Universidad Tecnológica del Perú

FACULTAD DE INGENIERÍA

Domina Python y Conquista el Mundo de los Datos.

PYTHON Y EL MUNDO DE LOS DATOS

SESIÓN 04

Introducción a las Bibliotecas NumPy y Pandas

Logros de Aprendizaje	
General	Desarrollar competencias en programación con Python, desde fundamentos básicos hasta el manejo avanzado de datos con NumPy y Pandas. Los participantes aprenderán a implementar estructuras, funciones, conceptos de POO (herencia y encapsulamiento) y librerías de análisis, integrando estos conocimientos en un caso práctico para resolver problemas reales y presentar resultados de manera efectiva.
Sesión	Al finalizar la sesión, el estudiante desarrolla programas con las bibliotecas NumPy y Pandas para la manipulación de DataFrames y arrays mediante la resolución de casos planteados

PYTHON Y EL MUNDO DE LOS DATOS

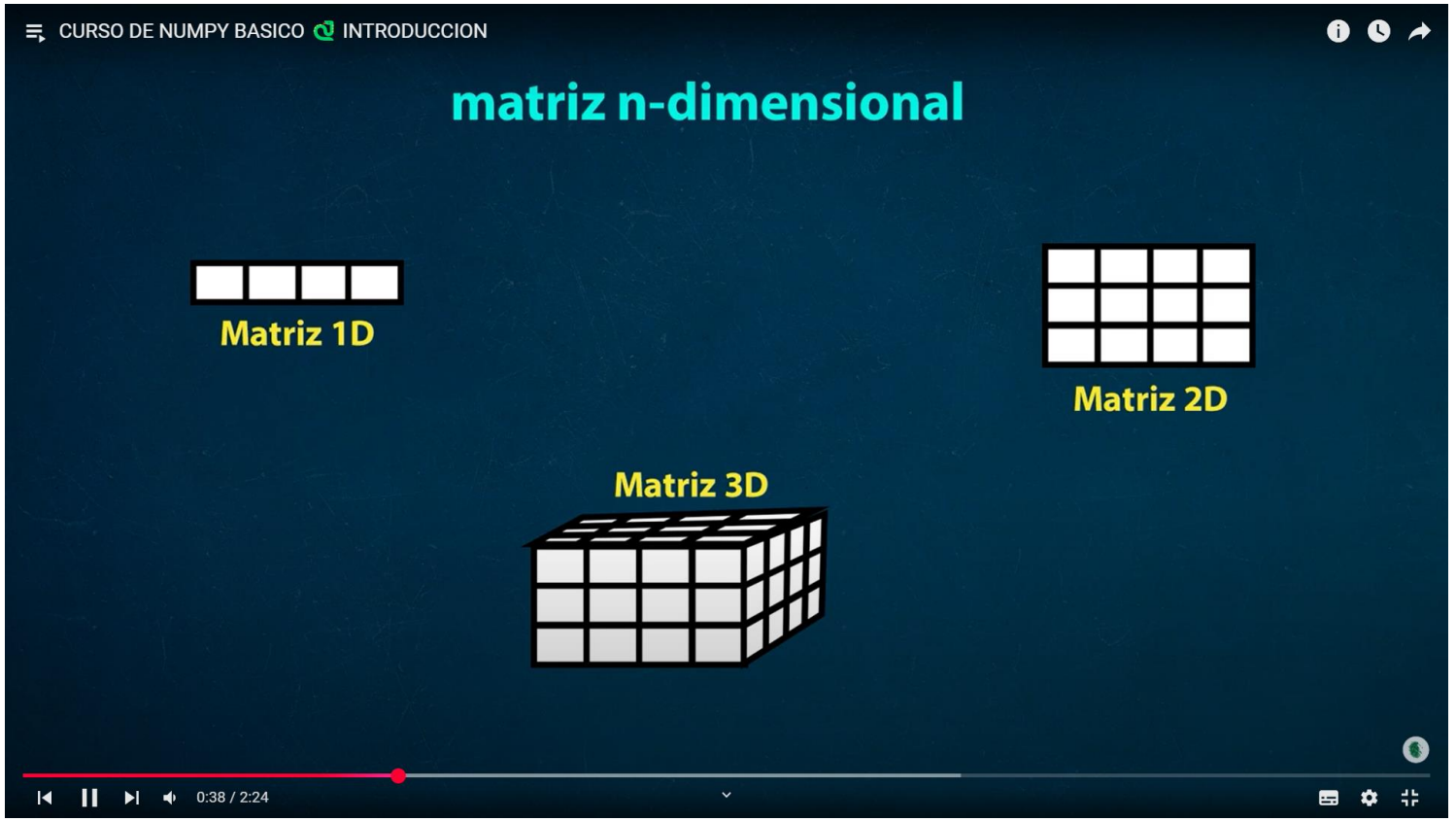
SESIÓN 04

Bibliotecas NumPy y Pandas

- Introducción a NumPy:
- Creación de Arrays y Operaciones Básicas.
- Introducción a Pandas:
- Creación y Manipulación de DataFrames, Selección y Filtrado de Datos.

