



ADVANCE
Consortium



ATENEA 2025 BOOTCAMPS



Identificación Material

Bootcamp	Laboratorio de IA y Cloud Computing - Intermedio
Módulo	Desarrollo de soluciones basadas en datos
Eje Temático	Librerías Básicas Análisis de Datos

Análisis de Datos

Python . Librerías Básicas

MAPA DE CONTENIDOS

ANÁLISIS DE DATOS - PYTHON - PARTE I

01



NUMPY

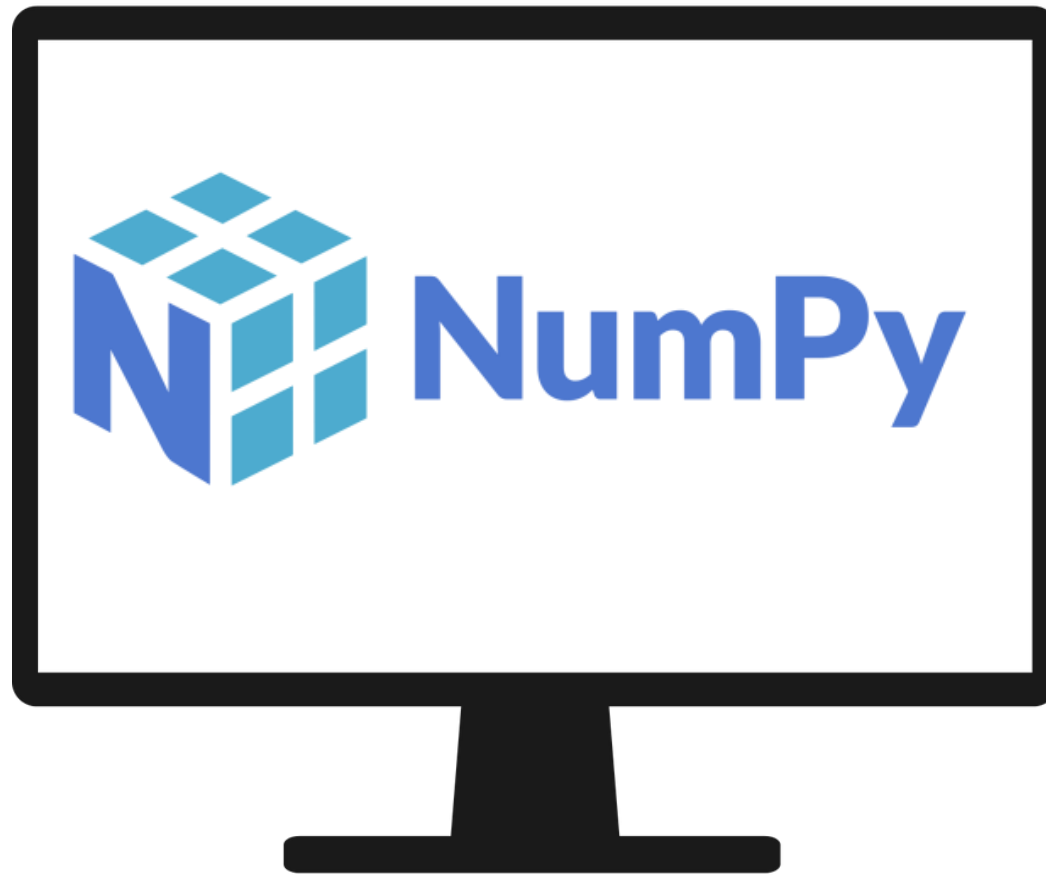
02



PANDAS



Universidad Tecnológica de Bolívar



NumPy es una biblioteca en Python que proporciona una estructura de datos llamada **array**, que es similar a las listas en Python, pero más eficiente y está optimizada para operaciones numéricas. Esta biblioteca también proporciona una amplia gama de funciones para trabajar con estos arrays, lo que la hace extremadamente útil para el procesamiento numérico y científico.



- **Arrays N-dimensionales:** permite crear arrays de cualquier dimensión, el procesamiento de los arrays se realiza mucho más rápido (hasta 50 veces más) que las listas.
- **Operaciones vectorizadas:** NumPy proporciona operaciones vectorizadas, lo que significa que las operaciones se aplican de manera eficiente a todos los elementos de un array sin necesidad de usar bucles explícitos.
- **Funciones matemáticas y de álgebra lineal:** NumPy incluye una amplia variedad de funciones matemáticas y de álgebra lineal, como funciones trigonométricas, exponenciales, operaciones de álgebra lineal (por ejemplo, multiplicación de matrices, cálculo de determinantes, etc.).
- **Generación de números aleatorios:** NumPy tiene funciones para generar números aleatorios de diversas distribuciones estadísticas.
- **Integración con otras bibliotecas:** NumPy es una base fundamental para muchas otras bibliotecas de Python utilizadas en ciencia de datos, aprendizaje automático, procesamiento de imágenes, y más. Por ejemplo, bibliotecas como Pandas y SciPy están construidas sobre NumPy.

- **Paso No.1:** Nos ubicamos en la ruta de la carpeta de Python.

Ejemplo: C:\Users\USUARIO\AppData\Local\Programs\Python\Python311

- **Paso No.2:** Abrimos **cmd** con esa ruta y ejecutamos el comando PIP:

py -m pip install numpy

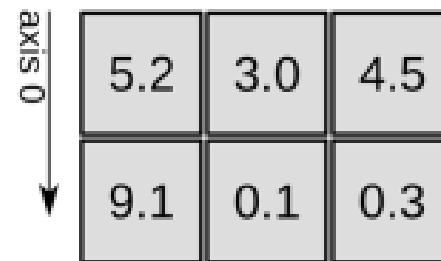
- Un array es una estructura de datos de un mismo tipo organizada en forma de tabla o cuadrícula de distintas dimensiones.

