Tema 3. Ciencia de Datos

La ciencia de datos es una multi técnica utilizada para interpretar datos, relacionarlos y concluir nuevas ideas a fines de mejorar el desempeño o monitorear una variable o parámetro. La ciencia de datos combina múltiples campos, entre los que se incluyen estadísticas, métodos científicos, inteligencia artificial (IA) y análisis de datos, para extraer valor de los datos. Ya que la tecnología moderna ha permitido la creación y almacenamiento de cantidades cada vez mayores de información, los volúmenes de datos se han incrementado. Se estima que el 90% de los datos en el mundo se crearon en los últimos dos años. Por ejemplo, los usuarios de Facebook suben 10 millones de fotos por hora. Muchos datos a analizar son adquiridos por las mismas empresas con el fin de monitorear su producción con el uso de sensores y generar una base de datos para poder procesarla y poder tomar decisiones que generen un cambio positivo en la producción, por ejemplo, de cultivo hidropónico.

La ciencia de datos revela tendencias y genera información que las empresas pueden utilizar para tomar mejores decisiones y crear productos y servicios más innovadores. Quizás lo más importante es que permite que los modelos de aprendizaje automático extraigan conocimientos de las grandes cantidades de datos que se les suministran, evitando así depender principalmente de los analistas empresariales para ver qué pueden descubrir a partir de los datos. Otra ventaja del análisis de datos es poder tener una mirada global de tendencias en algún parámetro, donde en vez de leer miles o millones de datos, tal vez podemos realizar algunas lecturas con solo mirar un gráfico, de los cuales hablamos sobre el final del Episodio 2.

Opinión de un mendocino que se dedica a Data Science en Luján de Cuyo: ¿Cómo ves a Mendoza en 30 años? https://www.youtube.com/watch?v=qeA5QXA2nvA&ab_channel=GabrielConte

Tema 3.1- Archivos CSV:

- Base de datos

La ciencia de datos se enfoca en estudiar, analizar y encontrar correlaciones entre datos guardados de forma sistemática y ordenada, pertenecientes a un mismo contexto, a esta información la llamamos base de datos.

Las bases de datos son el producto de la necesidad humana de almacenar la información, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente. En ese sentido, la aparición de la electrónica y la computación brindó el elemento digital indispensable para almacenar enormes cantidades de datos en espacios físicos limitados, gracias a su conversión en señales eléctricas o magnéticas.

Tipos de bases de datos:

❖ **Según su variabilidad:**

➢ Bases de datos estáticas: datos inmutables, no cambian.

➢ Bases de datos dinámicas: datos que cambian o pueden ir cambiando.

❖ **Según su contenido:**

➢ Bibliográficas: material de lectura con datos ordenados.

➢ De texto completo: documentales, papers, textos históricos, etc.

➢ Directorios: listados con datos como direcciones de correo, números de teléfono, etc.

➢ Especializadas: Bases de datos de información hiperespecializada o técnica, pensadas a partir de las necesidades puntuales de un público determinado que consume dicha información.

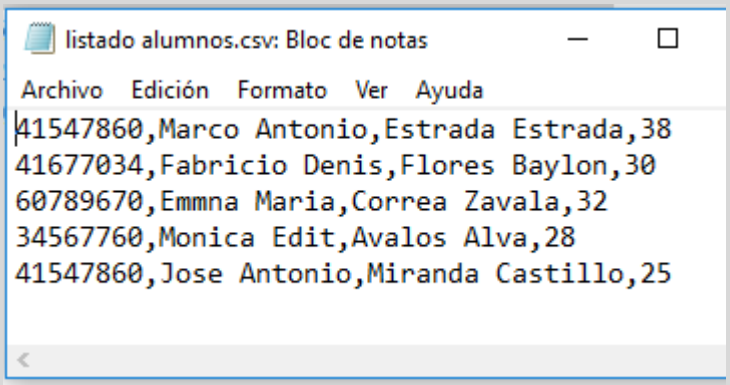
Ejemplos de bases de datos:

- Guías telefónicas
- Censos nacionales, provinciales, municipales o encuestas.
- Bibliotecas públicas
- Personal de una empresa
- Visualizaciones en un video
- Registro de actividades
- Historiales médicos

Las bases de datos suelen guardarse de manera ordenada y sencilla. Los archivos CSV (Comma Separated Value) son un tipo de archivos que contienen sus valores separados por coma (en realidad punto y coma) haciendo una especie de tabla en filas y columnas. Las columnas quedan definidas por cada punto y coma (;), mientras que cada fila se define mediante una línea adicional en el texto (un enter). De esta manera, se pueden crear archivos CSV con gran facilidad. Es por esto que los archivos .csv están asociados directamente a la creación de tablas de contenido.

Un archivo CSV suele identificarse con el programa Microsoft Excel, el cual se basa en cuadrículas que conforman una tabla en filas y columnas. Lo más común es leer archivos CSV desde Excel, ya que el programa identifica automáticamente los separadores y forma la tabla sin tener que hacer nada por nuestra parte y son muy fáciles de abrir/leer/escribir/crear desde python. La facilidad de pasar de una tabla a CSV y viceversa, junto al poco espacio de almacenamiento y exigencias de cómputo que requieren, hace que estos archivos sean prácticamente universales.

	A	B	C	D
1	indice_tiempo,total,colectivo,lancha,subte,tren			
2	2020-01-01,966457,907421,40,72180,112267			
3	2020-01-02,3879833,3564004,94,511136,719721			
4	2020-01-03,4213711,3886005,106,536129,779512			
5	2020-01-04,2899714,2728108,129,232491,461000			
6	2020-01-05,1785016,1687791,94,127770,203694			
7	2020-01-06,4160267,3817677,111,543433,772203			
8	2020-01-07,4195485,3855716,61,553337,774509			
9	2020-01-08,4113127,3775151,80,546992,776612			
10	2020-01-09,4114983,3777667,93,551974,771768			
11	2020-01-10,4261741,3930629,122,551782,802904			
12	2020-01-11,2872425,2709650,100,233528,442738			
13	2020-01-12,1770824,1673993,86,128084,212369			
14	2020-01-13,4110363,3776600,103,537425,767740			
15	2020-01-14,3850434,3533051,82,519193,711038			
16	2020-01-15,3845908,3538310,61,517225,662890			
17	2020-01-16,4217823,3873449,91,561236,794638			
18	2020-01-17,4316121,3983349,124,541901,813476			
19	2020-01-18,2924307,2764358,111,197352,477500			
20	2020-01-19,1779502,1692829,101,101496,210334			
21	2020-01-20,3905584,3627245,86,428750,701578			
22	2020-01-21,3878022,3593730,83,433795,700158			
23	2020-01-22,4158654,3860207,88,452305,761511			
24	2020-01-23,4105439,3809202,77,450568,755454			
25	2020-01-24,4135504,3850127,148,440591,760993			



Ejemplo de un archivo CSV abierto con Excel

¿Cómo se crea un CSV?