INICIO - Conocimientos Previos

¿Qué es NumPy en Python?



https://www.youtube.com/watch?v=V9wNBGVq3hg&list=PLg9145ptuAij_8zYgMeqwOV8ABwRYLuR3

UTILIDAD - Sesión 04

Universidad Tecnológica del Perú

¿Cuál es la utilidad de Numpy en Python?

NumPy es una biblioteca fundamental en Python para el manejo de datos numéricos y cálculos científicos, ya que permite trabajar con arrays N-dimensionales de manera eficiente, ofreciendo operaciones matemáticas rápidas y optimizadas; por ejemplo, para calcular el promedio de un conjunto de datos.

```
1   import numpy as np
2   notas = np.array([15, 18, 20, 17, 19])
3   print("Promedio:", np.mean(notas))
```

Esto produce como salida:

```
Promedio: 17.8
```

NumPy simplifica y acelera cálculos que serían más lentos con estructuras tradicionales como listas.

TRANSFORMACIÓN

Definición NumPy:
Creación de Arrays y operaciones básicas.

- **NumPy** = Numerical Python (https://numpy.org/)
- Creada en 2005 por Travis Oliphant como proyecto de código abierto.
- **NumPy** es una biblioteca fundamental para el cálculo numérico en Python. Proporciona soporte para trabajar con **arrays n-dimensionales** y funciones matemáticas de alto rendimiento.

Un **array** es una estructura de datos optimizada para realizar cálculos numéricos de manera más eficiente. Pueden ser de una o más dimensiones.

Características de NumPy en Python.

1. **Arrays n-dimensionales (ndarrays):**
   - Soporta arrays de 1D (vectores), 2D (matrices) y más dimensiones.
2. **Rendimiento:**
   - Las operaciones con arrays son más rápidas que con las listas de Python.
3. **Variedad de Funciones:**
   - Incluye funciones matemáticas avanzadas (trigonométricas, estadísticas, etc.).
4. **Tipado Eficiente:**
   - Los arrays tienen un tipo de datos homogéneo, lo que mejora la eficiencia.
5. **Integración con Otras Bibliotecas:**
   - Compatible con bibliotecas como Pandas, Matplotlib y SciPy.

Instalación de NumPy y creación de Arrays.

Universidad Tecnológica del Perú

1. Instalación de NumPy

Desde VS Code abre una nueva terminar y escribe el comando:

```
pip install numpy
```

2. Creación de Arrays

a) Importación de NumPy

b) Creación de arrays:

```
import numpy as np


# Arreglo de 1 dimensión
arr1 = np.array([1, 2, 3, 4, 5])
print(arr1)

# Arreglo de 2 dimensiones
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
```

Instalación de NumPy y creación de Arrays.

Universidad Tecnológica del Perú

c) Arrays con Valores Predefinidos

```
arr1 = np.array([1, 2, 3, 4, 5])  # array creado desde una lista
print(arr1)

arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)

array_2d_ceros = np.zeros((3, 3))  # Matriz 3x3 de ceros
print(array_2d_ceros)

array_2d_unos = np.ones((2, 4))  # Matriz 2x4 de unos
print(array_2d_unos)

array_aleatorios = np.random().rand(3, 2)  # Matriz 3x2 con aleatorios
print(array_aleatorios)

array_rango = np.arange(0, 10, 2)  # Valores 0-10 con paso 2
print(array_rango)

array_espaciado = np.linspace(0, 10, 5)  # 5 valores equidistantes entre 0-10
print(array_espaciado)
```

```
[1 2 3 4 5]

[[1 2 3]
 [4 5 6]]

[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]

[[0.37190177 0.71203156]
 [0.82369659 0.06396568]
 [0.70593086 0.59130443]]

[0 2 4 6 8]

[ 0.   2.5  5.   7.5 10. ]
```

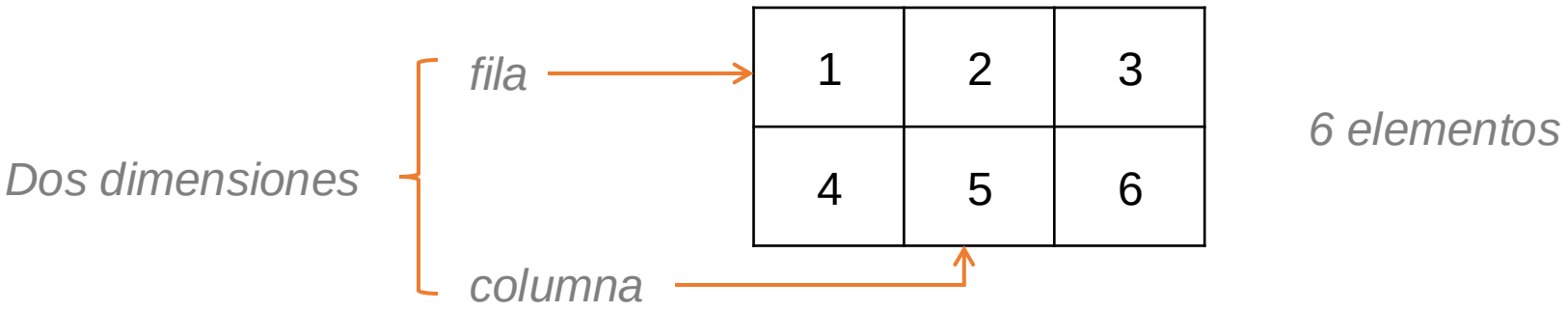
Propiedades de NumPy y creación de Arrays.

