               NaN                                  NaN
2   goles_visitante           arbitro                              estadio
3                 3    Javier Estrada                       Abanca-Balaídos
4                 1  Aitor Gorostegui      Campos de Sport de El Sardinero
5                 1         Jesús Gil                   Estadio de Mestalla
6                 0          Saúl Ais  Estadio de los Juegos Mediterráneos
7               NaN               NaN                                  NaN
```

8	4	Adrián Cordero	Estadio de la Cerámica
9	1	Javier Alberola	Estadio Municipal de Butarque
10	0	Dámaso Arcediano	Estadio de la Romareda
11	2	Juan Pulido	Estadio de Vallecas
12	0	César Soto	Estadio de Mendizorroza
13	NaN	NaN	NaN
14	1	Alejandro Muñiz	Nuevo Estadio Los Pajaritos
15	2	Daniel Ocón	Estadio Abanca-Riazor
16	1	Daniel Trujillo	Estadi Municipal de Montilivi
17	2	Juan Martínez	RCDE Stadium
18	1	Jorge Figueroa	Estadio de Gran Canaria
19	0	Guillermo Cuadra	Estadio Wanda Metropolitano
20	0	David Medié	Estadio El Sadar

	Unnamed: 12	Unnamed: 13
0	NaN	NaN
1	NaN	NaN
2	odd_1	odd_x
3	4.75	4.2
4	2.87	3.1
5	1.66	3.75
6	2.37	3.1
7	NaN	NaN
8	1.6	3.8
9	2	3.2
10	2	3.3
11	1.53	3.75
12	2.15	3.2
13	NaN	NaN
14	1.95	3.3
15	1.75	3.3
16	2.2	3.1
17	3.2	3.3
18	2.25	3.25
19	1.44	4.33
20	2.5	3.2

Si además quieres saber más sobre como manejar hojas excel desde Python, te recomiendo que visites la documentacion de openpyxl

04_Ficheros_json_I

November 29, 2023

ETL Y DATOS

vamos a empezar otros tipo de texto plano con una extrutura menos tabular que excel o csv, y texto pero tiene mucho uso en programacion y datos , no encontraremos grandes cantidades de datos en este formato, si lo recuperaremos al hablar de bases de datos NoSQL y lo usaremos en setdatos como parametros de configuracion o ficheros similares

0.1 ACCESO LOCAL: JSON (I)

Los fichero Json, o *JavaScript Objet Notation* es otro formato de texto plano que se utiliza para el itercambio de datos. Originalmente se utilizaba como notación literal de objetos en JavaScript, pero actualmente es un formato de datos independiente del lenguaje. JavaScript es un lenguaje de programación web, por lo que JSON se utiliza mucho en el intercambio de objetos entre cliente y servidor.

Y su extensión (o la extensión de los ficheros Json) es... tachán!!: `.json`

¿Qué diferencia hay con un CSV o un Excel? Ya no tenemos esa estructura de fila/columna, sino que ahora es un formato tipo clave/valor, como si fuese un diccionario. En una tabla en la fila 1, columna 1, tienes un valor. En un JSON no, en la clave "mi_calve" puedes tener almacenado un valor, una lista o incluso un objeto. Salimos del formato tabla al que estamos acostubrados para ganar en flexibilidad. Es como un fichero para guardar diccionarios o listas de ellos.

0.1.1 Datos en Json

El contenido de un archivo JSON tiene la siguiente pinta:

[1]:
```
### Diccionario único
with open("./data/single_json.json","r") as f:
    print(f.read())
```

```
{
        "firstName": "Jane",
        "lastName": "Doe",
        "hobbies": ["running", "sky diving", "singing"],
        "age": 35,
        "children": [
            {
                "firstName": "Alice",
                "age": 6
            },
            {
                "firstName": "Bob",
                "age": 8
            }
        ]
    }
```

Sí, pues eso un diccionario, ojo pero también tiene esta pinta:

[2]:
```
# Una lista de diccionarios
with open("./data/presidentes_short.json", "r") as f:
    for linea in f:
        print(linea, end ="")
```

```
[
  {
    "id": 1,
    "president": 1,
    "nm": "George Washington",
    "pp": "None, Federalist",
    "tm": "1789-1797"
  },
  {
    "id": 2,
    "president": 2,
    "nm": "John Adams",
    "pp": "Federalist",
    "tm": "1797-1801"
  },
  {
    "id": 3,
    "president": 3,
    "nm": "Thomas Jefferson",
    "pp": "Democratic-Republican",
    "tm": "1801-1809"
  }
]
```

O esta otra:

[3]:
```
# Un diccionario por línea
with open("./data/Musical_short.json","r") as f:
    for linea in f:
        print(linea, end ="")
```

{"reviewerID": "A2IBPI20UZIR0U", "asin": "1384719342", "reviewerName": "cassandra tu \"Yeah, well, that's just like, u…", "helpful": [0, 0], "reviewText": "Not much to write about here, but it does exactly what it's supposed to. filters out the pop sounds. now my recordings are much more crisp. it is one of the lowest prices pop filters on amazon so might as well buy it, they honestly work the same despite their pricing,", "overall": 5.0, "summary": "good", "unixReviewTime": 1393545600, "reviewTime": "02 28, 2014"}
{"reviewerID": "A14VAT5EAX3D9S", "asin": "1384719342", "reviewerName": "Jake", "helpful": [13, 14], "reviewText": "The product does exactly as it should and is quite affordable.I did not realized it was double screened until it arrived, so it was even better than I had expected.As an added bonus, one of the screens carries a small hint of the smell of an old grape candy I used to buy, so for reminiscent's sake, I cannot stop putting the pop filter next to my nose and smelling it after recording. :DIf you needed a pop filter, this will work just as well as the expensive ones, and it may even come with a pleasing aroma like mine did!Buy this product! :]", "overall": 5.0, "summary": "Jake", "unixReviewTime": 1363392000, "reviewTime": "03 16, 2013"}
{"reviewerID": "A195EZSQDW3E21", "asin": "1384719342", "reviewerName": "Rick Bennette \"Rick Bennette\"", "helpful": [1, 1], "reviewText": "The primary job of this device is to block the breath that would otherwise produce a popping sound, while allowing your voice to pass through with no noticeable reduction of volume or high frequencies. The double cloth filter blocks the pops and lets the voice through with no coloration. The metal clamp mount attaches to the mike stand secure enough to keep it attached. The goose neck needs a little coaxing to stay where you put it.", "overall": 5.0, "summary": "It Does The Job Well", "unixReviewTime": 1377648000, "reviewTime": "08 28, 2013"}
{"reviewerID": "A2C00NNG1ZQQG2", "asin": "1384719342", "reviewerName": "RustyBill \"Sunday Rocker\"", "helpful": [0, 0], "reviewText": "Nice windscreen protects my MXL mic and prevents pops. Only thing is that the gooseneck is only marginally able to hold the screen in position and requires careful positioning of the clamp to avoid sagging.", "overall": 5.0, "summary": "GOOD WINDSCREEN FOR THE MONEY", "unixReviewTime": 1392336000, "reviewTime": "02 14, 2014"}
{"reviewerID": "A94QU4C90B1AX", "asin": "1384719342", "reviewerName": "SEAN MASLANKA", "helpful": [0, 0], "reviewText": "This pop filter is great. It looks and performs like a studio filter. If you're recording vocals this will eliminate the pops that gets recorded when you sing.", "overall": 5.0, "summary": "No more pops when I record my vocals.", "unixReviewTime": 1392940800, "reviewTime": "02 21, 2014"}
{"reviewerID": "A2A039TZMZHH9Y", "asin": "B00004Y2UT", "reviewerName": "Bill Lewey \"blewey\"", "helpful": [0, 0], "reviewText": "So good that I bought another one. Love the heavy cord and gold connectors. Bass sounds great. I just learned last night how to coil them up. I guess I should read instructions more carefully. But no harm done, still works great!", "overall": 5.0,

"summary": "The Best Cable", "unixReviewTime": 1356048000, "reviewTime": "12 21, 2012"}
{"reviewerID": "A1UPZM995ZAH90", "asin": "B00004Y2UT", "reviewerName": "Brian", "helpful": [0, 0], "reviewText": "I have used monster cables for years, and with good reason. The lifetime warranty is worth the price alone. Simple fact: cables break, but getting to replace them at no cost is where it's at.", "overall": 5.0, "summary": "Monster Standard 100 - 21' Instrument Cable", "unixReviewTime": 1390089600, "reviewTime": "01 19, 2014"}
{"reviewerID": "AJNFQI3YR6XJ5", "asin": "B00004Y2UT", "reviewerName": "Fender Guy \"Rick\"", "helpful": [0, 0], "reviewText": "I now use this cable to run from the output of my pedal chain to the input of my Fender Amp. After I bought Monster Cable to hook up my pedal board I thought I would try another one and update my guitar. I had been using a high end Planet Waves cable that I bought in the 1980's… Once I found out the input jacks on the new Monster cable didn't fit into the Fender Strat jack I was a little disappointed… I didn't return it and as stated I use it for the output on the pedal board. Save your money… I went back to my Planet Waves Cable…I payed $30.00 back in the eighties for the Planet Waves which now comes in at around $50.00. What I'm getting at is you get what you pay for. I thought Waves was a lot of money back in the day…but I haven't bought a guitar cable since this one…20 plus years and still working…Planet Waves wins.", "overall": 3.0, "summary": "Didn't fit my 1996 Fender Strat…", "unixReviewTime": 1353024000, "reviewTime": "11 16, 2012"}
{"reviewerID": "A3M1PLEYNDEYO8", "asin": "B00004Y2UT", "reviewerName": "G. Thomas \"Tom\"", "helpful": [0, 0], "reviewText": "Perfect for my Epiphone Sheraton II. Monster cables are well constructed. I have several and never had any problems with any of them over the years. Got this one because I wanted the 90 degree plug.", "overall": 5.0, "summary": "Great cable", "unixReviewTime": 1215302400, "reviewTime": "07 6, 2008"}
{"reviewerID": "AMNTZU1YQN1TH", "asin": "B00004Y2UT", "reviewerName": "Kurt Robair", "helpful": [0, 0], "reviewText": "Monster makes the best cables and a lifetime warranty doesnt hurt either. This isnt their top of the line series but it works great with my bass guitar rig and has for some time. You cant go wrong with Monster Cables.", "overall": 5.0, "summary": "Best Instrument Cables On The Market", "unixReviewTime": 1389139200, "reviewTime": "01 8, 2014"}

0.1.2 Lectura y extracción de datos de archivos Json

Con calma, los archivos json se leen como hemos hecho en los ejemplos anteriores (como un texto plano, con `open`), pero luego NO se procesan igual que un fichero de texto. Ha sido sólo para mostrarte el contenido ya que son archivos de texto plano.

Existe una librería de Python específica para "leer" el contenido de los `json` con el original nombre de.... premio: `json` (sencillo, directo y sin confusiones)

Haremos uso de ella cuando queramos extraer el contenido de un archivo .json (y sepamos que no es directamente convertible a pandas, que eso lo vemos en la siguiente sesión de Json)

[2]:
```
import json
```

Y ahora nos vamos a leer los tres archivos anteriores y ver que nos proporciona esa lectura:

[5]:
```
with open("./data/single_json.json","r") as f: # Esto igual que para leer texto
    datos = json.load(f) # esto es lo que cambia
print(type(datos))
datos
```