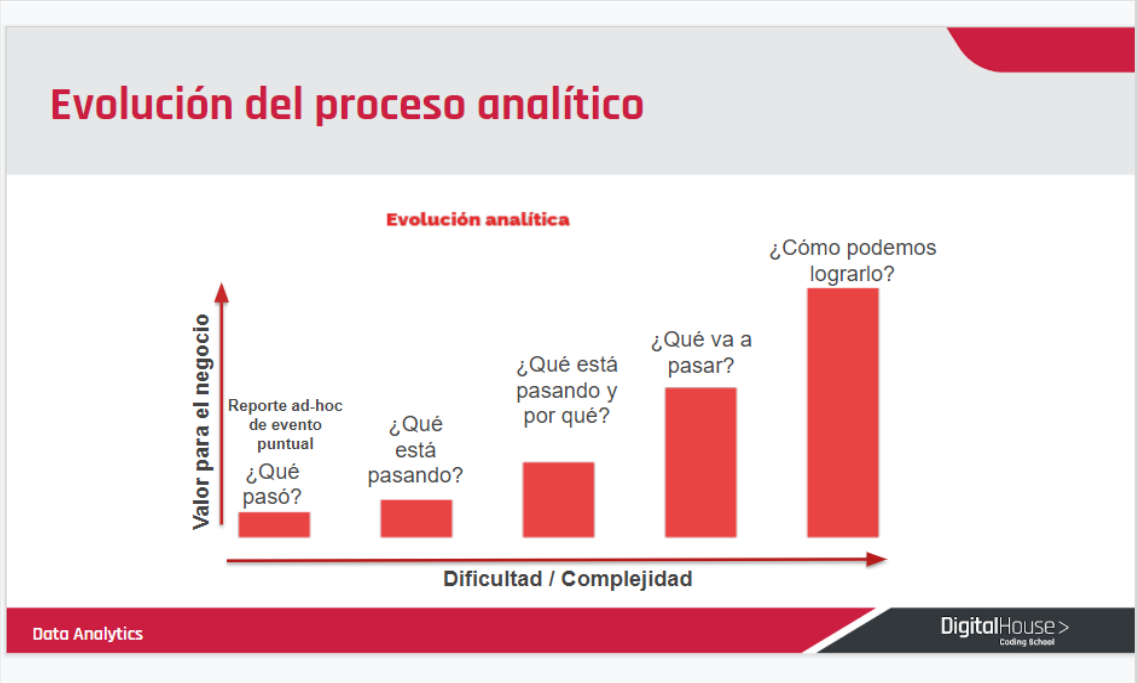
Los archivos CSV normalmente son creados por programas que manejan grandes cantidades de datos. Son una forma conveniente de exportar datos de hojas de cálculo y bases de datos, así como importarlos o usarlos en otros programas. Por ejemplo, puede exportar los resultados de un programa de adquisición de datos de un sensor en una planta industrial o proyecto de campo, a un archivo CSV y luego importarlo a una hoja de cálculo para analizar los datos, generar gráficos para una presentación o preparar un informe para su publicación.

Las bases de datos suelen llamarse dataSets, csv o dataFrame. Existen páginas dedicadas a recopilar datasets como datos.gob.ar o destinadas a la búsqueda de los mismos como Google Dataset Search.

¿Qué es Data Analytics?

Se trata de un conjunto de técnicas y procesos cuantitativos y cualitativos usados para la toma de decisiones, con el objetivo de mejorar un proyecto o negocio a través del conocimiento extraído de los datos.

El estadístico John Tukey definió el análisis de datos en 1961 de esta manera: "*Son los procedimientos para analizar datos, las técnicas para interpretar los resultados de dichos procedimientos y las formas de planear la recolección de datos para hacer el análisis más fácil, más preciso o más exacto.*"



¿Qué aspectos debemos fortalecer?

Los datos se están convirtiendo en la nueva fuente de combustible del mundo, por eso te compartimos algunas de las formas en que el análisis de datos se centra:

- Identificar tendencias y patrones.

- Para buscar nuevas oportunidades.

- Determinar posibles riesgos y beneficios.

- Elaborar una estrategia de acción.

Para el trabajo en ciencia de datos la fusión, entre creatividad y datos, sirve para producir resultados más precisos al potenciar los procesos creativos permite desarrollar excelentes estrategias de análisis. Sin embargo, para lograr una fusión perfecta, es necesario que haya armonía entre los datos y la creatividad. Es por eso que te proponemos trabajar de manera integrada las habilidades blandas y tecnológicas a partir del siguiente material.