1D array



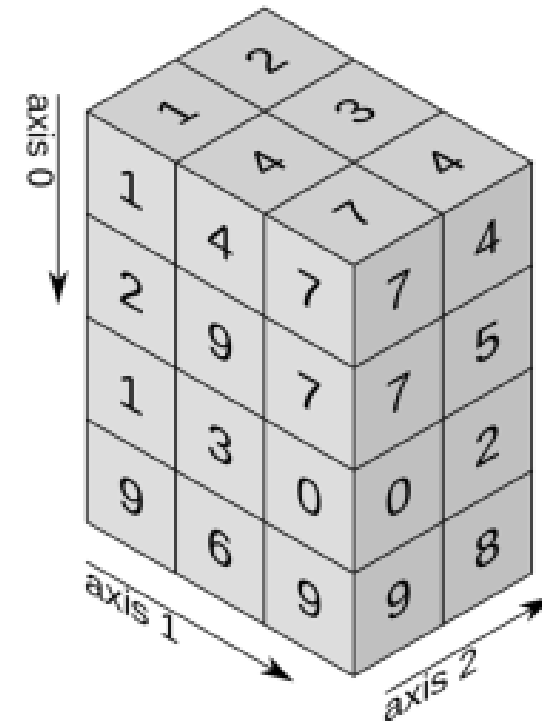
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

- **np.array(lista):** Crea un array a partir de la lista o tupla y devuelve una referencia a él. El número de dimensiones del array dependerá de las listas o tuplas anidadas en lista.
- Para una lista de valores se crea un array de una dimensión, también conocido como **vector**.
- Para una lista de listas de valores se crea un array de dos dimensiones, también conocido como **matriz**.
- Para una lista de listas de listas de valores se crea un array de tres dimensiones, también conocido como **cubo**.
- Y así sucesivamente. No hay límite en el número de dimensiones del array más allá de la memoria disponible en el sistema.
- **Los elementos de la lista o tupla deben ser del mismo tipo.**

Ejemplo:

```
import numpy as np  
a1 = np.array([1, 2, 3])  
print(a1)
```

Ejemplo: Creación de una matriz bidimensional

```
import numpy as np  
a2 = np.array([[1, 2, 3], [4, 5, 6]])  
print(a2)
```



Dimensión 1

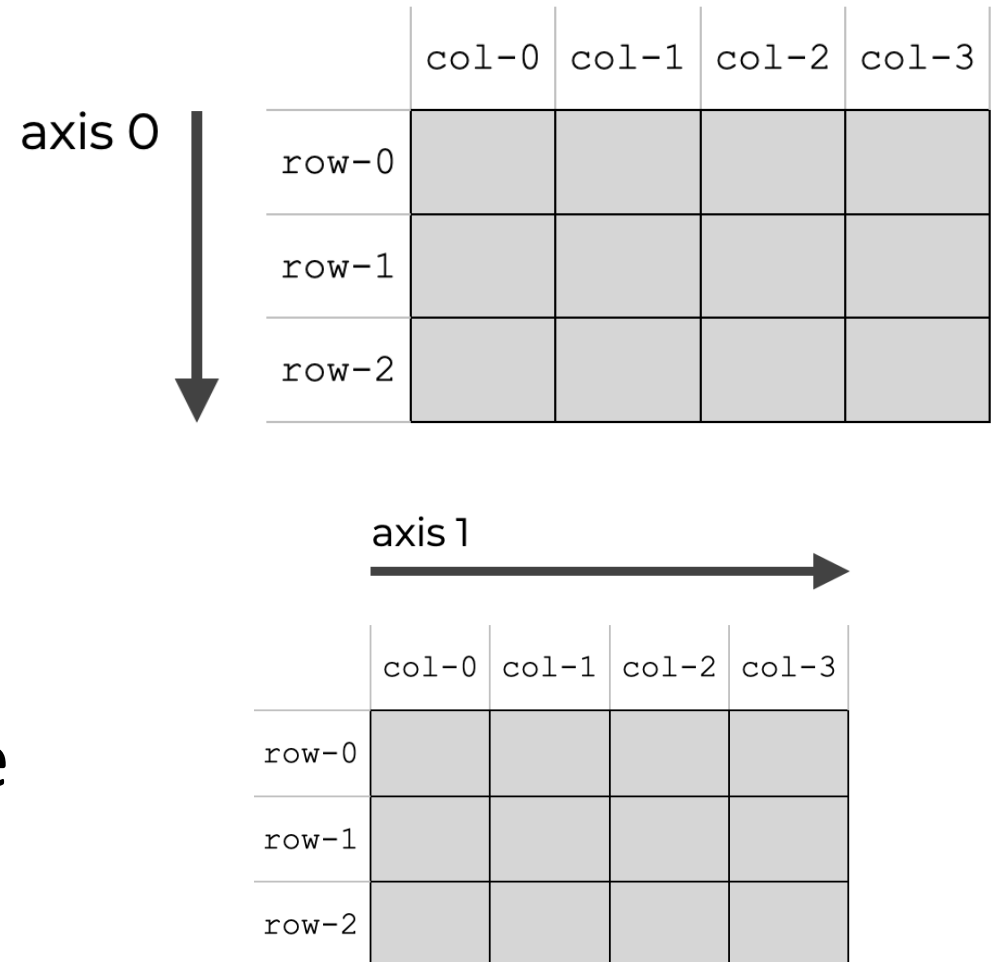


Dimensión 2

Ejemplo:

```
import numpy as np
matriz = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])
```

#axis=0, significa que la operación se
#realiza columna por columna
np.sum(matriz, **axis=0**)