```
<class 'dict'>
```

[5]:
```
{'firstName': 'Jane',
 'lastName': 'Doe',
 'hobbies': ['running', 'sky diving', 'singing'],
 'age': 35,
 'children': [{'firstName': 'Alice', 'age': 6},
  {'firstName': 'Bob', 'age': 8}]}
```

[ ]:
```
#Lo esperado, un diccionario. Vamos con los presidentes:
```

[6]:
```
with open("./data/presidentes_short.json","r") as f:
    datos = json.load(f)
print(type(datos))
datos
```

```
<class 'list'>
```

[6]:
```
[{'id': 1,
  'president': 1,
  'nm': 'George Washington',
  'pp': 'None, Federalist',
  'tm': '1789-1797'},
 {'id': 2,
  'president': 2,
  'nm': 'John Adams',
  'pp': 'Federalist',
  'tm': '1797-1801'},
 {'id': 3,
  'president': 3,
  'nm': 'Thomas Jefferson',
  'pp': 'Democratic-Republican',
  'tm': '1801-1809'}]
```

Pues también una lista de dicts, pero eso sí ya como una lista, no tenemos que leer nosotros el texto y hacernos algo para construir la lista.

Por último veamos que ocurre con las críticas de instrumentos musicales:

[3]:
```
with open("./data/Musical_short.json","r") as f:
    datos  = json.load(f)
print(type(datos))
datos
```

```
---------------------------------------------------------------------------
JSONDecodeError                           Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING␣
 ↪5\Acceso_local_y_Tratamiento_de_Datos\unit␣
 ↪2_Acceso_y_Procesado_de_Datos_Internos__Ficheros\04_Ficheros_json_I.ipynb␣
 ↪Celda 21 line 2
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/04_Ficheros_json_I.
 ↪ipynb#X26sZmlsZQ%3D%3D?line=0'>1</a> with open("./data/Musical_short.
 ↪json","r") as f:
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/04_Ficheros_json_I.
 ↪ipynb#X26sZmlsZQ%3D%3D?line=1'>2</a>     datos  = json.load(f)
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/04_Ficheros_json_I.
 ↪ipynb#X26sZmlsZQ%3D%3D?line=2'>3</a> print(type(datos))
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/04_Ficheros_json_I.
 ↪ipynb#X26sZmlsZQ%3D%3D?line=3'>4</a> datos

File c:\Users\victo\AppData\Local\Programs\Python\Python310\lib\json\__init__.p
 ↪293, in load(fp, cls, object_hook, parse_float, parse_int, parse_constant,␣
 ↪object_pairs_hook, **kw)
    274 def load(fp, *, cls=None, object_hook=None, parse_float=None,
    275         parse_int=None, parse_constant=None, object_pairs_hook=None,␣
 ↪**kw):
    276     """Deserialize ``fp`` (a ``.read()``-supporting file-like object␣
 ↪containing
    277     a JSON document) to a Python object.
    278
   (…)
    291     kwarg; otherwise ``JSONDecoder`` is used.
    292     """
--> 293     return loads(fp.read(),
    294         cls=cls, object_hook=object_hook,
    295         parse_float=parse_float, parse_int=parse_int,
    296         parse_constant=parse_constant,␣
 ↪object_pairs_hook=object_pairs_hook, **kw)

File c:\Users\victo\AppData\Local\Programs\Python\Python310\lib\json\__init__.p
 ↪346, in loads(s, cls, object_hook, parse_float, parse_int, parse_constant,␣
 ↪object_pairs_hook, **kw)
    341     s = s.decode(detect_encoding(s), 'surrogatepass')
    343 if (cls is None and object_hook is None and
    344         parse_int is None and parse_float is None and
    345         parse_constant is None and object_pairs_hook is None and not kw :
```

```
--> 346     return _default_decoder.decode(s)
    347 if cls is None:
    348     cls = JSONDecoder

File c:\Users\victo\AppData\Local\Programs\Python\Python310\lib\json\decoder.py
 ↪340, in JSONDecoder.decode(self, s, _w)
    338 end = _w(s, end).end()
    339 if end != len(s):
--> 340     raise JSONDecodeError("Extra data", s, end)
    341 return obj

JSONDecodeError: Extra data: line 2 column 1 (char 523)
```

Pues que en realidad ese está preparado para leerlo con la función de lectura de pandas de json. Si lo queremos leer nosotros tendríamos que hacer algo así:

[8]:
```
with open("./data/Musical_short.json","r") as f:
    datos = [json.loads(linea) for linea in f] # Fijate que tiene una "s",␣
 ↪porque aquí no estamos leyendo de un archivo sino de un string
print(type(datos))
datos
```