Tema 4. Biblioteca Pandas para Python:

La biblioteca Pandas da las herramientas necesarias para leer y escribir en archivos CSV de manera sencilla. Diseñada para trabajar con cualquier archivo CSV y permite el procesamiento de datos de una manera rápida y fácil.

Esta biblioteca ya viene instalada por defecto en nuestro IDE de Python (Spyder), para usarla colocamos:

```
import pandas
```

En el caso de no tenerla instalada deberemos colocar en la terminal de Spyder:

```
!pip install pandas
```

Tema 4.1- Leer archivos CSV con Pandas:

La lectura de un archivo .csv se realiza utilizando nombre_de_la_lista=pandas.read_csv("nombre archivo o url")

El archivo CSV se abre como un archivo de texto y crea un objeto de archivo (llamado datos en el ejemplo que sigue).

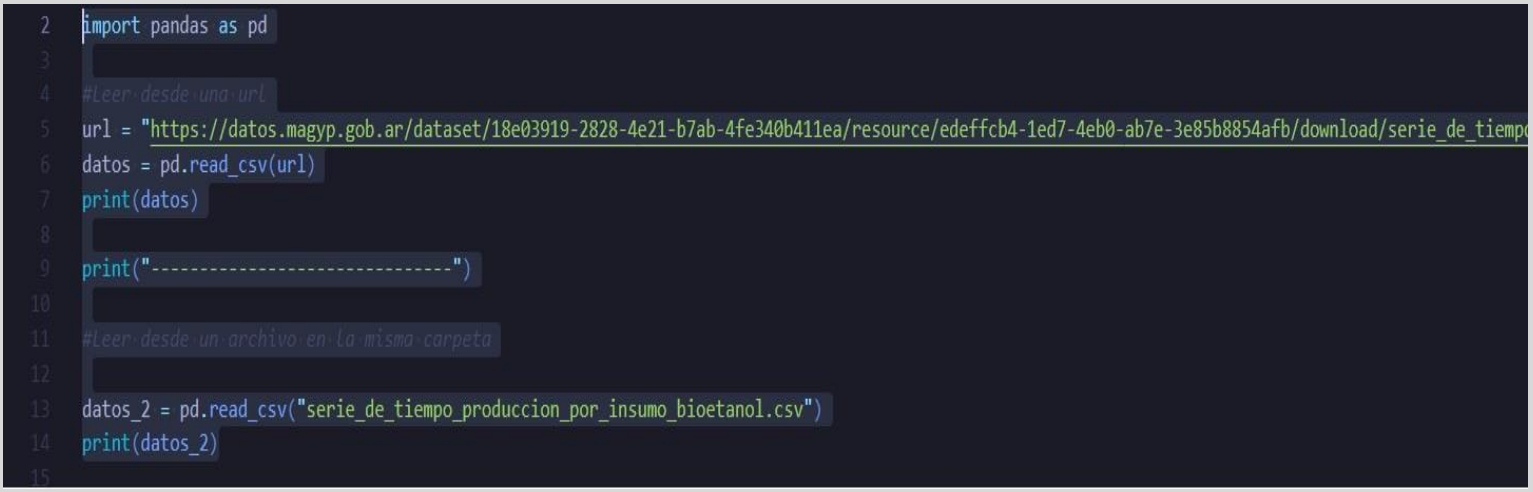
Hay dos posibilidades, leer un csv desde internet o desde la misma computadora.