Programa: Creación de arreglos con Numpy

```
import numpy as np

# Creación de un arreglo de una dimensión: Vector
vector=np.array([1,2,3])
print("Vector: ")
print (vector)

# Creación de un arreglo de una dimensión: Vector
print("\nMatriz ")
matriz=np.array([[1,2,3],[4,5,6]])

print (matriz)

# Creación de un arreglo de una dimensión: Vector
cubo=np.array([[1,2,3],[4,5,6],[7,8,9]])
print("\nCubo: ")
print (cubo)
```

Otras funciones útiles que permiten generar arrays son:

- **np.empty(dimensiones):** Crea y devuelve un array vacío con las dimensiones especificadas en el paréntesis.
- **np.zeros(dimensiones):** Crea y devuelve un array con las dimensiones especificadas en el paréntesis cuyos elementos son todos ceros.
- **np.ones(dimensiones):** Crea y devuelve un array con las dimensiones especificadas en el paréntesis cuyos elementos son todos unos.
- **np.full(dimensiones, valor):** Crea y devuelve un array con las dimensiones especificadas y cuyos elementos son todos **valor**.
- **np.arange(inicio, fin, salto):** Crea y devuelve un array de una dimensión cuyos elementos son la secuencia desde **inicio** hasta **fin** tomando valores cada **salto**. Ej:
`np.arange(1,11,2)` #Devuelve array([1, 3, 5, 7, 9])

Programa: Creación de arreglos con Numpy

Casos especiales:

```
import numpy as np

#Creación de arreglo vacío y llenado con determinada información
arreglo1=np.empty([2,2],dtype=int)
arreglo1[0,0]=10
arreglo1[0,1]=15
arreglo1[1,0]=20
arreglo1[1,1]=25
print("Arreglo creado vacío y llenado con información particular:")
print(arreglo1)

#Creación de arreglo con ceros
arreglo2=np.zeros([2,2])
print("\nArreglo creado con ceros:")
print(arreglo2)

#Creación de arreglo con unos
arreglo3=np.ones([2,2])
print("\nArreglo creado con unos:")
print(arreglo3)

#Creación de arreglo y llenado con un valor determinado
arreglo4=np.full([2,2],4)

print("\nArreglo creado con el valor determinado(4):")
print(arreglo4)

#Creación de arreglo con un rango de valores
arreglo5=np.arange(2,20,2)
print("\nArreglo creado ccon el rango de valores, de 2 a 18 con incremento de 2:")
print(arreglo5)
```

- **shape:** Determina la forma de un array **Ej:** `print(a3.shape)`
- **max():** Encuentra el máximo valor de un arreglo **Ej:** `print(array.max())`
- **argmax():** Encuentra el índice del máximo valor dentro de un arreglo **Ej:** `print(array.argmax())`
- **min():** Encuentra el mínimo valor de un arreglo **Ej:** `print(array.min())`
- **argmin():** Encuentra el índice del mínimo valor dentro de un arreglo **Ej:** `print(array.argmin())`
- **mean():** Retorna el promedio de los valores del array.
- **sum():** Retorna la suma de los valores del array.
- **np.where(condition, x, y):** Devuelve una nueva matriz basada en una condición aplicada a cada elemento de una matriz, **X** es el valor que se mostrará en las posiciones que coincidan con la condición y **Y** es el valor que se mostrará en las posiciones que NO coincidan con la condición. **Ej:** `Array = np.array([1, -2, -3, 4, 5])`
`condition = Array < 0`
`result = np.where(condition, 0, Array)`
`print(result)`

Programa: Métodos con NumPy:

```
import numpy as np

matriz=np.array([[1,2,3],[4,5,6],[7,8,9]])
print (matriz)
#Dimesiones del array
print("Dimensiones: ",matriz.shape)

vector=np.array([2,1,3,5,7,9,8])
print("\nVector: ",vector)
#Máximo del array
print("Máximo: ",vector.max())
#Ubicación máximo (Índice o posición)
print("Ubicación del Máximo: ",vector.argmax())
#Mínimo del array
print("Mínimo: ",vector.min())
#Ubicación mínimo (Índice o posición)
print("Ubicación del Mínimo: ",vector.argmin())
# Promedio del array
print("Media: ",vector.mean())
# Suma del array
print("Suma: ",vector.sum())

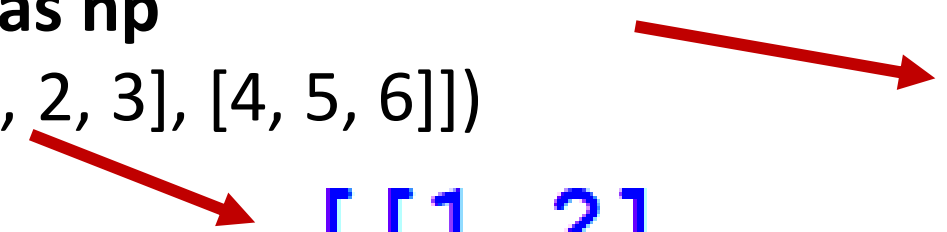
#Condiciones en array
Array = np.array([1, -2, -3, 4, 5])
condicion=Array<0
resultado=np.where(condicion,0,Array)
print("\nCondición valores menores que cero, reemplazados por cero")
print("Arreglo: ",Array)
print("Arreglo con condiciones: ",resultado)
```

- Para acceder a los elementos contenidos en un array se usan índices al igual que para acceder a los elementos de una lista, pero indicando los índices de cada dimensión separados por comas.
- Al igual que para listas, los índices de cada dimensión comienzan en 0.
- También es posible obtener subarrays con el operador **dos puntos** : indicando el índice inicial y el siguiente al final para cada dimensión, de nuevo separados por comas.