```
<class 'list'>
```

[8]:
```
[{'reviewerID': 'A2IBPI20UZIR0U',
  'asin': '1384719342',
  'reviewerName': 'cassandra tu "Yeah, well, that\'s just like, u…',
  'helpful': [0, 0],
  'reviewText': "Not much to write about here, but it does exactly what it's
supposed to. filters out the pop sounds. now my recordings are much more crisp.
it is one of the lowest prices pop filters on amazon so might as well buy it,
they honestly work the same despite their pricing,",
  'overall': 5.0,
  'summary': 'good',
  'unixReviewTime': 1393545600,
  'reviewTime': '02 28, 2014'},
 {'reviewerID': 'A14VAT5EAX3D9S',
  'asin': '1384719342',
  'reviewerName': 'Jake',
  'helpful': [13, 14],
  'reviewText': "The product does exactly as it should and is quite affordable.I
did not realized it was double screened until it arrived, so it was even better
than I had expected.As an added bonus, one of the screens carries a small hint
of the smell of an old grape candy I used to buy, so for reminiscent's sake, I
cannot stop putting the pop filter next to my nose and smelling it after
recording. :DIf you needed a pop filter, this will work just as well as the
expensive ones, and it may even come with a pleasing aroma like mine did!Buy
```

this product! :]",
  'overall': 5.0,
  'summary': 'Jake',
  'unixReviewTime': 1363392000,
  'reviewTime': '03 16, 2013'},
 {'reviewerID': 'A195EZSQDW3E21',
  'asin': '1384719342',
  'reviewerName': 'Rick Bennette "Rick Bennette"',
  'helpful': [1, 1],
  'reviewText': 'The primary job of this device is to block the breath that would otherwise produce a popping sound, while allowing your voice to pass through with no noticeable reduction of volume or high frequencies. The double cloth filter blocks the pops and lets the voice through with no coloration. The metal clamp mount attaches to the mike stand secure enough to keep it attached. The goose neck needs a little coaxing to stay where you put it.',
  'overall': 5.0,
  'summary': 'It Does The Job Well',
  'unixReviewTime': 1377648000,
  'reviewTime': '08 28, 2013'},
 {'reviewerID': 'A2C00NNG1ZQQG2',
  'asin': '1384719342',
  'reviewerName': 'RustyBill "Sunday Rocker"',
  'helpful': [0, 0],
  'reviewText': 'Nice windscreen protects my MXL mic and prevents pops. Only thing is that the gooseneck is only marginally able to hold the screen in position and requires careful positioning of the clamp to avoid sagging.',
  'overall': 5.0,
  'summary': 'GOOD WINDSCREEN FOR THE MONEY',
  'unixReviewTime': 1392336000,
  'reviewTime': '02 14, 2014'},
 {'reviewerID': 'A94QU4C90B1AX',
  'asin': '1384719342',
  'reviewerName': 'SEAN MASLANKA',
  'helpful': [0, 0],
  'reviewText': "This pop filter is great. It looks and performs like a studio filter. If you're recording vocals this will eliminate the pops that gets recorded when you sing.",
  'overall': 5.0,
  'summary': 'No more pops when I record my vocals.',
  'unixReviewTime': 1392940800,
  'reviewTime': '02 21, 2014'},
 {'reviewerID': 'A2A039TZMZHH9Y',
  'asin': 'B00004Y2UT',
  'reviewerName': 'Bill Lewey "blewey"',
  'helpful': [0, 0],
  'reviewText': 'So good that I bought another one. Love the heavy cord and gold connectors. Bass sounds great. I just learned last night how to coil them

up. I guess I should read instructions more carefully. But no harm done, still works great!',
  'overall': 5.0,
  'summary': 'The Best Cable',
  'unixReviewTime': 1356048000,
  'reviewTime': '12 21, 2012'},
 {'reviewerID': 'A1UPZM995ZAH90',
  'asin': 'B00004Y2UT',
  'reviewerName': 'Brian',
  'helpful': [0, 0],
  'reviewText': "I have used monster cables for years, and with good reason. The lifetime warranty is worth the price alone. Simple fact: cables break, but getting to replace them at no cost is where it's at.",
  'overall': 5.0,
  'summary': "Monster Standard 100 - 21' Instrument Cable",
  'unixReviewTime': 1390089600,
  'reviewTime': '01 19, 2014'},
 {'reviewerID': 'AJNFQI3YR6XJ5',
  'asin': 'B00004Y2UT',
  'reviewerName': 'Fender Guy "Rick"',
  'helpful': [0, 0],
  'reviewText': "I now use this cable to run from the output of my pedal chain to the input of my Fender Amp. After I bought Monster Cable to hook up my pedal board I thought I would try another one and update my guitar. I had been using a high end Planet Waves cable that I bought in the 1980's… Once I found out the input jacks on the new Monster cable didn't fit into the Fender Strat jack I was a little disappointed… I didn't return it and as stated I use it for the output on the pedal board. Save your money… I went back to my Planet Waves Cable…I payed $30.00 back in the eighties for the Planet Waves which now comes in at around $50.00. What I'm getting at is you get what you pay for. I thought Waves was a lot of money back in the day…but I haven't bought a guitar cable since this one…20 plus years and still working…Planet Waves wins.",
  'overall': 3.0,
  'summary': "Didn't fit my 1996 Fender Strat…",
  'unixReviewTime': 1353024000,
  'reviewTime': '11 16, 2012'},
 {'reviewerID': 'A3M1PLEYNDEYO8',
  'asin': 'B00004Y2UT',
  'reviewerName': 'G. Thomas "Tom"',
  'helpful': [0, 0],
  'reviewText': 'Perfect for my Epiphone Sheraton II. Monster cables are well constructed. I have several and never had any problems with any of them over the years. Got this one because I wanted the 90 degree plug.',
  'overall': 5.0,
  'summary': 'Great cable',
  'unixReviewTime': 1215302400,
  'reviewTime': '07 6, 2008'},

```
 {'reviewerID': 'AMNTZU1YQN1TH',
  'asin': 'B00004Y2UT',
  'reviewerName': 'Kurt Robair',
  'helpful': [0, 0],
  'reviewText': 'Monster makes the best cables and a lifetime warranty doesnt
hurt either. This isnt their top of the line series but it works great with my
bass guitar rig and has for some time. You cant go wrong with Monster Cables.',
  'overall': 5.0,
  'summary': 'Best Instrument Cables On The Market',
  'unixReviewTime': 1389139200,
  'reviewTime': '01 8, 2014'}]
```

Y tendríamos una lista de diccionarios. En la siguiente sesión veremos cómo aprovechar este último archivo para directamente leerlo con pandas en un dataframe.

De esta forma ya tendríamos el contenido de los json en nuestras variables y ya sería cosa de jugar con ellos, como veremos en la siguiente sesión. Ahora, para terminar, veamos...

0.1.3 Escritura en archivos json

Lo primero antes de ponernos a escribir como locos es puntualizar, para poder escribir en archivos json, lo que vayas a escribir debe cumplir con la especificación de lo que es un objeto Json (aqui, por si no te consigues dormir esta noche).

Básicamente para nosotros cualquier diccionario o lista de diccionarios podría escribirse (empleando la libreía json) a un fichero y que luego se pueda leer por cualquier otro programa y lenguaje como json. Pero ojo, no siempre será así (las imágenes no se pueden "serializar" en json por ejemplo) y alguna vez nos dirá "imposible serializar en Json" que será la forma amable de mandarnos a la porra. [Vale olvidate de esto por ahora y piensa que mientras no ocurra lo contrario podemos guardar eso listas de diccionarios y diccionarios]

[ ]:
```
```

Por ejemplo, creemos el diccionario:

[9]:
```
dicc_ejemplo = {"nombre": "Motomami","cantante":"Rosalia", "anyo": "2021"}
```

Y guardémolso en un fichero .json:

[10]:
```
with open("./data/ejemplo.json","w") as g:
    json.dump(dicc_ejemplo, g)
```

Ahora cualquier programador en el lenguaje que quiera con su lector de json específico puede leer ese fichero.

[11]:
```
with open("./data/ejemplo.json","r") as f:
    datos_ejemplo = json.load(f)

datos_ejemplo
```

[11]:
```
{'nombre': 'Motomami', 'cantante': 'Rosalia', 'anyo': '2021'}
```

Para terminar, puedes hacer lo mismo con una lista de diccionarios:

[12]:
```
lista = []
for i in range(4):
    dicc_valor = dicc_ejemplo.copy()
    dicc_valor["cantante"] = f"Rosalia_{i}"
    lista.append(dicc_valor)
print(lista)
```

```
[{'nombre': 'Motomami', 'cantante': 'Rosalia_0', 'anyo': '2021'}, {'nombre':
'Motomami', 'cantante': 'Rosalia_1', 'anyo': '2021'}, {'nombre': 'Motomami',
'cantante': 'Rosalia_2', 'anyo': '2021'}, {'nombre': 'Motomami', 'cantante':
'Rosalia_3', 'anyo': '2021'}]
```

[13]:
```
with open("./data/ejemplo_lista.json","w") as g:
    json.dump(lista,g)
```

[14]:
```
with open("./data/ejemplo_lista.json","r") as f:
    print(json.load(f))
```

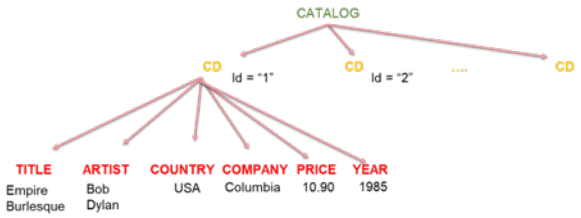
```
[{'nombre': 'Motomami', 'cantante': 'Rosalia_0', 'anyo': '2021'}, {'nombre':
'Motomami', 'cantante': 'Rosalia_1', 'anyo': '2021'}, {'nombre': 'Motomami',
'cantante': 'Rosalia_2', 'anyo': '2021'}, {'nombre': 'Motomami', 'cantante':
'Rosalia_3', 'anyo': '2021'}]
```

[ ]:
```

```

06_Ficheros_xml_I

November 29, 2023



nos servira para el webscrping y el procesado de ficheros XNL en el proximo spring

0.1 ACCESO LOCAL: XML (I)

El formato **XML** (eXtensible Markup Language) es parecido al HTML, pero es más estructurado y es un tipo de formato de fichero que tiene cierto uso, y a diferencia del json, es más probable que encuentres datos que te interesen en este formato. [Ojo que eso no le quita importancia al Json].

Estructura de un archivo XML.- es un fichero de texto plano (como los txt) y su comntenido es asi, y las etiquetas quye lo conforman cada uno los pone donde quiera(arbitrarias), lo unico obligatorio es la XML declaration (

), las etiquetas entre parentesis picudos y se abrane y cierran como HTML . Este archivo abre el nod contenido de otros nodos(result), el cual tien un atributo "time="200" y que dentro pueden tener otros subnodos(value) el cual tiene los atributos id , los cuales tienen unos valores (33.3, 1.00)

Además de lo visto para el ejemplo de arriba sobre etiquetas, una forma de ver los archivos XML es como si formasen un estructura de arbol en el que van apareciendo conjuntos de valores que pertenencen a otros conjuntos de valores a partir de un nodo raíz:

0.1.1 Lectura de ficheros XML

Antes de utilizar la correspondiente librería de python (ojo esta no se llama `xml`) que nos ayude a procesar un fichero XML y extraer la información que pueda contener, veamos uno como texto plano que es:

[3]:
```
with open ("./data/cd_catalog.xml", "r") as f:
    for line in f:
        print(line, end="")
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
  <CD id="1">
```