3. Propiedades de los Arrays.

```
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

print("Dimensiones:", arr2.ndim)
print("Forma:", arr2.shape)
print("Tamaño:", arr2.size)
print("Tipo de datos:", arr2.dtype)
```

```
Dimensiones: 2
Forma: (2, 3)
Tamaño: 6
Tipo de datos: int64
```



Operaciones con NumPy y creación de Arrays.

Universidad Tecnológica del Perú

4. Operaciones Básicas.

a) Operaciones Matemáticas.

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

print("Suma:", arrayA + arrayB)
print("Resta:", arrayA - arrayB)
print("Multiplicación:", arrayA * arrayB)
print("División:", arrayA / arrayB)
```

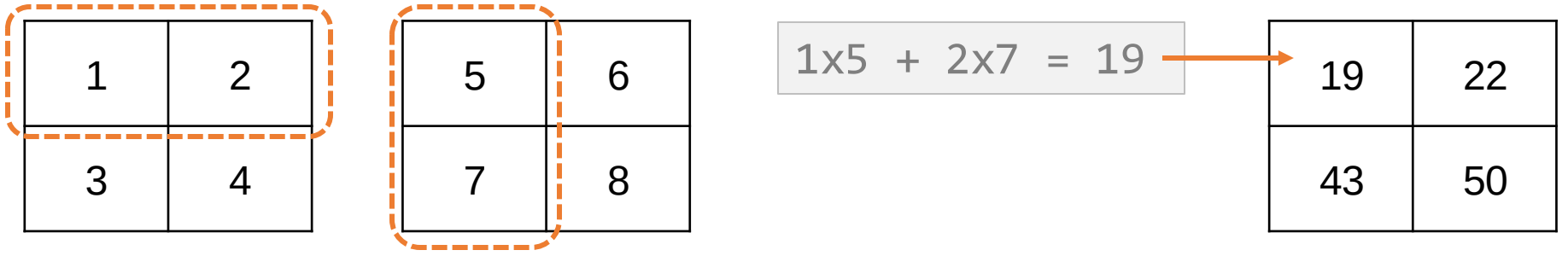
```
Suma: [5 7 9]
Resta: [-3 -3 -3]
Multiplicación: [ 4 10 18]
División: [0.25 0.4  0.5 ]
```

Operaciones con NumPy y creación de Arrays.

- Operando matrices:

```
matriz1 = np.array([[1, 2], [3, 4]])
matriz2 = np.array([[5, 6], [7, 8]])
matriz_producto = np.dot(matriz1, matriz2)
print(matriz_producto)
```

```
[[19  22]
 [43  50]]
```



Operaciones con NumPy y creación de Arrays.

Universidad Tecnológica del Perú

4. Operaciones Básicas.

b) Funciones Matemáticas.

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

print("Raíz cuadrada:", np.sqrt(arr))
print("Seno:", np.sin(arr))
print("Logaritmo:", np.log(arr))
print("Promedio:", np.mean(arr))
print("Suma total:", np.sum(arr))
```

```
Raíz cuadrada: [1.         1.41421356 1.73205081 2.         2.23606798]
Seno: [ 0.84147098  0.90929743  0.14112001 -0.7568025  -0.95892427]
Logaritmo: [0.         0.69314718 1.09861229 1.38629436 1.60943791]
Promedio: 3.0
Suma total: 15
```

Operaciones con NumPy y creación de Arrays.

4. Operaciones Básicas.

c) Indexación y slicing.

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

print("Elemento [0, 1]:", arr[0, 1])
print("Primera fila:", arr[0, :])
print("Segunda columna:", arr[:, 1])
```

1	2	3
4	5	6

```
Elemento [0, 1]: 2
Primera fila: [1 2 3]
Segunda columna: [2 5]
```

Ejemplo completo de la aplicación de NumPy.