Leer csv desde un link de internet:

```
import pandas as pd

url = "https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-4fe340b411ea/resource/edeffcb4-1ed7-4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"

datos = pd.read_csv(url)

print(datos)
```

Leer csv desde un archivo guardado en el mismo directorio -la misma carpeta- desde donde ejecutamos el código:

```
import pandas as pd

datos = pd.read_csv("serie_de_tiempo_produccion_por_insumo_bioetanol.csv")

print(datos)
```



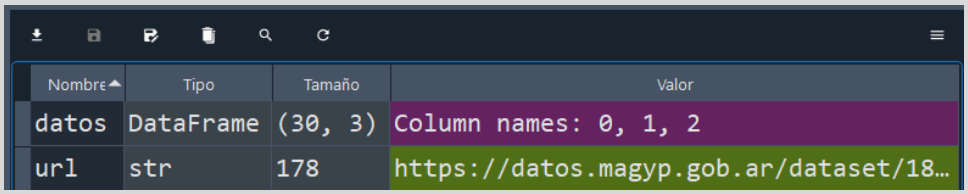
descargar csv:

https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-4fe340b411ea/resource/edeffcb4-1ed7-4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv

Resultado:

```
In [4]: datos
Out[4]:
                0                 1      2
0   indice_tiempo  cania_de_azucar   maiz
1         2017-01             37794  46551
2         2017-02             35607  44489
3         2017-03             38696  44100
4         2017-04             35355  45218
5         2017-05             44053  40222
6         2017-06             51159  41800
7         2017-07             53924  43396
8         2017-08             55711  48852
9         2017-09             55345  47483
10        2017-10             60415  50850
11        2017-11             48724  47841
```

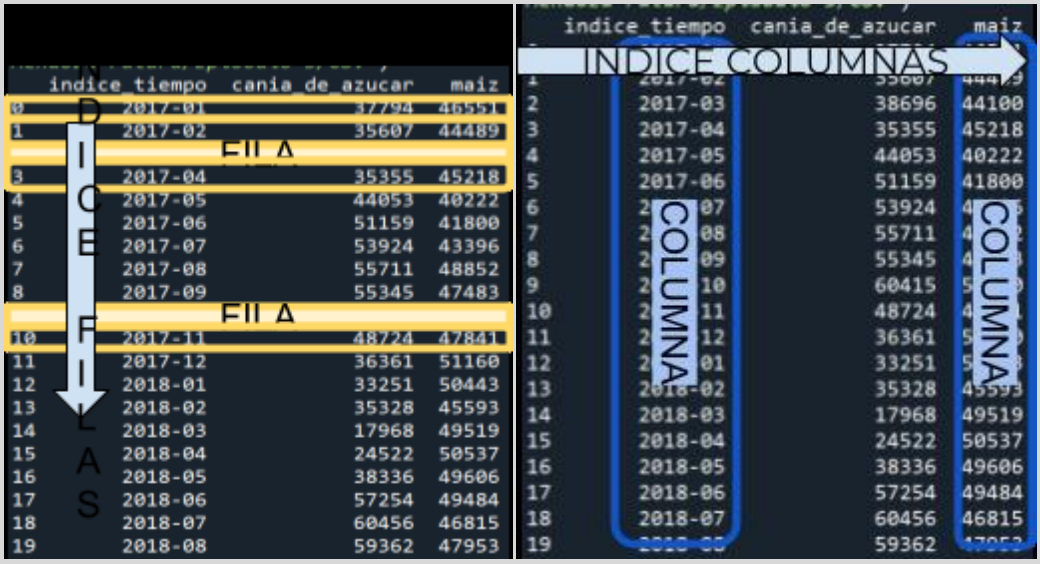
Se crea un tipo de dato "DataFrame" -similar a una tabla- que tiene filas y columnas:



Nombre	Tipo	Tamaño	Valor
datos	DataFrame	(30, 3)	Column names: 0, 1, 2
url	str	178	https://datos.magyp.gob.ar/dataset/18…

Se indica la cantidad de filas y luego de columnas, este DataFrame tiene 29 filas y 3 columnas.