```
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
</CD>
<CD id="2">
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>CBS Records</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1988</YEAR>
</CD>
<CD id="3">
  <TITLE>Greatest Hits</TITLE>
  <ARTIST>Dolly Parton</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>RCA</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1982</YEAR>
</CD>
<CD id="4">
  <TITLE>Still got the blues</TITLE>
  <ARTIST>Gary Moore</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Virgin records</COMPANY>
  <PRICE>10.20</PRICE>
  <YEAR>1990</YEAR>
</CD>
<CD id="5">
  <TITLE>Eros</TITLE>
  <ARTIST>Eros Ramazzotti</ARTIST>
  <COUNTRY>EU</COUNTRY>
  <COMPANY>BMG</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1997</YEAR>
</CD>
<CD id="6">
  <TITLE>One night only</TITLE>
  <ARTIST>Bee Gees</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Polydor</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1998</YEAR>
</CD>
<CD id="7">
```

```
  <TITLE>Sylvias Mother</TITLE>
  <ARTIST>Dr.Hook</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>CBS</COMPANY>
  <PRICE>8.10</PRICE>
  <YEAR>1973</YEAR>
</CD>
<CD id="8">
  <TITLE>Maggie May</TITLE>
  <ARTIST>Rod Stewart</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Pickwick</COMPANY>
  <PRICE>8.50</PRICE>
  <YEAR>1990</YEAR>
</CD>
<CD id="9">
  <TITLE>Romanza</TITLE>
  <ARTIST>Andrea Bocelli</ARTIST>
  <COUNTRY>EU</COUNTRY>
  <COMPANY>Polydor</COMPANY>
  <PRICE>10.80</PRICE>
  <YEAR>1996</YEAR>
</CD>
<CD id="10">
  <TITLE>When a man loves a woman</TITLE>
  <ARTIST>Percy Sledge</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Atlantic</COMPANY>
  <PRICE>8.70</PRICE>
  <YEAR>1987</YEAR>
</CD>
<CD id="11">
  <TITLE>Black angel</TITLE>
  <ARTIST>Savage Rose</ARTIST>
  <COUNTRY>EU</COUNTRY>
  <COMPANY>Mega</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1995</YEAR>
</CD>
<CD id="12">
  <TITLE>1999 Grammy Nominees</TITLE>
  <ARTIST>Many</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Grammy</COMPANY>
  <PRICE>10.20</PRICE>
  <YEAR>1999</YEAR>
</CD>
<CD id="13">
```

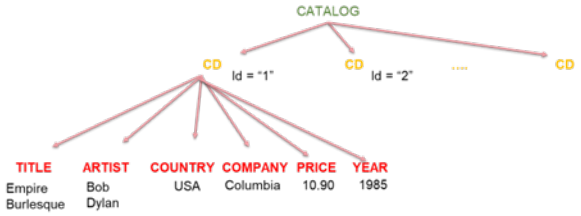
```
  <TITLE>For the good times</TITLE>
  <ARTIST>Kenny Rogers</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Mucik Master</COMPANY>
  <PRICE>8.70</PRICE>
  <YEAR>1995</YEAR>
</CD>
<CD id="14">
  <TITLE>Big Willie style</TITLE>
  <ARTIST>Will Smith</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1997</YEAR>
</CD>
<CD id="15">
  <TITLE>Tupelo Honey</TITLE>
  <ARTIST>Van Morrison</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Polydor</COMPANY>
  <PRICE>8.20</PRICE>
  <YEAR>1971</YEAR>
</CD>
<CD id="16">
  <TITLE>Soulsville</TITLE>
  <ARTIST>Jorn Hoel</ARTIST>
  <COUNTRY>Norway</COUNTRY>
  <COMPANY>WEA</COMPANY>
  <PRICE>7.90</PRICE>
  <YEAR>1996</YEAR>
</CD>
<CD id="17">
  <TITLE>The very best of</TITLE>
  <ARTIST>Cat Stevens</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Island</COMPANY>
  <PRICE>8.90</PRICE>
  <YEAR>1990</YEAR>
</CD>
<CD id="18">
  <TITLE>Stop</TITLE>
  <ARTIST>Sam Brown</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>A and M</COMPANY>
  <PRICE>8.90</PRICE>
  <YEAR>1988</YEAR>
</CD>
<CD id="19">
```

```
  <TITLE>Bridge of Spies</TITLE>
  <ARTIST>T'Pau</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Siren</COMPANY>
  <PRICE>7.90</PRICE>
  <YEAR>1987</YEAR>
</CD>
<CD id="20">
  <TITLE>Private Dancer</TITLE>
  <ARTIST>Tina Turner</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Capitol</COMPANY>
  <PRICE>8.90</PRICE>
  <YEAR>1983</YEAR>
</CD>
<CD id="21">
  <TITLE>Midt om natten</TITLE>
  <ARTIST>Kim Larsen</ARTIST>
  <COUNTRY>EU</COUNTRY>
  <COMPANY>Medley</COMPANY>
  <PRICE>7.80</PRICE>
  <YEAR>1983</YEAR>
</CD>
<CD id="22">
  <TITLE>Pavarotti Gala Concert</TITLE>
  <ARTIST>Luciano Pavarotti</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>DECCA</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1991</YEAR>
</CD>
<CD id="23">
  <TITLE>The dock of the bay</TITLE>
  <ARTIST>Otis Redding</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Stax Records</COMPANY>
  <PRICE>7.90</PRICE>
  <YEAR>1968</YEAR>
</CD>
<CD id="24">
  <TITLE>Picture book</TITLE>
  <ARTIST>Simply Red</ARTIST>
  <COUNTRY>EU</COUNTRY>
  <COMPANY>Elektra</COMPANY>
  <PRICE>7.20</PRICE>
  <YEAR>1985</YEAR>
</CD>
<CD id="25">
```

```
    <TITLE>Red</TITLE>
    <ARTIST>The Communards</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>London</COMPANY>
    <PRICE>7.80</PRICE>
    <YEAR>1987</YEAR>
  </CD>
  <CD id="99">
    <TITLE>Unchain my heart</TITLE>
    <ARTIST>Joe Cocker</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>EMI</COMPANY>
    <PRICE>8.20</PRICE>
    <YEAR>1987</YEAR>
  </CD>
</CATALOG>
```

[ ]:

Y si hacemos un pequeño ejercicio de interpretar las etiquetas llegaríamos a un árbol como este:



Lo que podría interesarnos es sacar los datos por CD para construirnos una tabla Catalog y ahí poner una fila por cd con su identificador (que podría ser nuestro índice de `DataFrame` por ejemplo) y los valores de cada una de esas etiquetas (TITLE, ARTIST, etc) las columnas... ¿Y cómo hacerlo?

Lo vamos a hacer con ayuda de la librería `ElementTree`:

[4]:
```
import xml.etree.ElementTree as ET
```

Lo primero es leer y parsear el fichero de esta forma:

[5]:
```
tree= ET.parse("./data/cd_catalog.xml")
```

0.1.2 Procesando un fichero XML

Esta librería trata el XML como si fuese un árbol. En este formato de árbol, disponemos de diversos métodos con los que podemos extraer partes del XML.

- `tag` muestra el texto dentro de la etiqueta
- `attrib` muestra los atributos de la etiqueta
- `text` muestra el texto del nodo
- La función `iter()` permite conocer la estructura del XML

- La función `find()` busca en el XML y devuelve el elemento que coincide con la etiqueta especificada.

- La función `findall()` devuelve todos los elementos con cierta etiqueta

Una vez cargado juguemos con los métodos anteriores para obtener la información o los datos útiles de nuestro catálogo de CDs:

[6]:
```
# Obtener la etiqueta del nodo raiz:

raiz = tree.getroot()
raiz.tag
```

[6]:
```
'CATALOG'
```

Bien, poco hemos avanzado, porque esto ya lo intuíamos, vamos más allá y capturemos para cada elemento el nombre de sus etiquetas (aquí sólo tenemos CD como elemento, en el caso de la figura anterior teníamos DEPARTMENT, EMPLOYEE)

[7]:
```
# Para cada elemento sus etiquetas
for elemento in raiz.iter():
    print(elemento.tag)
```

```
CATALOG
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
```

COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST

```
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
```

```
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
```

```
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
CD
TITLE
ARTIST
COUNTRY
COMPANY
PRICE
YEAR
```

Esto lo aplana un poco y nos dice mucho sobre la jerarquía pero podemos ver jugar un poco para ver la estructura de forma más jerarquica además de ver los atributos y los valores (si hubiera unos u otros):

[8]:
```
# Recorrido con cierta jerarquia
for hijo in raiz:
    tabs ="\t"
    print(hijo.tag, hijo.attrib, hijo.texto)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING␣
 ↪5\Acceso_local_y_Tratamiento_de_Datos\unit␣
 ↪2_Acceso_y_Procesado_de_Datos_Internos__Ficheros\06_Ficheros_xml_I.ipynb Cell␣
 ↪25 line 4
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/06_Ficheros_xml_I.
 ↪ipynb#X32sZmlsZQ%3D%3D?line=1'>2</a> for hijo in raiz:
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/06_Ficheros_xml_I.
 ↪ipynb#X32sZmlsZQ%3D%3D?line=2'>3</a>     tabs ="\t"
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/06_Ficheros_xml_I.
 ↪ipynb#X32sZmlsZQ%3D%3D?line=3'>4</a>     print(hijo.tag, hijo.attrib, hijo.
 ↪texto)

AttributeError: 'xml.etree.ElementTree.Element' object has no attribute 'texto'
```

[9]:
```
for hijo in raiz:
    tabs ="\t"
    print(hijo.tag, hijo.attrib)# si attrib tiene valor te devuelve un␣
 ↪diccionario
```

```
CD {'id': '1'}
CD {'id': '2'}
CD {'id': '3'}
CD {'id': '4'}
CD {'id': '5'}
CD {'id': '6'}
CD {'id': '7'}
CD {'id': '8'}
CD {'id': '9'}
CD {'id': '10'}
CD {'id': '11'}
CD {'id': '12'}
CD {'id': '13'}
CD {'id': '14'}
CD {'id': '15'}
CD {'id': '16'}
CD {'id': '17'}
CD {'id': '18'}
CD {'id': '19'}
CD {'id': '20'}
CD {'id': '21'}
CD {'id': '22'}
CD {'id': '23'}
CD {'id': '24'}
CD {'id': '25'}
CD {'id': '99'}
```

[10]:
```
for hijo in raiz:
    tabs ="\t"
    print(hijo.tag, hijo.attrib)
    for nieto in hijo:
        print(tabs, nieto.tag,nieto.text)
```

```
CD {'id': '1'}
         TITLE Empire Burlesque
         ARTIST Bob Dylan
         COUNTRY USA
         COMPANY Columbia
         PRICE 10.90
         YEAR 1985
CD {'id': '2'}
         TITLE Hide your heart
         ARTIST Bonnie Tyler
         COUNTRY UK
         COMPANY CBS Records
         PRICE 9.90
         YEAR 1988
CD {'id': '3'}
```

```
        TITLE Greatest Hits
        ARTIST Dolly Parton
        COUNTRY USA
        COMPANY RCA
        PRICE 9.90
        YEAR 1982
CD {'id': '4'}
        TITLE Still got the blues
        ARTIST Gary Moore
        COUNTRY UK
        COMPANY Virgin records
        PRICE 10.20
        YEAR 1990
CD {'id': '5'}
        TITLE Eros
        ARTIST Eros Ramazzotti
        COUNTRY EU
        COMPANY BMG
        PRICE 9.90
        YEAR 1997
CD {'id': '6'}
        TITLE One night only
        ARTIST Bee Gees
        COUNTRY UK
        COMPANY Polydor
        PRICE 10.90
        YEAR 1998
CD {'id': '7'}
        TITLE Sylvias Mother
        ARTIST Dr.Hook
        COUNTRY UK
        COMPANY CBS
        PRICE 8.10
        YEAR 1973
CD {'id': '8'}
        TITLE Maggie May
        ARTIST Rod Stewart
        COUNTRY UK
        COMPANY Pickwick
        PRICE 8.50
        YEAR 1990
CD {'id': '9'}
        TITLE Romanza
        ARTIST Andrea Bocelli
        COUNTRY EU
        COMPANY Polydor
        PRICE 10.80
        YEAR 1996
```

```
CD {'id': '10'}
         TITLE When a man loves a woman
         ARTIST Percy Sledge
         COUNTRY USA
         COMPANY Atlantic
         PRICE 8.70
         YEAR 1987
CD {'id': '11'}
         TITLE Black angel
         ARTIST Savage Rose
         COUNTRY EU
         COMPANY Mega
         PRICE 10.90
         YEAR 1995
CD {'id': '12'}
         TITLE 1999 Grammy Nominees
         ARTIST Many
         COUNTRY USA
         COMPANY Grammy
         PRICE 10.20
         YEAR 1999
CD {'id': '13'}
         TITLE For the good times
         ARTIST Kenny Rogers
         COUNTRY UK
         COMPANY Mucik Master
         PRICE 8.70
         YEAR 1995
CD {'id': '14'}
         TITLE Big Willie style
         ARTIST Will Smith
         COUNTRY USA
         COMPANY Columbia
         PRICE 9.90
         YEAR 1997
CD {'id': '15'}
         TITLE Tupelo Honey
         ARTIST Van Morrison
         COUNTRY UK
         COMPANY Polydor
         PRICE 8.20
         YEAR 1971
CD {'id': '16'}
         TITLE Soulsville
         ARTIST Jorn Hoel
         COUNTRY Norway
         COMPANY WEA
         PRICE 7.90
```

```
        YEAR 1996
CD {'id': '17'}
        TITLE The very best of
        ARTIST Cat Stevens
        COUNTRY UK
        COMPANY Island
        PRICE 8.90
        YEAR 1990
CD {'id': '18'}
        TITLE Stop
        ARTIST Sam Brown
        COUNTRY UK
        COMPANY A and M
        PRICE 8.90
        YEAR 1988
CD {'id': '19'}
        TITLE Bridge of Spies
        ARTIST T'Pau
        COUNTRY UK
        COMPANY Siren
        PRICE 7.90
        YEAR 1987
CD {'id': '20'}
        TITLE Private Dancer
        ARTIST Tina Turner
        COUNTRY UK
        COMPANY Capitol
        PRICE 8.90
        YEAR 1983
CD {'id': '21'}
        TITLE Midt om natten
        ARTIST Kim Larsen
        COUNTRY EU
        COMPANY Medley
        PRICE 7.80
        YEAR 1983
CD {'id': '22'}
        TITLE Pavarotti Gala Concert
        ARTIST Luciano Pavarotti
        COUNTRY UK
        COMPANY DECCA
        PRICE 9.90
        YEAR 1991
CD {'id': '23'}
        TITLE The dock of the bay
        ARTIST Otis Redding
        COUNTRY USA
        COMPANY Stax Records
```

```
          PRICE 7.90
          YEAR 1968
CD {'id': '24'}
          TITLE Picture book
          ARTIST Simply Red
          COUNTRY EU
          COMPANY Elektra
          PRICE 7.20
          YEAR 1985
CD {'id': '25'}
          TITLE Red
          ARTIST The Communards
          COUNTRY UK
          COMPANY London
          PRICE 7.80
          YEAR 1987
CD {'id': '99'}
          TITLE Unchain my heart
          ARTIST Joe Cocker
          COUNTRY USA
          COMPANY EMI
          PRICE 8.20
          YEAR 1987
```

Además puedo buscar elementos por sus etiquetas y hacer lo anterior de otra forma

[11]:
```
cds = tree.findall("CD")
for cd in cds:
    print("Id:", cd.attrib["id"])
    print("Titulo:", cd.find("TITLE").text)
    print("Artista:", cd.find("ARTIST").text)
```

```
Id: 1
Titulo: Empire Burlesque
Artista: Bob Dylan
Id: 2
Titulo: Hide your heart
Artista: Bonnie Tyler
Id: 3
Titulo: Greatest Hits
Artista: Dolly Parton
Id: 4
Titulo: Still got the blues
Artista: Gary Moore
Id: 5
Titulo: Eros
Artista: Eros Ramazzotti
Id: 6
Titulo: One night only
```

```
Artista: Bee Gees
Id: 7
Titulo: Sylvias Mother
Artista: Dr.Hook
Id: 8
Titulo: Maggie May
Artista: Rod Stewart
Id: 9
Titulo: Romanza
Artista: Andrea Bocelli
Id: 10
Titulo: When a man loves a woman
Artista: Percy Sledge
Id: 11
Titulo: Black angel
Artista: Savage Rose
Id: 12
Titulo: 1999 Grammy Nominees
Artista: Many
Id: 13
Titulo: For the good times
Artista: Kenny Rogers
Id: 14
Titulo: Big Willie style
Artista: Will Smith
Id: 15
Titulo: Tupelo Honey
Artista: Van Morrison
Id: 16
Titulo: Soulsville
Artista: Jorn Hoel
Id: 17
Titulo: The very best of
Artista: Cat Stevens
Id: 18
Titulo: Stop
Artista: Sam Brown
Id: 19
Titulo: Bridge of Spies
Artista: T'Pau
Id: 20
Titulo: Private Dancer
Artista: Tina Turner
Id: 21
Titulo: Midt om natten
Artista: Kim Larsen
Id: 22
Titulo: Pavarotti Gala Concert
```