```python
matriz = np.arange(1, 10).reshape(3, 3)
print("Matriz original:\n", matriz)

suma_columnas = np.sum(matriz, axis = 0)
print("Suma de cada columna:\n", suma_columnas)

suma_filas = np.sum(matriz, axis = 1)
print("Suma de cada fila:\n", suma_filas)

nueva_matriz = matriz * 2
print("Nueva matriz multiplicada x 2:\n", nueva_matriz)

transpuesta = matriz.T
print("Matriz transpuesta:\n", transpuesta)
```

```
Matriz original:
 [[1 2 3]
  [4 5 6]
  [7 8 9]]

Suma de cada columna: [12 15 18]


Suma de cada fila: [ 6 15 24]


Matriz multiplicada x 2:
 [[ 2  4  6]
  [ 8 10 12]
  [14 16 18]]

Matriz transpuesta:
 [[1 4 7]
  [2 5 8]
  [3 6 9]]
```

Conclusión del uso NumPy en Python

- NumPy es una herramienta poderosa que permite trabajar con datos numéricos de manera eficiente.

- Aprender a crear arrays y realizar operaciones básicas es el primer paso para aprovechar todo su potencial en análisis de datos, aprendizaje automático y cálculos científicos.

TRANSFORMACIÓN

Introducción a Pandas: Creación y Manipulación de DataFrames, Selección y Filtrado de Datos.

- **Pandas** es una biblioteca utilizada para **análisis y manipulación de datos** (https://pandas.pydata.org/).
- Proporciona dos estructuras de datos principales:
  - **Series:** Unidimensional, similar a una lista o un array.
  - **DataFrame:** Bidimensional, similar a una tabla (como en Excel o SQL).
- Con Pandas puedes importar, limpiar, manipular, analizar y exportar datos de manera eficiente.

Características de Pandas en Python.

1. **Estructuras de Datos Flexibles:**
    - Tablas bidimensionales con etiquetas (DataFrames).
    - Columnas con diferentes tipos de datos.
2. **Importación y Exportación:**
    - Compatible con archivos CSV, Excel, SQL, JSON, entre otros.
3. **Herramientas de Selección y Filtrado:**
    - Permite seleccionar filas/columnas según condiciones.
4. **Funciones de Agregación y Estadísticas:**
    - Promedios, sumas, conteos, etc.
5. **Manejo de Datos Faltantes:**
    - Detecta y llena datos faltantes fácilmente.

Instalación de Pandas.

- Abre una nueva terminal en VS Code y escribe el comando:

```
pip install pandas
```

- Obtendrás una notificación de la instalación:

```
PS D:\UTP\Material Estandarizado\Taller Python\Practica_Python> pip install pandas
Collecting pandas
  Downloading pandas-2.2.3-cp310-cp310-win_amd64.whl (11.6 MB)
     ---------------------------------------- 11.6/11.6 MB 38.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.22.4 in d:\utp\material estandarizado\taller python\practica_python\pythonenv\lib\site-packages (from pandas) (2.2.2)
Collecting python-dateutil>=2.8.2
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
     ---------------------------------------- 229.9/229.9 kB 13.7 MB/s eta 0:00:00
Collecting tzdata>=2022.7
  Downloading tzdata-2025.1-py2.py3-none-any.whl (346 kB)
     ---------------------------------------- 346.8/346.8 kB 21.0 MB/s eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2024.2-py2.py3-none-any.whl (508 kB)
     ---------------------------------------- 508.0/508.0 kB 31.1 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, tzdata, six, python-dateutil, pandas
Successfully installed pandas-2.2.3 python-dateutil-2.9.0.post0 pytz-2024.2 six-1.17.0 tzdata-2025.1
```

- Para verificar que funciona correctamente, escribe el siguiente código en un archivo .py

```
import pandas as pd

print(pd.__version__)
```

Creación de DataFrame con Pandas

1. Creación de DataFrames

a) Crear un DataFrame desde un Diccionario.

```
diccionario = {
    'Nombre': ['Ana', 'Luis', 'Carlos', 'Marta'],
    'Edad': [22, 35, 45, 23],
    'Ciudad': ['Lima', 'Chiclayo', 'Trujillo', 'Piura']
}

data_frame = pd.DataFrame(diccionario)
print(data_frame)
```

```
   Nombre  Edad    Ciudad
0     Ana    22      Lima
1    Luis    35  Chiclayo
2  Carlos    45  Trujillo
3   Marta    23     Piura
```

Creación de DataFrame con Pandas

Universidad Tecnológica del Perú

- Creamos un DataFrame a partir de una lista de listas:

```
lista_datos = [['Ana', 22, 'Lima'], \
               ['Luis', 35, 'Chiclayo'], \
               ['Carlos', 45, 'Trujillo'], \
               ['Marta', 23, 'Piura']]