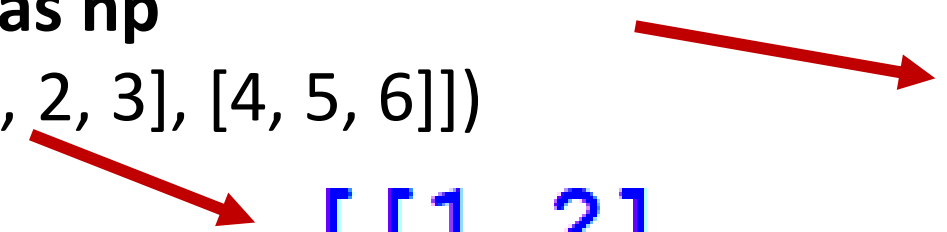
```
import numpy as np
```

```
a = np.array([[1, 2, 3], [4, 5, 6]])
```

```
print(a[:, 0:2])
```



**[[1 2]
[4 5]]**



	0	1	2
0	1	2	3
1	4	5	6

- **:** en la parte de las filas significa que se seleccionan todas las filas.
- **0:2** en la parte de las columnas significa que se seleccionan las columnas desde el índice 0 hasta el índice 2 (sin incluir el índice 2).
- **a[:, 0:2]** selecciona todas las filas y las dos primeras columnas de la matriz a.

Ejemplo:

```
import numpy as np  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a[1, 0])  
print(a[1][0]) #Igual que el anterior
```



Programa: Acceso a arreglos con NumPy:

```
import numpy as np
```

```
matriz = np.array([[1, 2, 3], [4, 5, 6]])  
print(matriz)  
print("Visualiza elemento de la fila 1. columna 0")  
print(matriz[1, 0])  
print(matriz[1][0]) #Igual que el anterior
```

```
|matriz = np.array([[1, 2, 3], [4, 5, 6]])  
print("\nSeleccionando elementos(todas las filas,columnas 0 y 1 ")  
print(matriz)  
print(matriz[:, 0:2])
```


Ejemplo:

Tenemos los datos de las temperaturas diarias (en grados Celsius) registradas en una ciudad durante una semana. Utiliza NumPy para calcular la temperatura media, la temperatura máxima y la temperatura mínima de la semana.

Día	Temperaturas
Lunes	22
Martes	24
Miércoles	23
Jueves	25
Viernes	21
Sábado	26
Domingo	20

Programa: Estadística descriptiva con un arreglo de temperaturas

```
import numpy as np

#Datos de temperaturas diarias
temperaturas_diarias = np.array([22, 24, 23, 25, 21, 26, 20])
#Calculamos la temperatura media
temperatura_media = np.mean(temperaturas_diarias)
#Calculamos la temperatura máxima
temperatura_maxima = np.max(temperaturas_diarias)
#Calculamos la temperatura mínima
temperatura_minima = np.min(temperaturas_diarias)
#Visualizar los resultados
print("Temperaturas: ",temperaturas_diarias)
print("Temperatura media:", temperatura_media)
print("Temperatura máxima:", temperatura_maxima)
print("Temperatura mínima:", temperatura_minima)
```

Práctica: Operación básica con NumPy:

- Crea un array llamado vector con los números del 1 al 5.
- Calcula la media y la desviación estándar del array vector.
- Crea una matriz llamada matriz con 3 filas y 2 columnas, llenándola con números aleatorios entre 1 y 10.
- Multiplica todos los elementos de la matriz por 2 y muestra por pantalla el resultado de la multiplicación

Programa: Práctica operaciones con Numpy

```
import numpy as np
```

```
#Crear arreglo con los números del 1 al 5  
arreglo=np.array([1,2,3,4,5])  
print ("Arreglo: ",arreglo)
```

```
#Calcular media y desviación estándar  
media=np.mean(arreglo)  
desviacion=np.std(arreglo)  
print("\nMedia: ",media)  
print("Desviación estándar: ",desviacion)
```

```
#Crear matriz de 3X2 y llenarla con los números del 1 al 10  
matriz=np.random.randint(1,11,size=(3,2))  
print("\nMatriz de 3X2 con datos aleatorios entre 1 y 10")  
print(matriz)
```

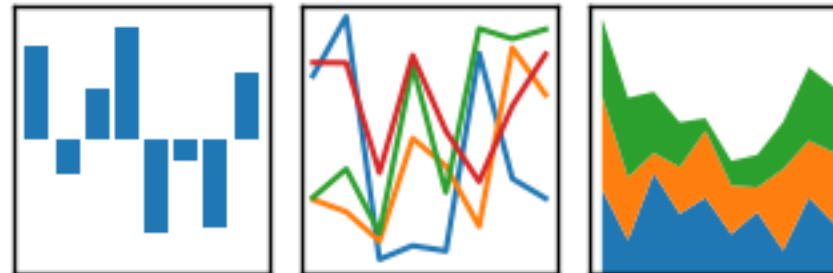
```
#Multiplica los elementos de la matriz por 2  
matriz1=matriz*2  
print("\nMatriz original:")  
print(matriz)  
print("Matriz con datos multiplicados por 2")  
print(matriz1)
```



- Pandas es una librería o biblioteca de **código abierto** en Python que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar (por ejemplo, tablas). Fue creado originalmente por Wes McKinney en 2008 y desde entonces se ha convertido en una de las herramientas más populares para el análisis de datos en Python (es decir, estadísticas).
- Pandas permite realizar tareas importantes, como alinear datos para su comparación, fusionar conjuntos de datos, gestión de datos perdidos, etc

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Define nuevas estructuras de datos basadas en los arrays de la librería **NumPy** pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente archivos en formato CSV, JSON, Excel, bases SQL, parquet y HDF5 entre otros.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Permite trabajar con series temporales.
- Seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas.
- Hacer gráficas
- Realiza todas estas operaciones de manera muy eficiente.

- **Paso No.1:** Nos ubicamos en la ruta de la carpeta de Python.