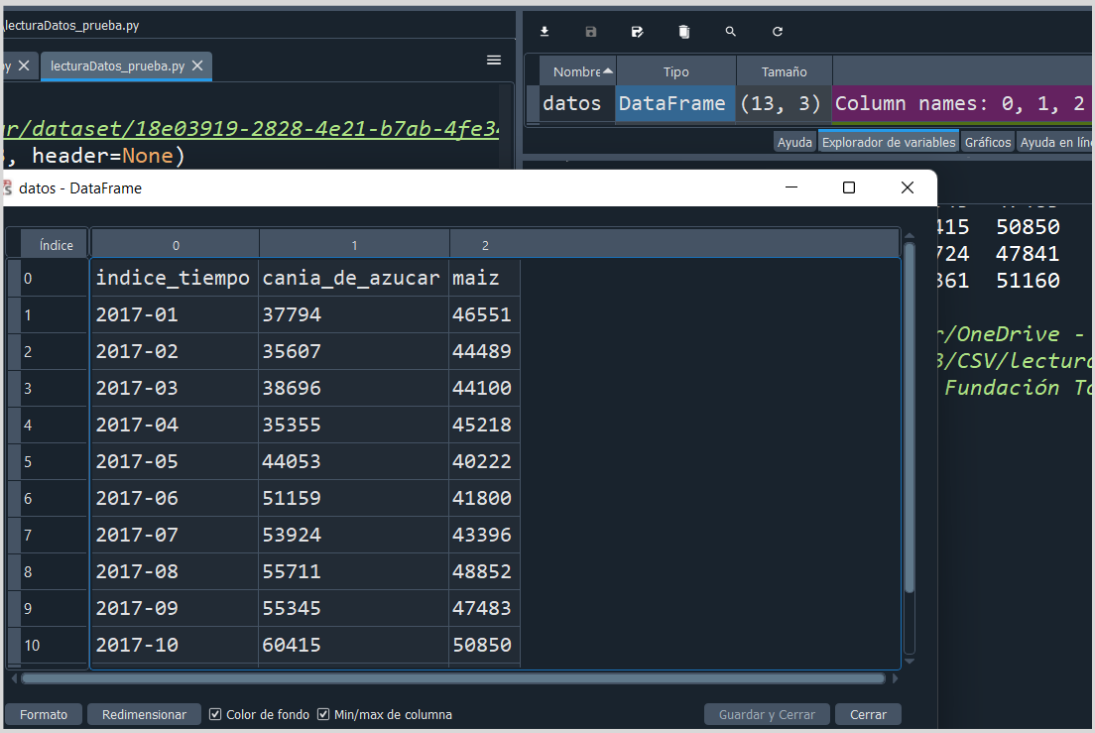
Son 29 datos para 29 meses desde el 2017 al 2019. Son 3 columnas, tiempo y producción de bioetanol a partir de caña de azúcar y maíz expresados en metros cúbicos.



El .csv es perteneciente a los datos públicos que el gobierno recopila y sube a www.datos.gob.ar. En este caso se ha estudiado la cantidad total de metros cúbicos de bioetanol producidos a partir de la caña de azúcar y maíz desde marzo del 2017 a mayo del 2019.

Los nombres de las columnas son: indice_tiempo, cania_de_azucar y maiz. Para imprimir las columnas guardadas en el objeto datos simplemente hay que tratarla como a una lista: la columna indice_tiempo es datos[0], la columna cania_de_azucar es datos[1] y la columna cania_de_azucar es datos[2].

Se observa esto mismo dentro del explorador de variables, en datos, que es un objeto del tipo DataFrame:



Un DataFrame es una estructura de datos con dos dimensiones en la cual se puede guardar datos de distintos tipos (como caracteres, enteros, valores de punto flotante, factores y más) en columnas. Es similar a una hoja de cálculo o excel o csv. Un DataFrame siempre tiene un índice (con inicio en 0). El índice se refiere a la posición de un elemento en la estructura de datos, al igual que en listas.