data_frame2 = pd.DataFrame(lista_datos, columns=['Nombre', 'Edad', 'Ciudad'])
print(data_frame2)
```

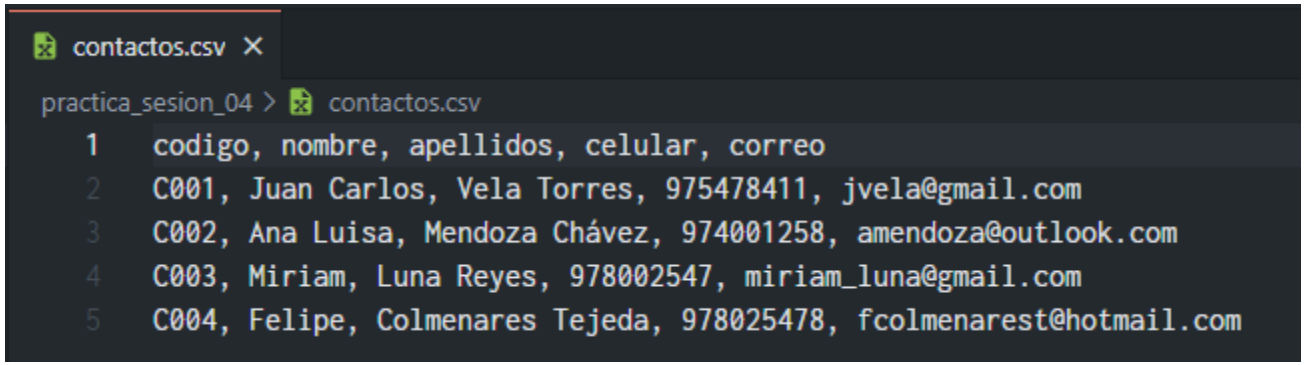
```
   Nombre  Edad    Ciudad
0     Ana    22      Lima
1    Luis    35  Chiclayo
2  Carlos    45  Trujillo
3   Marta    23     Piura
```

Creación de DataFrame con Pandas

Universidad Tecnológica del Perú

1. Creación de DataFrames

b) Crear un DataFrame desde un Archivo CSV.




```
df = pd.read_csv("D:\\contactos.csv")
print(df)
```

```
  codigo        nombre          apellidos    celular                    correo
0   C001   Juan Carlos        Vela Torres  975478411          jvela@gmail.com
1   C002     Ana Luisa     Mendoza Chávez  974001258       amendoza@outlook.com
2   C003        Miriam        Luna Reyes  978002547     miriam_luna@gmail.com
3   C004        Felipe  Colmenares Tejeda  978025478  fcolmenarest@hotmail.com
```

Propiedades Básicas de Pandas

```
print(df.columns)  # Columnas
```

```
Index(['codigo', ' nombre', ' apellidos', ' celular', ' correo'], dtype='object')
```

```
print(df.index) # Índices
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
print(df.shape) # Dimensiones
```

```
(4, 5)
```

```
print(df.head(2)) # Primeras dos filas
```

```
  codigo        nombre        apellidos    celular                 correo
0   C001   Juan Carlos      Vela Torres  975478411       jvela@gmail.com
1   C002     Ana Luisa   Mendoza Chávez  974001258   amendoza@outlook.com
```

```
print(df.tail(2)) # Últimas dos filas
```

```
  codigo   nombre           apellidos    celular                    correo
2   C003   Miriam         Luna Reyes  978002547     miriam_luna@gmail.com
3   C004   Felipe   Colmenares Tejeda  978025478   fcolmenarest@hotmail.com
```

Selección y Filtrado de Datos

```
lista_datos = [['Ana', 22, 'Lima'], \
               ['Luis', 35, 'Chiclayo'], \
               ['Carlos', 45, 'Trujillo'], \
               ['Marta', 23, 'Piura']]

df = pd.DataFrame(lista_datos, columns=['Nombre', 'Edad', 'Ciudad'])
```

	Nombre	Edad	Ciudad
0	Ana	22	Lima
1	Luis	35	Chiclayo
2	Carlos	45	Trujillo
3	Marta	23	Piura

a) Seleccionar Columnas.

```
# Seleccionar una columna
print(df["Nombre"])
```

```
0     Ana
1    Luis
2  Carlos
3   Marta
Name: Nombre, dtype: object
```

```
# Seleccionar múltiples columnas
print(df[["Nombre", "Ciudad"]])
```

```
   Nombre    Ciudad
0     Ana      Lima
1    Luis  Chiclayo
2  Carlos  Trujillo
3   Marta     Piura
```

Selección y Filtrado de Datos

```
lista_datos = [['Ana', 22, 'Lima'], \
               ['Luis', 35, 'Chiclayo'], \
               ['Carlos', 45, 'Trujillo'], \
               ['Marta', 23, 'Piura']]

df = pd.DataFrame(lista_datos, columns=['Nombre', 'Edad', 'Ciudad'])
```

	Nombre	Edad	Ciudad
0	Ana	22	Lima
1	Luis	35	Chiclayo
2	Carlos	45	Trujillo
3	Marta	23	Piura

b) Seleccionar Filas.