Ejemplo: C:\Users\USUARIO\AppData\Local\Programs\Python\Python311

- **Paso No.2:** Abrimos **cmd** con esa ruta y ejecutamos el comando PIP:

py -m pip install pandas

Pandas con librerías para leer archivos de Excel:

py -m pip install "pandas[excel]"

Pandas con librerías para leer tablas de HTML:

py -m pip install "pandas[html]"

Pandas dispone de tres estructuras de datos diferentes:

- **Series**: Estructura de una dimensión.
- **DataFrame**: Estructura de dos dimensiones (tablas).
- **Panel**: Estructura de tres dimensiones (cubos).

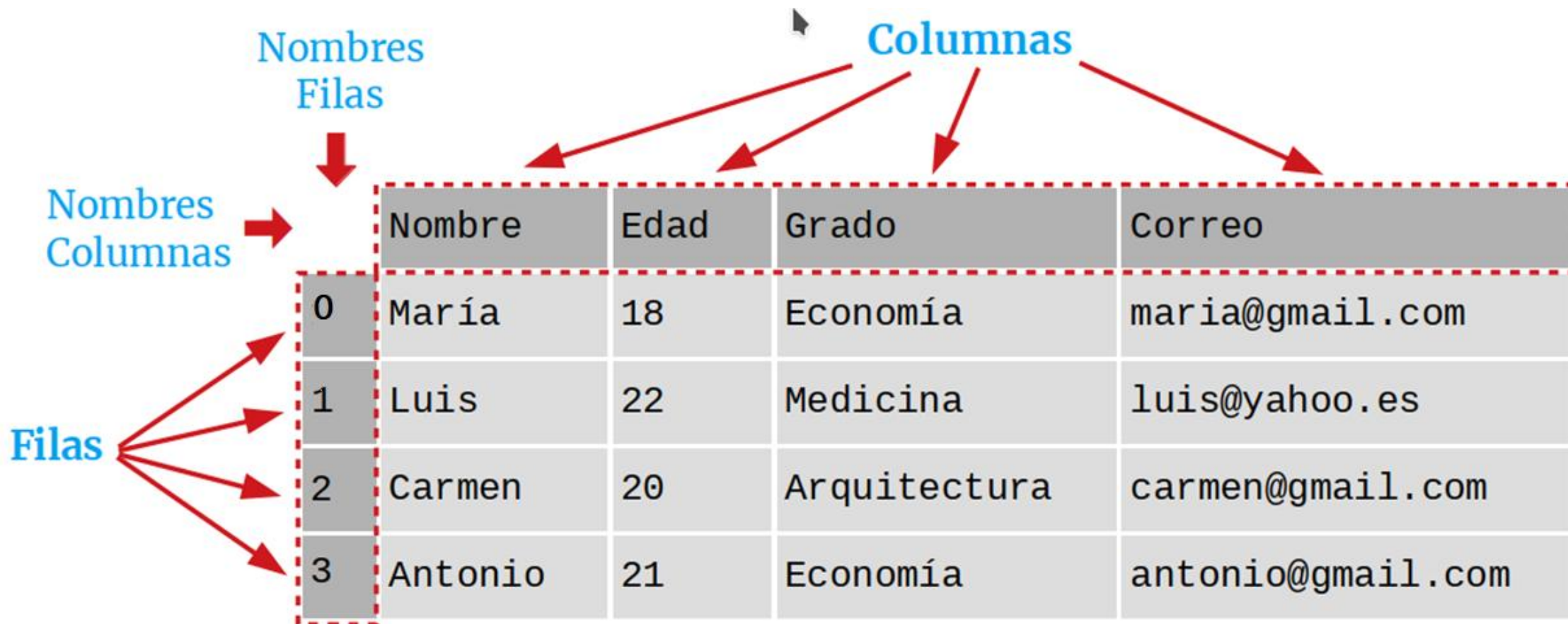
Estas estructuras se construyen a partir de arrays de la librería **NumPy**, añadiendo nuevas funcionalidades.

- Un objeto del tipo **DataFrame** define un conjunto de datos estructurado, tabular bidimensional y mutable, en forma de tabla donde todos los datos de una misma columna son del mismo tipo, y las filas son registros que pueden contener datos de distintos tipos.
- Un DataFrame contiene dos índices, uno para las filas y otro para las columnas, y se puede acceder a sus elementos mediante los nombres de las filas y las columnas.
- Es similar a una hoja de cálculo o una tabla de base de datos SQL o una matriz.

Características principales de un DataFrame:

- **Bidimensional:** Los datos están organizados en filas y columnas, formando una tabla.
- **Índice:** Cada fila del DataFrame tiene un índice que identifica de manera única esa fila. El índice puede ser numérico o de otro tipo, como etiquetas de texto o fechas.
- **Columnas etiquetadas:** Cada columna del DataFrame tiene un nombre o etiqueta que la identifica.
- **Heterogeneidad de datos:** Las columnas de un DataFrame pueden contener diferentes tipos de datos (por ejemplo, enteros, flotantes, cadenas, etc.).
- **Mutable:** Los datos en un DataFrame pueden ser modificados, añadiendo, eliminando o modificando filas o columnas.

Ejemplo. El siguiente DataFrame contiene información sobre los alumnos de un curso. Cada fila corresponde a un alumno y cada columna a una variable.



The diagram illustrates a DataFrame structure with annotations. A red dashed box highlights the entire table. Red arrows point from labels to specific parts of the table: 'Nombres Filas' points to the row index column, 'Nombres Columnas' points to the column header row, 'Filas' points to the row index column, and 'Columnas' points to the column header row.

	Nombre	Edad	Grado	Correo
0	María	18	Economía	maria@gmail.com
1	Luis	22	Medicina	luis@yahoo.es
2	Carmen	20	Arquitectura	carmen@gmail.com
3	Antonio	21	Economía	antonio@gmail.com

1. Creación de un DataFrame a partir de un archivo CSV:

`read_csv(archivo.csv, sep=separador, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal):`

- Devuelve un objeto del tipo DataFrame con los datos del **archivo CSV** usando como separador de los datos la cadena con el **separador**. Como nombres de columnas se utiliza los valores de la fila **n** y como nombres de filas los valores de la columna **m**. Si no se indica **m** se utilizan como nombres de filas los enteros empezando en 0. Los valores incluidos en la lista **no-validos** se convierten en NaN. Para los datos numéricos se utiliza como separador de decimales el carácter indicado en **separador-decimal**.

