**pANDAS**

Pandas es una biblioteca muy popular para el análisis y manipulación de datos en Python.

Documentación oficial de Pandas para conocer más detalles y ejemplos: <https://pandas.pydata.org/docs/>

Aquí se enumeran las principales características y funcionalidades que ofrece:

**Estructuras de datos**: pandas proporciona dos estructuras de datos principales: Series y DataFrame. Una Serie es un array unidimensional con etiquetas, mientras que un DataFrame es una estructura tabular bidimensional similar a una tabla de base de datos.

**Lectura y escritura de datos**: pandas permite leer datos desde una variedad de fuentes, como archivos CSV, Excel, bases de datos SQL, archivos JSON, entre otros. También es posible escribir los datos en diferentes formatos.

**Manipulación de datos**: pandas ofrece una amplia gama de herramientas para manipular y transformar datos. Puedes filtrar, ordenar, agrupar, fusionar y unir datos, así como realizar cálculos estadísticos y operaciones matemáticas en ellos.

**Indexación y selección de datos**: pandas proporciona una variedad de métodos para seleccionar y acceder a datos dentro de las estructuras. Puedes indexar por etiquetas, posiciones, condiciones booleanas y utilizar expresiones lógicas y aritméticas para filtrar los datos.

**Limpieza y preprocesamiento de datos**: pandas facilita la detección y manejo de valores faltantes, datos duplicados, valores atípicos y otros problemas comunes en los conjuntos de datos. También puedes aplicar transformaciones y normalizaciones a tus datos.

**Operaciones numéricas y estadísticas**: pandas incluye funciones y métodos para realizar cálculos estadísticos y operaciones matemáticas en los datos, como promedios, sumas, desviaciones estándar, correlaciones y más.

**Visualización de datos**: pandas se integra con bibliotecas de visualización como matplotlib y ofrece métodos simples para crear gráficos y visualizaciones directamente desde los datos en un DataFrame o Serie.

**Integración con otras bibliotecas**: pandas se integra bien con otras bibliotecas populares de Python, como numpy, scikit-learn y matplotlib, lo que permite un análisis y visualización de datos más potente y eficiente.

# 

# Ejemplos de uso

A continuación, te presento algunas de las principales funcionalidades de Pandas junto con ejemplos:

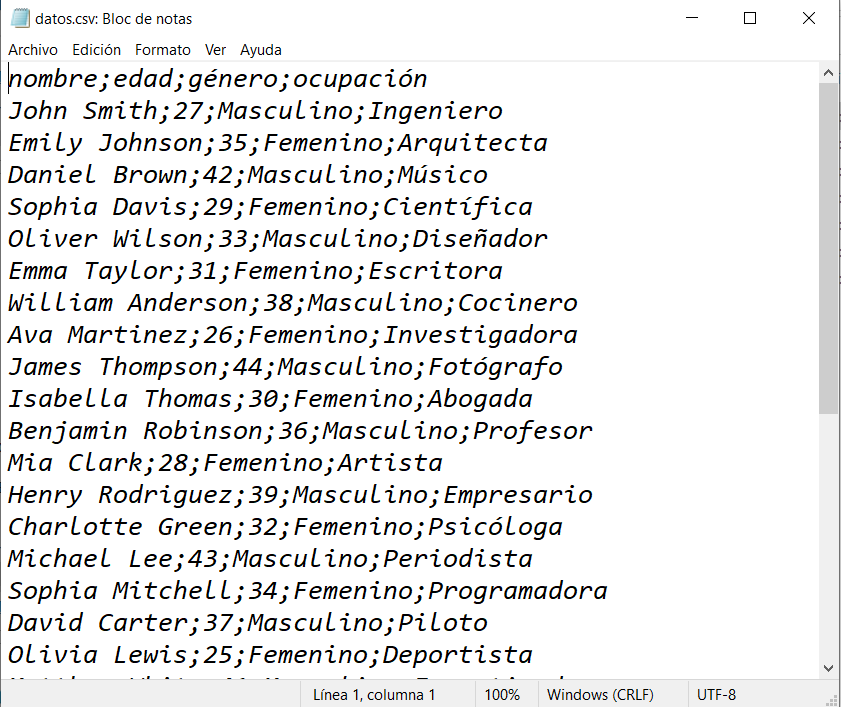
## Creación y manipulación de DataFrames