```
# Seleccionar una fila por índice
# fila con índice 1
print(df.iloc[1])
```

```
Nombre        Luis
Edad            35
Ciudad    Chiclayo
Name: 1, dtype: object
```

```
# Seleccionar filas con etiquetas
# filas con índics 0 al 2
print(df.loc[0:2])
```

```
   Nombre  Edad    Ciudad
0     Ana    22      Lima
1    Luis    35  Chiclayo
2  Carlos    45  Trujillo
```

Selección y Filtrado de Datos

```
lista_datos = [['Ana', 22, 'Lima'], \
               ['Luis', 35, 'Chiclayo'], \
               ['Carlos', 45, 'Trujillo'], \
               ['Marta', 23, 'Piura']]

df = pd.DataFrame(lista_datos, columns=['Nombre', 'Edad', 'Ciudad'])
```

	Nombre	Edad	Ciudad
0	Ana	22	Lima
1	Luis	35	Chiclayo
2	Carlos	45	Trujillo
3	Marta	23	Piura

c) Filtrar Datos:

```
# filtrar por una condición
df_filtro_edad = df[df["Edad"] > 25]
print(df_filtro_edad)
```

	Nombre	Edad	Ciudad
1	Luis	35	Chiclayo
2	Carlos	45	Trujillo

```
# filtrar por múltiples condiciones
df_filtro_multiple = df[(df["Edad"] < 25) & \
                        (df["Ciudad"] == "Lima")]
print(df_filtro_multiple)
```

	Nombre	Edad	Ciudad
0	Ana	22	Lima

Manipulación de DataFrames

- Agregamos una columna *(utilizando el DataFrame del slide anterior)*

```
df['Profesión'] = ['Ingeniera', \
                   'Médico', \
                   'Abogado', \
                   'Diseñadora']
print(df)
```

```
   Nombre  Edad    Ciudad   Profesión
0     Ana    22      Lima   Ingeniera
1    Luis    35  Chiclayo      Médico
2  Carlos    45  Trujillo     Abogado
3   Marta    23     Piura  Diseñadora
```

- Modificamos una columna:

```
df["Edad"] = df["Edad"] + 3
print(df)
```

```
   Nombre  Edad    Ciudad
0     Ana    25      Lima
1    Luis    38  Chiclayo
2  Carlos    48  Trujillo
3   Marta    26     Piura
```

Manipulación de DataFrames

- Eliminamos una columna:

```
df = df.drop("Edad", axis = 1)
print(df)
```

```
   Nombre    Ciudad
0     Ana      Lima
1    Luis  Chiclayo
2  Carlos  Trujillo
3   Marta     Piura
```

- Eliminamos una fila:

```
df = df.drop(2, axis = 0)
print(df)
```

```
  Nombre    Ciudad
0    Ana      Lima
1   Luis  Chiclayo
3  Marta     Piura
```

Manipulación de DataFrames

Universidad Tecnológica del Perú

c) Manejo de Datos Faltantes.

Sea el DataFrame:

```
dic_datos = {
    'id': [1, 2, 3, 4],
    'codigo': ['A034', 'C983', None, 'M099'],
    'nombre': ['Mouse', 'Teclado', 'Cámara', 'Audífonos'],
    'precio': [None, 279.40, None, 310.95] }
df = pd.DataFrame(dic_datos)
print(df)
```

```
   id codigo     nombre  precio
0   1   A034      Mouse     NaN
1   2   C983    Teclado  279.40
2   3   None     Cámara     NaN
3   4   M099  Audífonos  310.95
```

- Detectar valores faltantes:

```
print(df.isnull())
```

```
      id  codigo  nombre  precio
0  False   False   False    True
1  False   False   False   False
2  False    True   False    True
3  False   False   False   False
```

Manipulación de DataFrames

- Llenar datos faltantes con un valor:

```
df.fillna("(Desconocido)", inplace = True)
```

	id	codigo	nombre	precio
0	1	A034	Mouse	(Desconocido)
1	2	C983	Teclado	279.4
2	3	(Desconocido)	Cámara	(Desconocido)
3	4	M099	Audífonos	310.95

- Eliminar filas con datos faltantes:

```
df = df.dropna()
```

	id	codigo	nombre	precio
1	2	C983	Teclado	279.40
3	4	M099	Audífonos	310.95

Operación Básicas con DataFrames

```
dic_datos = { 'id': [1, 2, 3, 4],
    'codigo': ['A034', 'C983', 'B365', 'M099'],
    'nombre': ['Mouse', 'Teclado', 'Cámara', 'Audífonos'],
    'precio': [45.6, 279.4, 254.5, 310.95] }
df = pd.DataFrame(dic_datos)
print(df)
```

```
   id codigo     nombre  precio
0   1   A034      Mouse   45.60
1   2   C983    Teclado  279.40
2   3   B365     Cámara  254.50
3   4   M099  Audífonos  310.95
```

```
print("Precio promedio:", df["precio"].mean())
print("Precio más alto:", df["precio"].max())
print("Conteo de filas:", len(df))
```

```
Precio promedio: 222.6125
Precio más alto: 310.95
Conteo de filas: 4
```

Ejemplo Completo de la aplicación de Pandas

Universidad Tecnológica del Perú

```python
df = pd.DataFrame({
    "Nombre": ['Cristian', 'Violeta', 'Armando', 'Mariela'],
    "Edad": [32, 28, 35, 29],
    "Area": ['Ventas', 'RRHH', 'Finanzas', 'RRHH'],
    "Salario": [3400, 4300, 4800, 3850] })