2. Creación de un DataFrame a partir de un archivo Excel:

`read_excel(archivo.xlsx, sheet_name=hoja, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal):`

- Devuelve un objeto del tipo DataFrame con los datos de la hoja de cálculo **hoja** del archivo **archivo.xlsx**. Como nombres de columnas se utilizan los valores de la fila **n** y como nombres de filas los valores de la columna **m**. Si no se indica **m** se utilizan como nombres de filas los enteros empezando en 0. Los valores incluidos en la lista **no-validos** se convierten en NaN. Para los datos numéricos se utiliza como separador de decimales el carácter indicado en **separador-decimal**.

Programa: Leer archivos CSV:

```
import pandas as pd

#Leer archivo csv e imprimir las primeras 5 filas
df = pd.read_csv('C:\\Otros\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\AnaliticaDatos\\colesterol.csv')
print("Archivo CSV colesterol.csv")
#print(df.head()) #head() visualiza los primeros 5 registros
print(df) #visualiza toda la información
```

Archivo CSV colesterol.csv

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
2	Javier García Sánchez	24	H	NaN	1.81	191.0
3	Carmen López Pinzón	35	M	65.0	1.7	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0
5	Antonio Ruiz Cruz	68	H	66.0	1.74	249.0
6	Antonio Fernández Ocaña	51	H	62.0	1.72	276.0
7	Pilar Martín González	22	M	60.0	1.66	NaN
8	Pedro Gálvez Tenorio	35	H	90.0	1.94	241.0
9	Santiago Reillo Manzano	46	H	75.0	1.85	280.0
10	Macarena Álvarez Luna	53	M	55.0	1.62	262.0
11	José María de la Guía Sanz	58	H	78.0	1.87	198.0
12	Miguel Angel Cuadrado Gutiérrez	27	H	109.0	1.98	210.0
13	Carolina Rubio Moreno	20	M	61.0	1.77	194.0

Programa: Leer archivos EXCEL:

```
#Leer archivo excel e imprimir las primeras 5 filas
df = pd.read_excel('C:\\Otros\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\AnaliticaDatos\\Mercado_casa
print("\nArchivo EXCEL Mercado_casa.xlsx")
print(df.head()) #los primeros 5 registros
#print(df) #Todos los primeros registros
```

Archivo EXCEL Mercado_casa.xlsx								
	Año	Mes	Viveres (Harinas, pan, enlatados, etc.)			Verduras	Frutas	
0	2024	Ene	200000			60000	40000	
1	2024	Feb	210000			65000	40000	
2	2024	Mar	210000			70000	45000	
3	2024	Abr	225000			80000	42000	
4	2024	May	240000			85000	50000	
			Carnes	Lácteos	Aseo personal	Limpieza	Mascotas	Papeleria
0			250000	100000	150000	100000	80000	100000
1			270000	105000	160000	110000	70000	300000
2			275000	105000	165000	115000	75000	150000
3			275000	110000	165000	117000	80000	120000
4			280000	115000	165000	120000	85000	80000

El acceso a los datos de un DataFrame se puede **hacer a través de posiciones** o través de los nombres de las filas y columnas:

- **Acceso por posición**

- **`df.iloc[i, j]`**: Devuelve el elemento que se encuentra en la fila `i` y la columna `j` (por índice numérico) del DataFrame `df`. **Ej:**
`print(df_estudiantes.iloc[2,3])`
- **`df.iloc[filas, columnas]`**: Devuelve un DataFrame con los elementos de las filas de la lista `filas` y de las columnas de la lista `columnas`. **Ej:**
`print(df_estudiantes.iloc[[0,3],[0,1]])`
- **`df.iloc[i]`**: Devuelve una serie con los elementos de la fila `i` del DataFrame `df`. **Ej:** `print(df_estudiantes.iloc[0])`

Programa: Acceso a DataFrame por posición:

```
import pandas as pd

df = pd.read_csv('C:\\Otros\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\AnaliticaDatos\\colesterol.csv')
print("1. Visualizar los 5 primeros elementos")
print(df.head()) #head() visualiza los primeros 5 registros
```

1. Visualizar los 5 primeros elementos

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
2	Javier García Sánchez	24	H	NaN	1.81	191.0
3	Carmen López Pinzón	35	M	65.0	1.7	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0

Programa: Acceso a DataFrame por posición:

```
print("\n2. Visualizar fila 1, columna 3")  
print(df.iloc[1, 3])
```

```
2. Visualizar fila 1, columna 3  
65.0
```

```
print("\n3. Visualizar fila 1, las 3 primeras coulmnas")  
print(df.iloc[1, :3])
```

```
3. Visualizar fila 1, las 3 primeras coulmnas  
nombre      Rosa Díaz Díaz  
edad                32  
sexo                M  
Name: 1, dtype: object
```

```
print("\n4. Visualizar fila 1")  
print(df.iloc[1])
```

```
4. Visualizar fila 1  
nombre      Rosa Díaz Díaz  
edad                32  
sexo                M  
peso                65.0  
altura            1.73  
colesterol        232.0  
Name: 1, dtype: object
```

- **Acceso a los elementos mediante nombres**
 - **df.loc[fila, columna]:** Devuelve el elemento que se encuentra en la fila con nombre **fila** y la columna de con nombre **columna** del DataFrame df. **Ej:** `print(df_estudiantes.loc[3,"Matemáticas"])`
 - **df[columna]:** Devuelve una serie con los elementos de la columna de nombre columna del DataFrame df. **Ej:** `print(df_estudiantes["Nombre"])`
 - **df.columna:** Devuelve una serie con los elementos de la columna de nombre columna del DataFrame df. Es similar al método anterior pero solo funciona cuando el nombre de la columna no tiene espacios en blanco. **Ej:** `print(df_estudiantes.Nombre)`