```
Artista: Luciano Pavarotti
Id: 23
Titulo: The dock of the bay
Artista: Otis Redding
Id: 24
Titulo: Picture book
Artista: Simply Red
Id: 25
Titulo: Red
Artista: The Communards
Id: 99
Titulo: Unchain my heart
Artista: Joe Cocker
```

[ ]:

[ ]:

Usando cualquiera de las dos formas anteriores podría recorrer el XML y crear una estructura con la que luego obtener un `DataFrame`, pero eso lo veremos en la siguiente sesión.

[ ]:

07_Ficheros_xml_II

November 29, 2023

ETL Y DATOS

0.1 ACCESO LOCAL: XML (II)

Vamos a crear un `DataFrame` a partir de los datos que puedes encontrar en el fichero "movies.xml" en el directorio "data". Este es un notebook especial, igual que están las sesiones vagas, esta es una sesión only coding... vamos a ello: (es decir sin texto de apoyo)

[1]:
```
import pandas as pd
import xml.etree.ElementTree as ET
```

[2]:
```
with open (".\data/movies.xml", "r") as f:
    for line in f:
        print(line, end="")
```

```
<?xml version="1.0" encoding="UTF-8"?>

<movies>
  <movie>
    <title>A History of Violence</title>
    <year>2005</year>
    <country>USA</country>
    <genre>Crime</genre>
    <summary>Tom Stall, a humble family man and owner of a
        popular neighborhood restaurant, lives a quiet but
        fulfilling existence in the Midwest. One night Tom
        foils a crime at his place of business and, to his
        chagrin, is plastered all over the news for his
        heroics. Following this, mysterious people follow
```

```
        the Stalls' every move, concerning Tom more than
        anyone else. As this situation is confronted, more
        lurks out over where all these occurrences have
        stemmed from compromising his marriage, family
        relationship and the main characters' former
        relations in the process.</summary>
 <director>         <last_name>Cronenberg</last_name>
            <first_name>David</first_name>
            <birth_date>1943</birth_date>
</director> <actor>
            <first_name>Vigo</first_name>
            <last_name>Mortensen</last_name>
            <birth_date>1958</birth_date>
            <role>Tom Stall</role>
</actor>
<actor>
            <first_name>Maria</first_name>
            <last_name>Bello</last_name>
            <birth_date>1967</birth_date>
            <role>Eddie Stall</role>
</actor>
<actor>
            <first_name>Ed</first_name>
            <last_name>Harris</last_name>
            <birth_date>1950</birth_date>
            <role>Carl Fogarty</role>
</actor>
<actor>
            <first_name>William</first_name>
            <last_name>Hurt</last_name>
            <birth_date>1950</birth_date>
            <role>Richie Cusack</role>
</actor>
 </movie>
  <movie>
    <title>Heat</title>
    <year>1995</year>
    <country>USA</country>
    <genre>Crime</genre>
    <summary>Hunters and their prey--Neil and his professional
        criminal crew hunt to score big money targets
        (banks, vaults, armored cars) and are, in turn,
        hunted by Lt. Vincent Hanna and his team of cops
        in the Robbery/Homicide police division. A botched
        job puts Hanna onto their trail while they regroup
        and try to put together one last big 'retirement'
        score. Neil and Vincent are similar in many ways,
        including their troubled personal lives. At a
```

```
        crucial moment in his life, Neil disobeys the
        dictum taught to him long ago by his criminal
        mentor--'Never have anything in your life that you
        can't walk out on in thirty seconds flat, if you
        spot the heat coming around the corner'--as he
        falls in love. Thus the stage is set for the
        suspenseful ending… </summary>
 <director>         <last_name>Mann</last_name>
            <first_name>Michael</first_name>
            <birth_date>1943</birth_date>
</director> <actor>
            <first_name>Al</first_name>
            <last_name>Pacino</last_name>
            <birth_date>1940</birth_date>
            <role>Lt. Vincent Hanna</role>
</actor>
<actor>
            <first_name>Robert</first_name>
            <last_name>De Niro</last_name>
            <birth_date>1943</birth_date>
            <role>Neil McCauley</role>
</actor>
<actor>
            <first_name>Val</first_name>
            <last_name>Kilmer</last_name>
            <birth_date>1959</birth_date>
            <role>Chris Shiherlis</role>
</actor>
<actor>
            <first_name>Jon</first_name>
            <last_name>Voight</last_name>
            <birth_date>1938</birth_date>
            <role>Nate</role>
</actor>
 </movie>
  <movie>
    <title>Unforgiven</title>
    <year>1992</year>
    <country>USA</country>
    <genre>Western</genre>
    <summary>The town of Big Whisky is full of normal people
        trying to lead quiet lives. Cowboys try to make a
        living. Sheriff 'Little Bill' tries to build a
        house and keep a heavy-handed order. The town
        whores just try to get by.Then a couple of cowboys
        cut up a whore. Unsatisfied with Bill's justice,
        the prostitutes put a bounty on the cowboys. The
        bounty attracts a young gun billing himself as
```

```
        'The Schofield Kid', and aging killer William
        Munny. Munny reformed for his young wife, and has
        been raising crops and two children in peace. But
        his wife is gone. Farm life is hard. And Munny is
        no good at it. So he calls his old partner Ned,
        saddles his ornery nag, and rides off to kill one
        more time, blurring the lines between heroism and
        villainy, man and myth.</summary>
 <director>         <last_name>Eastwood</last_name>
            <first_name>Clint</first_name>
            <birth_date>1930</birth_date>
</director> <actor>
            <first_name>Clint</first_name>
            <last_name>Eastwood</last_name>
            <birth_date>1930</birth_date>
            <role>William 'Bill' Munny</role>
</actor>
<actor>
            <first_name>Gene</first_name>
            <last_name>Hackman</last_name>
            <birth_date>1930</birth_date>
            <role>Little Bill Daggett</role>
</actor>
<actor>
            <first_name>Morgan</first_name>
            <last_name>Freeman</last_name>
            <birth_date>1937</birth_date>
            <role>Ned Logan</role>
</actor>
 </movie>
  <movie>
    <title>Match Point</title>
    <year>2005</year>
    <country>USA</country>
    <genre>Crime</genre>
    <summary>Chris Wilton is a former tennis pro, looking to
        find work as an instructor. He meets Tom Hewett, a
        well-off pretty boy. Tom's sister Chloe falls in
        love with Chris but Chris has his eyes on Tom's
        fiancÃ©e, the luscious Nola. Both Chris and Nola
        know it's wrong but what could be more right than
        love? Chris tries to juggle both women but at some
        point, he must choose between them…</summary>
 <director>         <last_name>Allen</last_name>
            <first_name>Woody</first_name>
            <birth_date>1935</birth_date>
</director> <actor>
            <first_name>Jonathan</first_name>
```

```
            <last_name>Rhys Meyers</last_name>
            <birth_date>1977</birth_date>
            <role>Chris Wilton</role>
</actor>
<actor>
            <first_name>Scarlett </first_name>
            <last_name>Johansson</last_name>
            <birth_date>1984</birth_date>
            <role>Nola Rice</role>
</actor>
 </movie>
  <movie>
    <title>Lost in Translation</title>
    <year>2003</year>
    <country>USA</country>
    <genre>Drama</genre>
 <director>         <last_name>Coppola</last_name>
            <first_name>Sofia</first_name>
            <birth_date>1971</birth_date>
</director> <actor>
            <first_name>Scarlett </first_name>
            <last_name>Johansson</last_name>
            <birth_date>1984</birth_date>
            <role>Charlotte</role>
</actor>
<actor>
            <first_name>Bill</first_name>
            <last_name>Murray</last_name>
            <birth_date>1950</birth_date>
            <role>Bob Harris</role>
</actor>
 </movie>
  <movie>
    <title>Marie Antoinette</title>
    <year>2006</year>
    <country>USA</country>
    <genre>Drama</genre>
    <summary>Based on Antonia Fraser's book about the ill-fated
        Archduchess of Austria and later Queen of France,
        'Marie Antoinette' tells the story of the most
        misunderstood and abused woman in history, from
        her birth in Imperial Austria to her later life in
        France. </summary>
 <director>         <last_name>Coppola</last_name>
            <first_name>Sofia</first_name>
            <birth_date>1971</birth_date>
</director> <actor>
            <first_name>Kirsten</first_name>
```

```
            <last_name>Dunst</last_name>
            <birth_date>1982</birth_date>
            <role>Marie Antoinette</role>
</actor>
<actor>
            <first_name>Jason </first_name>
            <last_name>Schwartzman</last_name>
            <birth_date>1980</birth_date>
            <role>Louis XVI</role>
</actor>
 </movie>
  <movie>
    <title>Spider-Man</title>
    <year>2002</year>
    <country>USA</country>
    <genre>Action</genre>
    <summary>On a school field trip, Peter Parker (Maguire) is
        bitten by a genetically modified spider. He wakes
        up the next morning with incredible powers. After
        witnessing the death of his uncle (Robertson),
        Parkers decides to put his new skills to use in
        order to rid the city of evil, but someone else
        has other plans. The Green Goblin (Dafoe) sees
        Spider-Man as a threat and must dispose of him.
        Even if it means the Goblin has to target Parker's
        Aunt (Harris) and the girl he secretly pines for
        (Dunst) </summary>
 <director>         <last_name>Raimi</last_name>
            <first_name>Sam</first_name>
            <birth_date>1959</birth_date>
</director> <actor>
            <first_name>Kirsten</first_name>
            <last_name>Dunst</last_name>
            <birth_date>1982</birth_date>
            <role>Mary Jane Watson</role>
</actor>
<actor>
            <first_name>Tobey</first_name>
            <last_name>Maguire</last_name>
            <birth_date>1975</birth_date>
            <role>Spider-Man / Peter Parker</role>
</actor>
<actor>
            <first_name>Willem</first_name>
            <last_name>Dafoe</last_name>
            <birth_date>1955</birth_date>
            <role>Green Goblin / Norman Osborn</role>
</actor>
```