# DataFrame inicial
print(df)


# Empleados de RRHH que ganan al menos 4000 soles
rrhh_filtrado = df[(df["Area"] == "RRHH") & (df["Salario"] >= 4000)]
print(rrhh_filtrado)


# Añadimos la columna Bonificacion (3.5% del salario)
df["Bonificacion"] = df["Salario"] * 0.035
print(df)


# Ordenamos por salario en orden descendente
df_ordenado = df.sort_values(by="Salario", ascending = False)
print(df_ordenado)
```

	Nombre	Edad	Area	Salario
0	Cristian	32	Ventas	3400
1	Violeta	28	RRHH	4300
2	Armando	35	Finanzas	4800
3	Mariela	29	RRHH	3850

	Nombre	Edad	Area	Salario
1	Violeta	28	RRHH	4300

	Nombre	Edad	Area	Salario	**Bonificacion**
0	Cristian	32	Ventas	3400	**119.00**
1	Violeta	28	RRHH	4300	**150.50**
2	Armando	35	Finanzas	4800	**168.00**
3	Mariela	29	RRHH	3850	**134.75**

	Nombre	Edad	Area	**Salario**	Bonificacion
2	Armando	35	Finanzas	**4800**	168.00
1	Violeta	28	RRHH	**4300**	150.50
3	Mariela	29	RRHH	**3850**	134.75
0	Cristian	32	Ventas	**3400**	119.00

Conclusión del uso Pandas en Python.

- Pandas es una herramienta poderosa para trabajar con datos tabulares. La creación y manipulación de DataFrames permite analizar y transformar datos de manera eficiente.
- Aprender las operaciones básicas como selección, filtrado y cálculo es fundamental para abordar proyectos de análisis de datos en Python.

PRÁCTICA – Sesión 4

- **Ejercicio 01**: Crear un array NumPy y realizar operaciones de suma, resta y promedio.

11	32	3	21	4	52	13	8	72	69

PRÁCTICA – Sesión 4

Universidad Tecnológica del Perú

- Solución:

```python
import numpy as np

# Creamos un array de 10 números aleatorios entre 1-100
arr_aleatorios = np.random.randint(1, 100, 10)
print("arr_aleatorios: ", arr_aleatorios)

# Sumando todos los valores del arreglo
sum = np.sum(arr_aleatorios)
print("Suma total: ", sum)

# Sumando solo los valores de índices pares
arr_indices_pares = arr_aleatorios[0:len(arr_aleatorios):2]
print("Suma de valores en índices pares: ", \
      arr_indices_pares, "=", np.sum(arr_indices_pares))

# Hallamos el promedio de los valores
print("Valor promedio:", np.mean(arr_aleatorios))
```

```
arr_aleatorios:
[57 64 58 84  2 31 58 90 56 46]


Suma total:  546



Suma de valores en índices pares:
[57 58  2 58 56] = 231

Valor promedio: 54.6
```

PRÁCTICA – Sesión 4

- Solución *(continuación)*:

```
# Creamos un segundo array de 10 aleatorios de 1-100
arr_aleatorios2 = np.random.randint(1, 100, 10)
print("arr_aleatorios2: ", arr_aleatorios2)

# Sumando los arreglos
print("Suma de arreglos: ", \
      arr_aleatorios + arr_aleatorios2)

# Restando los arreglos
print("Resta de arreglos: ", \
      arr_aleatorios - arr_aleatorios2)

# Multiplicando los arreglos
print("Producto de arreglos: ", \
      arr_aleatorios * arr_aleatorios2)
```

```
arr_aleatorios:
[57 64 58 84  2 31 58 90 56 46]
```

```
arr_aleatorios2:
[19 48 55 93 28 82 55 52  8  4]

Suma de arreglos:
[ 76 112 113 177  30 113 113 142  64  50]

Resta de arreglos:
[ 38  16   3  -9 -26 -51   3  38  48  42]

Producto de arreglos:
[1083 3072 3190 7812 56 2542 3190 4680 448 184]
```

PRÁCTICA – Sesión 4

- **Ejercicio 02**: Usa NumPy para generar una matriz de números aleatorios y calcular su media.

11	32	3
45	85	60
29	4	33

PRÁCTICA – Sesión 4

Universidad Tecnológica del Perú

- Solución:

```python
import numpy as np

# Creamos una matriz 4x4 de enteros entre 1-100
matriz = np.random.randint(1, 100, (4, 4))
print(matriz)

# Media de toda la matriz
print("Media de la matriz:", np.mean(matriz))

# Media por columna
print("Media por columna:", np.mean(matriz, axis = 0))

# Media por fila
print("Media por fila:", np.mean(matriz, axis = 1))
```

```
[[ 8 71  3 68]
 [86 27 36 72]
 [ 7 94 67 70]
 [21 62  6 60]]

Media de la matriz: 47.375


Media por columna:
[30.5 63.5 28.  67.5]

