## 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.

**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 carácter coma. Es 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 carácter 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.

### Fichero csv como fichero de texto

*# Leamos uno de los ficheros csv que hemos visto en sesiones anteriores*

**with** open("./data/df\_liga\_2019.csv","r", encoding**=**"utf8") **as** f:

datos**=**[linea.replace("\n","") **for** linea **in** f]

*# Veamos las primeras 10 filas*

datos[0:10]

Imagen que contiene Texto

Descripción generada automáticamente

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

### Pandas y csv: Lectura

**¿Cómo podemos leer un CSV en Python con Pandas?** [Detalle](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html)

Pero ya lo hemos hecho tantas veces que te será familiar, usando el método read\_csv:

**import** pandas **as** pd

df**=**pd.read\_csv("./data/df\_liga\_2019.csv")

df.head()

Imagen que contiene Calendario

Descripción generada automáticamente

Vamos a ver algunos **parámetros interesantes del read\_csv()**

1. index\_col: indica cuál 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)
3. sep: el separador de los datos, por defecto es coma, pero podría ser otro como veremos en ejemplos posteriores.
4. header: dónde se encuentran los nombres 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]

#### index\_col

df**=**pd.read\_csv("./data/df\_liga\_2019.csv",index\_col**=**"id\_partido")

df

Tabla

Descripción generada automáticamente

#### names

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

df.columns

Imagen que contiene Gráfico

Descripción generada automáticamente

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

Tabla

Descripción generada automáticamente

#### 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:

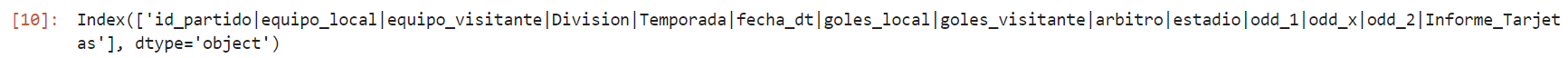
df**=**pd.read\_csv("./data/df\_liga\_2019\_pipe.csv")

df

Tabla

Descripción generada automáticamente

df.columns



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

df**=**pd.read\_csv("./data/df\_liga\_2019\_pipe.csv", sep**=**"|", index\_col**=**"id\_partido")

df

[11]:

Tabla

Descripción generada automáticamente

### Pandas y csv: Escritura

Para escribir un CSV usamos el método to\_csv(). Tienes [Detalle](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_csv.html).

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]

**with** open("./data/df\_ejemplo\_write.csv", "r", encoding **=** "utf8") **as** f:

datos **=** [linea.replace("\n","") **for** linea **in** f]

datos[0:12]

Carta

Descripción generada automáticamente con confianza baja

df = pd.read\_csv("./data/df\_ejemplo\_write.csv", sep = ";")