```
 </movie>

</movies>
```

tiene movies el nodo raiz, tiene algunos tags, y vamos a intenar crearnos un DF, calcularemos el nodo rariz y a aprtir de ahi ir sacando la info

[3]:
```
# primero parsear
tree_movies =ET.parse("./data/movies.xml")
raiz = tree_movies.getroot()
print(raiz.tag)
```

```
movies
```

[4]:
```
# recorrer raiz para hijos y nietos( este xml esta hecho para hacer una funcion␣
 ↪recursiva )
for hijo in raiz:
    tabs ="\t"# para que sea mas tabulada la info
    level = 0
    print(hijo.tag)
    for nieto in hijo:
        level=1
        print(tabs*level, nieto.tag, nieto.text)
        for bisnieto in nieto:
            level = 2
            print(tabs*level, bisnieto.tag, bisnieto.text)
            for tataranieto in bisnieto:
                level =3
                print(tabs*level, tataranieto.tag, tataranieto.text)
```

```
movie
         title A History of Violence
         year 2005
         country USA
         genre Crime
         summary Tom Stall, a humble family man and owner of a
        popular neighborhood restaurant, lives a quiet but
        fulfilling existence in the Midwest. One night Tom
        foils a crime at his place of business and, to his
        chagrin, is plastered all over the news for his
        heroics. Following this, mysterious people follow
        the Stalls' every move, concerning Tom more than
        anyone else. As this situation is confronted, more
        lurks out over where all these occurrences have
        stemmed from compromising his marriage, family
        relationship and the main characters' former
        relations in the process.
         director
                 last_name Cronenberg
```

```
                first_name David
                birth_date 1943
        actor

                first_name Vigo
                last_name Mortensen
                birth_date 1958
                role Tom Stall
        actor

                first_name Maria
                last_name Bello
                birth_date 1967
                role Eddie Stall
        actor

                first_name Ed
                last_name Harris
                birth_date 1950
                role Carl Fogarty
        actor

                first_name William
                last_name Hurt
                birth_date 1950
                role Richie Cusack
movie
        title Heat
        year 1995
        country USA
        genre Crime
        summary Hunters and their prey--Neil and his professional
       criminal crew hunt to score big money targets
       (banks, vaults, armored cars) and are, in turn,
       hunted by Lt. Vincent Hanna and his team of cops
       in the Robbery/Homicide police division. A botched
       job puts Hanna onto their trail while they regroup
       and try to put together one last big 'retirement'
       score. Neil and Vincent are similar in many ways,
       including their troubled personal lives. At a
       crucial moment in his life, Neil disobeys the
       dictum taught to him long ago by his criminal
       mentor--'Never have anything in your life that you
       can't walk out on in thirty seconds flat, if you
       spot the heat coming around the corner'--as he
       falls in love. Thus the stage is set for the
       suspenseful ending…
        director
```

```
                last_name Mann
                first_name Michael
                birth_date 1943
        actor

                first_name Al
                last_name Pacino
                birth_date 1940
                role Lt. Vincent Hanna
        actor

                first_name Robert
                last_name De Niro
                birth_date 1943
                role Neil McCauley
        actor

                first_name Val
                last_name Kilmer
                birth_date 1959
                role Chris Shiherlis
        actor

                first_name Jon
                last_name Voight
                birth_date 1938
                role Nate
movie
        title Unforgiven
        year 1992
        country USA
        genre Western
        summary The town of Big Whisky is full of normal people
       trying to lead quiet lives. Cowboys try to make a
       living. Sheriff 'Little Bill' tries to build a
       house and keep a heavy-handed order. The town
       whores just try to get by.Then a couple of cowboys
       cut up a whore. Unsatisfied with Bill's justice,
       the prostitutes put a bounty on the cowboys. The
       bounty attracts a young gun billing himself as
       'The Schofield Kid', and aging killer William
       Munny. Munny reformed for his young wife, and has
       been raising crops and two children in peace. But
       his wife is gone. Farm life is hard. And Munny is
       no good at it. So he calls his old partner Ned,
       saddles his ornery nag, and rides off to kill one
       more time, blurring the lines between heroism and
       villainy, man and myth.
```

```
        director
                last_name Eastwood
                first_name Clint
                birth_date 1930
        actor

                first_name Clint
                last_name Eastwood
                birth_date 1930
                role William 'Bill' Munny
        actor

                first_name Gene
                last_name Hackman
                birth_date 1930
                role Little Bill Daggett
        actor

                first_name Morgan
                last_name Freeman
                birth_date 1937
                role Ned Logan
movie
        title Match Point
        year 2005
        country USA
        genre Crime
        summary Chris Wilton is a former tennis pro, looking to
       find work as an instructor. He meets Tom Hewett, a
       well-off pretty boy. Tom's sister Chloe falls in
       love with Chris but Chris has his eyes on Tom's
       fiancée, the luscious Nola. Both Chris and Nola
       know it's wrong but what could be more right than
       love? Chris tries to juggle both women but at some
       point, he must choose between them…
        director
                last_name Allen
                first_name Woody
                birth_date 1935
        actor

                first_name Jonathan
                last_name Rhys Meyers
                birth_date 1977
                role Chris Wilton
        actor

                first_name Scarlett
```

```
                last_name Johansson
                birth_date 1984
                role Nola Rice
movie
         title Lost in Translation
         year 2003
         country USA
         genre Drama
         director
                last_name Coppola
                first_name Sofia
                birth_date 1971
         actor

                first_name Scarlett
                last_name Johansson
                birth_date 1984
                role Charlotte
         actor

                first_name Bill
                last_name Murray
                birth_date 1950
                role Bob Harris
movie
         title Marie Antoinette
         year 2006
         country USA
         genre Drama
         summary Based on Antonia Fraser's book about the ill-fated
        Archduchess of Austria and later Queen of France,
        'Marie Antoinette' tells the story of the most
        misunderstood and abused woman in history, from
        her birth in Imperial Austria to her later life in
        France.
         director
                last_name Coppola
                first_name Sofia
                birth_date 1971
         actor

                first_name Kirsten
                last_name Dunst
                birth_date 1982
                role Marie Antoinette
         actor

                first_name Jason
```

```
                last_name Schwartzman
                birth_date 1980
                role Louis XVI
movie
         title Spider-Man
         year 2002
         country USA
         genre Action
         summary On a school field trip, Peter Parker (Maguire) is
        bitten by a genetically modified spider. He wakes
        up the next morning with incredible powers. After
        witnessing the death of his uncle (Robertson),
        Parkers decides to put his new skills to use in
        order to rid the city of evil, but someone else
        has other plans. The Green Goblin (Dafoe) sees
        Spider-Man as a threat and must dispose of him.
        Even if it means the Goblin has to target Parker's
        Aunt (Harris) and the girl he secretly pines for
        (Dunst)
         director
                last_name Raimi
                first_name Sam
                birth_date 1959
         actor

                first_name Kirsten
                last_name Dunst
                birth_date 1982
                role Mary Jane Watson
         actor

                first_name Tobey
                last_name Maguire
                birth_date 1975
                role Spider-Man / Peter Parker
         actor

                first_name Willem
                last_name Dafoe
                birth_date 1955
                role Green Goblin / Norman Osborn
```

con la extructura que nos ha dado, vamos a fhacer un df por peliculas, años, pais, genero y sumario, director, nombre del mismo y en principo actores lo vamos a dejar porque es un campo mas variable, pero se podria hcer

[5]:
```
# creamos diccionario con los nombres de los tags de nombres de lista, para ser␣
↪mas facil
```

```
dict_df ={
    "title":[],
    "year":[],
    "country":[],
    "genre":[],
    "summary":[],
    "director":[]
}
```

[6]:
```
# recorremos el nodo raiz completo y vamos rellenado los datos
for hijo in raiz:
    tabs ="\t"# para que sea mas tabulada la info
    level = 0
    print(hijo.tag)
    campos_a_rellenar = list(dict_df.keys())#  ahora por cada pelicula versmos␣
 ↪todos los valores que son los tag y las listas a rellenar
    for nieto in hijo:
        level=1
        print(tabs*level, nieto.tag, nieto.text)
        if nieto.tag in dict_df and nieto.tag != "director": # require algo mas␣
 ↪de procesamiento
            dict_df[nieto.tag].append(nieto.text)
            campos_a_rellenar.remove(nieto.tag)
        elif nieto.tag =="director":
            for bisnieto in nieto:
                level = 2
                print(tabs*level, bisnieto.tag, bisnieto.text)
                if bisnieto.tag ==" first_name":
                    nombre = bisnieto.text
                elif bisnieto.tag =="last_name":
                    apellido = bisnieto.text
                nombre_completo = f"{nombre} {apellido}"
                dict_df["director"].append(nombre_completo)
                campos_a_rellenar.remove("director")
    for campo in campos_a_rellenar:
        dict_df[campo].append("no tengo datos")
```