Programa: Acceso a DataFrame por nombre:

```
import pandas as pd

df = pd.read_csv('C:\\Otros\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\AnaliticaDatos\\colesterol.csv')
print("1. Visualizar los 5 primeros elementos")
print(df.head()) #head() visualiza los primeros 5 registros
```

1. Visualizar los 5 primeros elementos

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
2	Javier García Sánchez	24	H	NaN	1.81	191.0
3	Carmen López Pinzón	35	M	65.0	1.7	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0

Programa: Acceso a DataFrame por nombre:

```
print("2. Visualizar fila 2, columna llamada colesterol")  
print(df.loc[2, 'colesterol'])
```

```
2. Visualizar fila 2, columna llamada colesterol  
191.0
```

```
print("3. Visualizar hasta la fila 3, las columna colesterol y peso")  
print(df.loc[:3, ('colesterol', 'peso')])
```

```
3. Visualizar hasta la fila 3, las columna colesterol y peso  
colesterol  peso  
0         182.0  85.0  
1         232.0  65.0  
2         191.0   NaN  
3         200.0  65.0
```


Programa: Acceso a DataFrame por nombre:

```
print("4. Visualizar la columna nombre")  
print(df["nombre"])
```

```
print("5. Visualizar la columna nombre segunda forma")  
print(df.nombre)
```

```
4. Visualizar la columna nombre  
0      José Luis Martínez Izquierdo  
1              Rosa Díaz Díaz  
2      Javier García Sánchez  
3      Carmen López Pinzón  
4      Marisa López Collado  
5      Antonio Ruiz Cruz  
6      Antonio Fernández Ocaña  
7      Pilar Martín González  
8      Pedro Gálvez Tenorio  
9      Santiago Reillo Manzano  
10     Macarena Álvarez Luna  
11     José María de la Guía Sanz  
12     Miguel Angel Cuadrado Gutiérrez  
13     Carolina Rubio Moreno  
Name: nombre, dtype: object
```

```
5. Visualizar la columna nombre segunda forma  
0      José Luis Martínez Izquierdo  
1              Rosa Díaz Díaz  
2      Javier García Sánchez  
3      Carmen López Pinzón  
4      Marisa López Collado  
5      Antonio Ruiz Cruz  
6      Antonio Fernández Ocaña  
7      Pilar Martín González  
8      Pedro Gálvez Tenorio  
9      Santiago Reillo Manzano  
10     Macarena Álvarez Luna  
11     José María de la Guía Sanz  
12     Miguel Angel Cuadrado Gutiérrez  
13     Carolina Rubio Moreno  
Name: nombre, dtype: object
```

- **Eliminar columna:**
`df_estudiantes = df_estudiantes.drop(columns=["Categoria"])`
- **Eliminar fila:** `df_estudiantes.drop(3)`
- **Sumar columna=** `suma_matematicas=df_estudiantes["Matemáticas"].sum()`
- **Promedio de columna=**
`promedio_matematicas=df_estudiantes["Matemáticas"].mean()`
- **Sumar columnas en una nueva=**
`df_estudiantes["Suma"]=df_estudiantes["Matemáticas"] + df_estudiantes["Ciencias"]`
- **Crear columna con filtro condicional=**
`df_estudiantes["Rendimiento"]=np.where(df_estudiantes["Promedio"]>85,'Excelente','Bueno')`
- **Modificar el valor de una celda=** `df_estudiantes.at[1,"Ciencias"]=95`
- **Restar valor a una columna=**
`df_estudiantes["Matemáticas"]=df_estudiantes["Matemáticas"] - 5`

Programa: Operaciones con DataFrame:

```
import pandas as pd
import numpy as np

df = pd.read_csv('C:\\Otros\\UTB\\TalentoTech\\AnalisisDatos\\Material\\Semana_4\\Ejercicios\\AnaliticaDatos\\colesterol.csv')
print("1. Visualizar los 5 primeros elementos")
print(df.head()) #head() visualiza los primeros 5 registros
```

1. Visualizar los 5 primeros elementos

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
2	Javier García Sánchez	24	H	NaN	1.81	191.0
3	Carmen López Pinzón	35	M	65.0	1.7	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0

Programa: Operaciones con DataFrame:

```
print("\n2. Sumar columna edad")  
print(df['edad'].sum())
```

2. Sumar columna edad
535

```
print("\n3. Promedio columna edad")  
print(df['edad'].mean())
```

3. Promedio columna edad
38.214285714285715

```
print("\n4. Crear columna con filtro edad <30 joven y mayor adulto ")  
df['tipoedad']=np.where(df['edad']>30,'Adulto','Joven')  
print(df.head()) #head() visualiza los primeros 5 registros
```

4. Crear columna con filtro edad <30 joven y mayor adulto

	nombre	edad	sexo	peso	altura	colesterol	tipoedad
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0	Joven
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0	Adulto
2	Javier García Sánchez	24	H	NaN	1.81	191.0	Joven
3	Carmen López Pinzón	35	M	65.0	1.7	200.0	Adulto
4	Marisa López Collado	46	M	51.0	1.58	148.0	Adulto

Programa: Operaciones con DataFrame:

```
print("\n5. Modifica celda, fila 1, columna nombre por Rosa Pérez Pérez ")
df.at[1, 'nombre'] = "Rosa Pérez Pérez"
print(df.head()) #head() visualiza los primeros 5 registros
```