Parámetros de read_csv():

DataFrame = pd.read_csv(url, header=0, names=['nombre1', 'nombre2', 'nombre3'])

read_csv() es una función encargada de leer un csv desde un archivo local o url que se coloque en el primer parámetro de la función. El segundo parámetro es header o "cabecera" y se utiliza para indicar en qué fila del dataset está la cabecera o nombres que clasifican las columnas, así las elimina porque por ejemplo no podemos graficar o promediar un título, normalmente header está en 0, si colocamos header = None, no elimina la fila de los títulos (haremos header = 0 casi siempre).

El tercer parámetro es una lista que indica los nombres de las columnas para el DataFrame, por ej: names=['Fecha','Caña Azúcar','Maíz']

Se le cambia/agrega nombre a las columnas con el parámetro names=[ ]. Esto es para que queden los datos ordenados y cada columna clasificada con un nombre:

```
import pandas as pd

url = "https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-4fe340b411ea/resource/edeffcb4-1ed7-

4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"

datos = pd.read_csv(url, header=0, names=['Fecha','Caña Azúcar','Maíz'])

print(datos)
```

Observar en el resultado que ahora hay una fila más.

Con nrows establecemos la cantidad de filas o datos que queremos analizar:

```
import pandas as pd
url = "https://archivos-datos.transporte.gob.ar/upload/Sube/total-usuarios-por-dia.csv"
datos2017 = pd.read_csv(url,nrows=13, header=None)
print(datos2017)
```

Resultado: imprime solo los 12 meses del año 2017.

Sumando valores de las columnas o filas

Para poder sumar los valores de cada columna o filas existe la función .sum(), hay que asegurarse que los tipos de datos sean enteros mediante el uso de la función .dtypes. Como son números enteros podemos operarlos matemáticamente. Caso contrario habría que hacer una conversión a números. Con el método .sum() podemos realizar la sumatoria de las filas que queramos o de todas las filas juntas:

```
import pandas as pd
url ="https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-4fe340b411ea/resource/edeffcb4-1
ed7-4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"
datos = pd.read_csv(url,header=0, names=['Fecha','CaniaAzucar','Maiz'])
print(datos.sum()) #hace sumatoria de todas las columnas (las fechas no las puede sumar porque siguen siendo str)
print("\n")
print(f"Total de bioetanol producido con caña de azucar: {datos['CaniaAzucar'].sum()} m3")
print(f"Total de bioetanol producido con maíz: {datos['Maiz'].sum()} m3")
```

```
#Supongamos que tenemos el siguiente DataFrame de panda
import pandas as pd
import numpy as np
# Creamos el DataFrame
df = pd.DataFrame({'rating': [90, 85, 82, 88, 94, 90, 76, 75, 87, 86],
                   'points': [25, 20, 14, 16, 27, 20, 12, 15, 14, 19],
                   'assists': [5, 7, 7, 8, 5, 7, 6, 9, 9, 5],
                   'rebounds': [np.nan, 8, 10, 6, 6, 9, 6, 10, 10, 7]})
```

#Podemos encontrar la suma de la columna titulada «puntos» usando la siguiente sintaxis:

```
df['points'].sum()
```

#resultado de la columna points = 182

#La función sum () también excluirá NA de forma predeterminada. Por ejemplo, si encontramos la suma de la columna «rebotes», el primer valor de «NaN» simplemente se excluirá del cálculo:

```
df['rebounds'].sum()
```

#resultado = 72.0

#Podemos encontrar la suma de **varias** columnas usando la siguiente sintaxis:

```
# Calculamos la suma para ciertas filas del DataFrame

df[['rebounds', 'points']].sum()

# Resultados: rebounds 72.0, points 182.0, dtype: float64


# Encontramos la suma para todas las columnas del DataFrame

df.sum()
```

#Resultado: rating 853.0, points 182.0, assists 68.0, rebounds 72.0, dtype: float64

Para analizar la información y llegar a conclusiones se deben realizar gráficas con la ayuda de librerías como matplotlib y numpy, algo que repasaremos pronto.