```
movie
         title A History of Violence
         year 2005
         country USA
         genre Crime
         summary Tom Stall, a humble family man and owner of a
        popular neighborhood restaurant, lives a quiet but
        fulfilling existence in the Midwest. One night Tom
        foils a crime at his place of business and, to his
        chagrin, is plastered all over the news for his
```

```
        heroics. Following this, mysterious people follow
        the Stalls' every move, concerning Tom more than
        anyone else. As this situation is confronted, more
        lurks out over where all these occurrences have
        stemmed from compromising his marriage, family
        relationship and the main characters' former
        relations in the process.
         director
                last_name Cronenberg
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING␣
 ↪5\Acceso_local_y_Tratamiento_de_Datos\unit␣
 ↪2_Acceso_y_Procesado_de_Datos_Internos__Ficheros\07_Ficheros_xml_II.ipynb␣
 ↪Celda 10 line 2
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/07_Ficheros_xml_II.
 ↪ipynb#X13sZmlsZQ%3D%3D?line=18'>19</a> elif bisnieto.tag =="last_name":
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/07_Ficheros_xml_II.
 ↪ipynb#X13sZmlsZQ%3D%3D?line=19'>20</a>     apellido = bisnieto.text
---> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/07_Ficheros_xml_II.
 ↪ipynb#X13sZmlsZQ%3D%3D?line=20'>21</a> nombre_completo = f"{nombre} {apellido
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/07_Ficheros_xml_II.
 ↪ipynb#X13sZmlsZQ%3D%3D?line=21'>22</a> dict_df["director"].
 ↪append(nombre_completo)
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
 ↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/07_Ficheros_xml_II.
 ↪ipynb#X13sZmlsZQ%3D%3D?line=22'>23</a> campos_a_rellenar.remove("director")

NameError: name 'nombre' is not defined
```

[ ]:
```
dict_df
```

[ ]:
```
{'title': ['A History of Violence',
  'A History of Violence',
  'A History of Violence',
  'A History of Violence',
  'A History of Violence'],
 'year': ['2005', '2005', '2005', '2005'],
 'country': ['USA', 'USA', 'USA', 'USA'],
 'genre': ['Crime', 'Crime', 'Crime', 'Crime'],
```

```
 'summary': ["Tom Stall, a humble family man and owner of a \n\tpopular
neighborhood restaurant, lives a quiet but \n\tfulfilling existence in the
Midwest. One night Tom \n\tfoils a crime at his place of business and, to his
\n\tchagrin, is plastered all over the news for his \n\theroics. Following this,
mysterious people follow \n\tthe Stalls' every move, concerning Tom more than
\n\tanyone else. As this situation is confronted, more \n\tlurks out over where
all these occurrences have \n\tstemmed from compromising his marriage, family
\n\trelationship and the main characters' former \n\trelations in the process.",
  "Tom Stall, a humble family man and owner of a \n\tpopular neighborhood
restaurant, lives a quiet but \n\tfulfilling existence in the Midwest. One night
Tom \n\tfoils a crime at his place of business and, to his \n\tchagrin, is
plastered all over the news for his \n\theroics. Following this, mysterious
people follow \n\tthe Stalls' every move, concerning Tom more than \n\tanyone
else. As this situation is confronted, more \n\tlurks out over where all these
occurrences have \n\tstemmed from compromising his marriage, family
\n\trelationship and the main characters' former \n\trelations in the process.",
  "Tom Stall, a humble family man and owner of a \n\tpopular neighborhood
restaurant, lives a quiet but \n\tfulfilling existence in the Midwest. One night
Tom \n\tfoils a crime at his place of business and, to his \n\tchagrin, is
plastered all over the news for his \n\theroics. Following this, mysterious
people follow \n\tthe Stalls' every move, concerning Tom more than \n\tanyone
else. As this situation is confronted, more \n\tlurks out over where all these
occurrences have \n\tstemmed from compromising his marriage, family
\n\trelationship and the main characters' former \n\trelations in the process.",
  "Tom Stall, a humble family man and owner of a \n\tpopular neighborhood
restaurant, lives a quiet but \n\tfulfilling existence in the Midwest. One night
Tom \n\tfoils a crime at his place of business and, to his \n\tchagrin, is
plastered all over the news for his \n\theroics. Following this, mysterious
people follow \n\tthe Stalls' every move, concerning Tom more than \n\tanyone
else. As this situation is confronted, more \n\tlurks out over where all these
occurrences have \n\tstemmed from compromising his marriage, family
\n\trelationship and the main characters' former \n\trelations in the
process."],
 'director': []}
```

[7]:
```
df= pd.DataFrame(dict_df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING␣
↪5\Acceso_local_y_Tratamiento_de_Datos\unit␣
↪2_Acceso_y_Procesado_de_Datos_Internos__Ficheros\07_Ficheros_xml_II.ipynb␣
↪Celda 12 line 1
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
↪ONLINE_DS_THEBRIDGE_V/SPRING%205/Acceso_local_y_Tratamiento_de_Datos/
↪unit%202_Acceso_y_Procesado_de_Datos_Internos__Ficheros/07_Ficheros_xml_II.
↪ipynb#X15sZmlsZQ%3D%3D?line=0'>1</a> df= pd.DataFrame(dict_df)
```

```
File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\frame.
 ↪py:733, in DataFrame.__init__(self, data, index, columns, dtype, copy)
    727     mgr = self._init_mgr(
    728         data, axes={"index": index, "columns": columns}, dtype=dtype,␣
 ↪copy=copy
    729     )
    731 elif isinstance(data, dict):
    732     # GH#38939 de facto copy defaults to False only in non-dict cases
--> 733     mgr = dict_to_mgr(data, index, columns, dtype=dtype, copy=copy,␣
 ↪typ=manager)
    734 elif isinstance(data, ma.MaskedArray):
    735     from numpy.ma import mrecords

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\internal
 ↪py:503, in dict_to_mgr(data, index, columns, dtype, typ, copy)
    499     else:
    500         # dtype check to exclude e.g. range objects, scalars
    501         arrays = [x.copy() if hasattr(x, "dtype") else x for x in array]
--> 503 return arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ,␣
 ↪consolidate=copy)

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\internal
 ↪py:114, in arrays_to_mgr(arrays, columns, index, dtype, verify_integrity, typ␣
 ↪consolidate)
    111 if verify_integrity:
    112     # figure out the index, if necessary
    113     if index is None:
--> 114         index = _extract_index(arrays)
    115     else:
    116         index = ensure_index(index)

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\internal
 ↪py:677, in _extract_index(data)
    675 lengths = list(set(raw_lengths))
    676 if len(lengths) > 1:
--> 677     raise ValueError("All arrays must be of the same length")
    679 if have_dicts:
    680     raise ValueError(
    681         "Mixing dicts with non-Series may lead to ambiguous ordering."
    682     )

ValueError: All arrays must be of the same length
```

[8]:
```
df_2 = pd.read_xml("./data/movie.xml")
```

```
C:\Users\victo\AppData\Local\Temp\ipykernel_4232\145867932.py:1: FutureWarning:
Passing literal xml to 'read_xml' is deprecated and will be removed in a future
version. To read from a literal string, wrap it in a 'StringIO' object.
  df_2 = pd.read_xml("./data/movie.xml")
```

```
Traceback (most recent call last):

  File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\IPytho \core\intera
↪py:3526 in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  Cell In[8], line 1
    df_2 = pd.read_xml("./data/movie.xml")

  File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas \io\xml.
↪py:1132 in read_xml
    return _parse(

  File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas \io\xml.
↪py:852 in _parse
    data_dicts = p.parse_data()

  File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas \io\xml.
↪py:556 in parse_data
    self.xml_doc = self._parse_doc(self.path_or_buffer)

  File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas \io\xml.
↪py:647 in _parse_doc
    document = fromstring(

  File src\lxml\etree.pyx:3257 in lxml.etree.fromstring

  File src\lxml\parser.pxi:1916 in lxml.etree._parseMemoryDocument

  File src\lxml\parser.pxi:1803 in lxml.etree._parseDoc

  File src\lxml\parser.pxi:1144 in lxml.etree._BaseParser._parseDoc

  File src\lxml\parser.pxi:618 in lxml.etree._ParserContext._handleParseResultD c

  File src\lxml\parser.pxi:728 in lxml.etree._handleParseResult
```

```
  File src\lxml\parser.pxi:657 in lxml.etree._raiseParseError


  File <string>:1
XMLSyntaxError: Start tag expected, '<' not found, line 1, column 1
```