5. Modifica celda, fila 1, columna nombre por Rosa Pérez Pérez

	nombre	edad	sexo	peso	altura	colesterol	tipoedad
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0	Joven
1	Rosa Pérez Pérez	32	M	65.0	1.73	232.0	Adulto
2	Javier García Sánchez	24	H	NaN	1.81	191.0	Joven
3	Carmen López Pinzón	35	M	65.0	1.7	200.0	Adulto
4	Marisa López Collado	46	M	51.0	1.58	148.0	Adulto

DataFrame: Crea un programa que utilice la librería Pandas para crear un DataFrame con la siguiente información sobre 5 estudiantes: nombres, edades y calificaciones. Muestra el DataFrame resultante.

```
#DataFrame: Crea un programa que utilice la librería Pandas para  
#crear un DataFrame con la siguiente información sobre 5  
#estudiantes: nombres, edades y calificaciones. Muestra el  
#DataFrame resultante.  
  
import pandas as pd  
  
# Paso 1  
datos = pd.DataFrame({  
    'Nombre': ['Ana', 'Juan', 'María', 'Carlos'],  
    'Edad': [22, 30, 25, 28],  
    'Calificacion': [3.5, 4.3, 2.9, 4.7]  
})  
  
# Paso 2  
print("Información del DataFrame:")  
print(datos)
```

	Nombre	Edad	Calificacion
0	Ana	22	3.5
1	Juan	30	4.3
2	María	25	2.9
3	Carlos	28	4.7

Manipulación básica con pandas: Crea un programa que utilice la biblioteca Pandas para realizar las siguientes tareas:

- Crear un DataFrame llamado datos con las siguientes columnas: Nombre, Edad, Ciudad.
- Mostrar por pantalla la información del DataFrame.
- Filtrar las filas donde la edad sea mayor que 25.
- Añadir una nueva columna llamada Categoría que clasifique a las personas en "Joven" si tienen 25 años o menos, y "Adulto" si tienen más de 25 años.
- Mostrar por pantalla el DataFrame actualizado.

Programa:

```
#DataFrame: Crea un programa que utilice la librería Pandas para  
#crear un DataFrame con la siguiente información sobre 5  
#estudiantes: nombres, edades y calificaciones. Muestra el  
#DataFrame resultante.  
  
import pandas as pd  
import numpy as np  
  
# Paso 1  
datos = pd.DataFrame({  
    'Nombre': ['Ana', 'Juan', 'María', 'Carlos'],  
    'Edad': [22, 30, 25, 28],  
    'Ciudad': ['Madrid', 'Barcelona', 'Valencia', 'Sevilla']  
})  
  
# Paso 2  
print("Información del DataFrame:")
```

Información del DataFrame:			
	Nombre	Edad	Ciudad
0	Ana	22	Madrid
1	Juan	30	Barcelona
2	María	25	Valencia
3	Carlos	28	Sevilla

Programa:

```
# Paso 3
datos_filtrados = datos[datos['Edad'] > 25]
print("\nPersonas con edad mayor que 25:")
print(datos_filtrados)
```

```
Personas con edad mayor que 25:
  Nombre  Edad  Ciudad
1   Juan   30  Barcelona
3  Carlos   28   Sevilla
```

Programa:

```
# Paso 4
datos['Categoria']=np.where(datos['Edad']<=25,"Joven","Adulto")
# Paso 5
print("\nDataFrame Actualizado:")
print(datos)
```

DataFrame Actualizado:

	Nombre	Edad	Ciudad	Categoria
0	Ana	22	Madrid	Joven
1	Juan	30	Barcelona	Adulto
2	María	25	Valencia	Joven
3	Carlos	28	Sevilla	Adulto

Filtrado con Pandas: Crea un programa que utilice Pandas para realizar las siguientes tareas:

- Descarga el conjunto de datos de Iris desde la URL: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>.
- Carga los datos en un DataFrame llamado iris.
- Muestra las primeras 5 filas del DataFrame.
- Filtra las filas donde la especie sea "setosa".
- Calcula la media de la longitud de los sépalos para las filas filtradas.

Programa:

```
import pandas as pd

# Paso 1
url_iris = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
iris = pd.read_csv(url_iris, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])

# Paso 2
print("Primeras 5 filas del DataFrame:")
print(iris.head())
```

Primeras 5 filas del DataFrame:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Programa:

```
# Paso 3
print("\nFiltrando las filas donde la especie es 'setosa':")
setosa_data = iris[iris['species'] == 'Iris-setosa']
print(setosa_data)
```

Filtrando las filas donde la especie es 'setosa':

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa

Programa:

```
# Paso 4
media_longitud_sepalos = setosa_data['sepal_length'].mean()
print(f"\nMedia de la longitud de los sépalos para la especie 'setosa': {media_longitud_sepalos}")
```

```
Media de la longitud de los sépalos para la especie 'setosa': 5.006
```

Concatenación de DataFrames: Crea un programa que utilice Pandas para realizar las siguientes tareas:

- Crea dos DataFrames llamados `df1` y `df2` con las siguientes columnas: A, B, C. Llena ambos DataFrames con datos.
- Concatena verticalmente los dos DataFrames para crear un nuevo DataFrame llamado `df_concat`. Muestra por pantalla el resultado de la concatenación.

Programa:

```
import pandas as pd

# Paso 1
df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
df2 = pd.DataFrame({'A': [10, 11, 12], 'B': [13, 14, 15], 'C': [16, 17, 18]})

# Paso 2
print("DataFrame 1:")
print(df1)
print("\nDataFrame 2:")
print(df2)
```

DataFrame 1:

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

DataFrame 2:

	A	B	C
0	10	13	16
1	11	14	17
2	12	15	18

Programa:

```
# Paso 3
df_concat = pd.concat([df1, df2])

# Paso 4
print("\nResultado de la Concatenación:")
print(df_concat)
```

Resultado de la Concatenación:

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9
0	10	13	16
1	11	14	17
2	12	15	18



ADVANCE
Consortium

GRACIAS