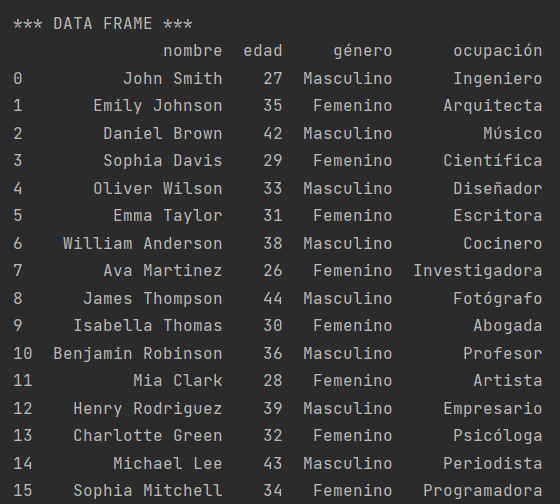
import pandas as pd  
  
  
# Crear un DataFrame a partir de un diccionario  
datos = {'Nombre': ['Benito', 'Andrea', 'Floro', 'María'],  
 'Edad': [25, 30, 35, 45],  
 'Ciudad': ['Madrid', 'Barcelona', 'Sevilla', 'Valencia']}  
  
  
  
df = pd.DataFrame(datos)  
  
# Mostrar el DataFrame  
print('\n\*\*\* DATA FRAME \*\*\*')  
print(df)  
  
# Filtrar filas basado en una condición  
print('\n\*\*\* DATA FRAME FILTRADO \*\*\*')  
df\_filtrado = df[df['Edad'] > 30]  
print(df\_filtrado)  
  
# Ordenar el DataFrame por una columna  
print('\n\*\*\* DATA FRAME ORDENADO POR EDAD \*\*\*')  
df\_ordenado = df.sort\_values('Edad')  
print(df\_ordenado)

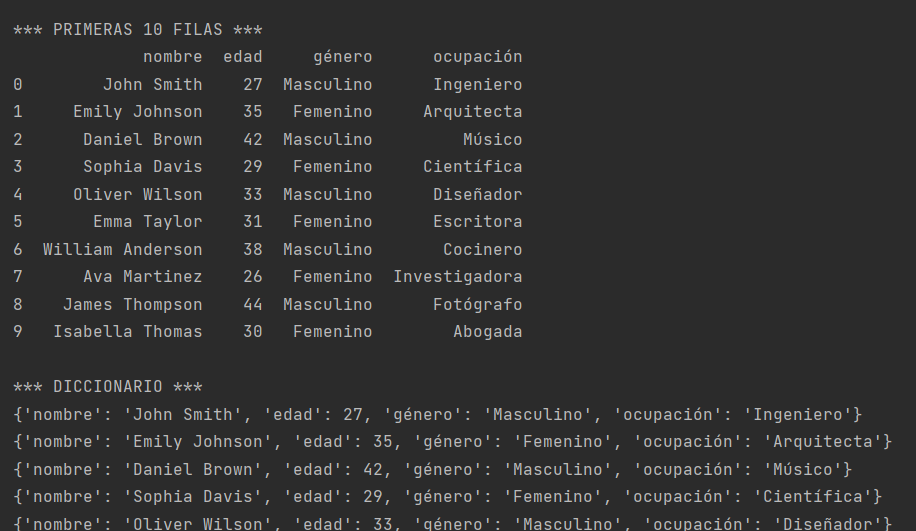
## Lectura de archivos csv

Haciendo uso del siguiente archivo csv



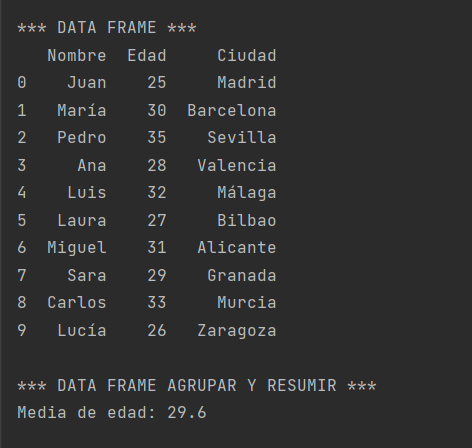
Desarrollar una aplicación que nos genere un Dataframe y podamos consultar los datos:





import pandas as pd  
  
# Cargar datos desde un archivo CSV  
df = pd.read\_csv('datos.csv', delimiter=';')  
  
  
# Mostrar el DataFrame  
print('\n\*\*\* DATA FRAME \*\*\*')  
print(df)  
  
print('\n\*\*\* PRIMERAS 10 FILAS \*\*\*')  
print(df.head(10))  
  
  
print('\n\*\*\* DICCIONARIO \*\*\*')  
'''  
Convertir el DataFrame en un diccionario de registros  
Cada registro es un diccionario con:  
\* claves correspondientes a los nombres de las columnas  
\* valores correspondientes a los datos de cada fila.  
'''  
diccionario = df.to\_dict(orient='records')  
  
for registro in diccionario:  
 print(registro)

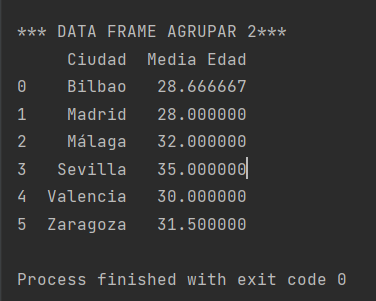
## Calcula el promedio de una columna específica en un DataFrame



import pandas as pd  
  
  
# Crear un DataFrame a partir de un diccionario  
datos0 = {'Nombre': ['Juan', 'María', 'Pedro', 'Ana', 'Luis', 'Laura', 'Miguel', 'Sara', 'Carlos', 'Lucía'],  
 'Edad': [25, 30, 35, 28, 32, 27, 31, 29, 33, 26],  
 'Ciudad': ['Madrid', 'Barcelona', 'Sevilla', 'Valencia', 'Málaga', 'Bilbao', 'Alicante', 'Granada', 'Murcia', 'Zaragoza']}  
  
df = pd.DataFrame(datos0)  
  
# Mostrar el DataFrame  
print('\n\*\*\* DATA FRAME \*\*\*')  
print(df)  
  
  
# Agrupar y resumir datos  
print('\n\*\*\* DATA FRAME AGRUPAR Y RESUMIR \*\*\*')  
df\_agrupado = df['Edad'].mean()  
print('Media de edad:', df\_agrupado)

## Agrupa un DataFrame por una columna y muestra la cantidad de registros en cada grupo.





# Day 17 - 21/06/2023  
# Ejercicios Libreria Panda.docx  
# pip install panda  
  
*'''  
Ejercicio 5  
Agrupa un DataFrame por una columna y muestra la cantidad de registros  
en cada grupo.  
'''*import pandas as pd  
  
  
# Crear un DataFrame a partir de un diccionario  
datos0 = {'Nombre': ['Juan', 'María', 'Pedro', 'Ana', 'Luis', 'Laura', 'Miguel', 'Sara', 'Carlos', 'Lucía'],  
 'Edad': [25, 30, 35, 30, 32, 27, 31, 29, 33, 30],  
 'Ciudad': ['Madrid', 'Bilbao', 'Sevilla', 'Valencia', 'Málaga', 'Bilbao', 'Madrid', 'Bilbao', 'Zaragoza', 'Zaragoza']}  
  
df = pd.DataFrame(datos0)  
  
  
# Mostrar el DataFrame  
print('\n\*\*\* DATA FRAME: SORTED \*\*\*')  
print(df.sort\_values('Ciudad'))  
  
  
# Filtrar filas basado en una condición  
# Agrupar  
print('\n\*\*\* DATA FRAME AGRUPAR 1\*\*\*')  
df\_agrupado = df.groupby('Ciudad').size()  
print(df\_agrupado)  
  
print('\n\*\*\* DATA FRAME AGRUPAR 2\*\*\*')  
#reset\_index: La función ha devuelto el DataFrame con un nuevo índice  
df\_agrupado = df.groupby('Ciudad')['Edad'].mean().reset\_index(name='Media Edad')  
print(df\_agrupado)