Media por fila:
[37.5  55.25 59.5  37.25]
```

PRÁCTICA – Sesión 4

- **Ejercicio 03**: Crear un DataFrame a partir del diccionario mostrado y realizar los siguientes filtros:

```
dic_lenguajes = {
    "lenguaje": ["Python", "JavaScript", "Java", "C#", "C++"],
    "creador": ["Guido van Rossum", "Brendan Eich", "James Gosling", \
                "Anders Hejlsberg", "Bjarne Stroustrup"],
    "lanzamiento": [1991, 1995, 1995, 2000, 1985],
    "version": ["3.10.5", "ECMAScript 2021", "22", "10.0", "C++20"],
    "ranking": [1, 2, 3, 5, 4],
    "usuarios": [8000000, 12000000, 7000000, 6000000, 6000000]
}
```

a) Mostrar información de los lenguajes que tienen 7M de usuarios o menos.
b) Mostrar información de los lenguajes creados antes del año 2000.
c) Mostrar información de los lenguajes creados por personas cuyo nombre inicia con “B”.
d) Mostrar información de los lenguajes cuyo nombre inicia con “C” o “J”

PRÁCTICA – Sesión 4

- Solución:

```
df = pd.DataFrame(dic_lenguajes)
print(df)
```

```
     lenguaje            creador  lanzamiento          version  ranking  usuarios
0      Python   Guido van Rossum         1991            3.10.5        1   8000000
1  JavaScript       Brendan Eich         1995  ECMAScript 2021        2  12000000
2        Java      James Gosling         1995               22        3   7000000
3          C#   Anders Hejlsberg         2000             10.0        5   6000000
4         C++  Bjarne Stroustrup         1985             C++20        4   6000000
```

PRÁCTICA – Sesión 4

Universidad Tecnológica del Perú

Filtrados:

a) Mostrar información de los lenguajes que tienen 7M de usuarios o menos.

```
print(df[df["usuarios"] <= 7000000])
```

b) Mostrar información de los lenguajes creados antes del año 2000.

```
print(df[df["lanzamiento"] < 2000])
```

c) Mostrar información de los lenguajes creados por personas cuyo nombre inicia con “B”.

```
print(df[df["creador"].astype(str).str[0] == "B"])
```

d) Mostrar información de los lenguajes cuyo nombre inicia con “C” o “J”

```
print(df[df["lenguaje"].str.startswith(("C","J"))])
```

PRÁCTICA – Sesión 4

- **Ejercicio 04**: Realizar las siguientes operaciones *sum()* o *mean()* en columnas específicas del DataFrame mostrado:
  a. Totalizar todas las columnas de la tabla, excepto "equipo“.
  b. Obtener la cantidad total de goles a favor obtenida por los equipos.
  c. Obtener el promedio de partidos ganados.
  d. Obtener el promedio de goles en contra de los 3 primeros equipos.

	equipo	jugados	ganados	empatados	perdidos	goles_favor	goles_contra	dif_goles	puntos
0	Liverpool	22	16	5	1	54	21	33	53
1	Arsenal	23	13	8	2	44	21	23	47
2	Nottm Forest	23	13	5	5	33	27	6	44
3	Man City	23	12	5	6	47	30	17	41
4	Newcastle	23	12	5	6	41	27	14	41
5	Chelsea	23	11	7	5	45	30	15	40

PRÁCTICA – Sesión 4

Universidad Tecnológica del Perú

- Solución:

```
dic_liga_premier = pd.DataFrame({
    "equipo": ["Liverpool", "Arsenal", "Nottm Forest", "Man City", "Newcastle", "Chelsea"],
    "jugados": [22, 23, 23, 23, 23, 23],
    "ganados": [16, 13, 13, 12, 12, 11],
    "empatados": [5, 8, 5, 5, 5, 7],
    "perdidos": [1, 2, 5, 6, 6, 5],
    "goles_favor": [54, 44, 33, 47, 41, 45],
    "goles_contra": [21, 21, 27, 30, 27, 30],
    "dif_goles": [33, 23, 6, 17, 14, 15],
    "puntos": [53, 47, 44, 41, 41, 40]
})

df = pd.DataFrame(dic_liga_premier)
print(df)
```

PRÁCTICA – Sesión 4

a. Totalizar todas las columnas de la tabla, excepto “equipo”.

```
print("Columnas totalizadas:\n", df.drop("equipo", axis = 1).sum())
```

b. Obtener la cantidad total de goles a favor obtenida por los equipos.

```
print("Total de goles a favor:", df["goles_favor"].sum())
```

c. Obtener el promedio de partidos ganados.

```
print("Promedio de partidos ganados:", round(df["ganados"].mean()))
```

d. Obtener el promedio de goles en contra de los 3 primeros equipos.

```
print("Promedio de goles en contra (3 primeros equipos):", \
          df.head(3)["goles_contra"].mean())
```

Espacio Práctico (tarea) – Sesión 4

Universidad Tecnológica del Perú

¡Ahora, inténtalo tú!

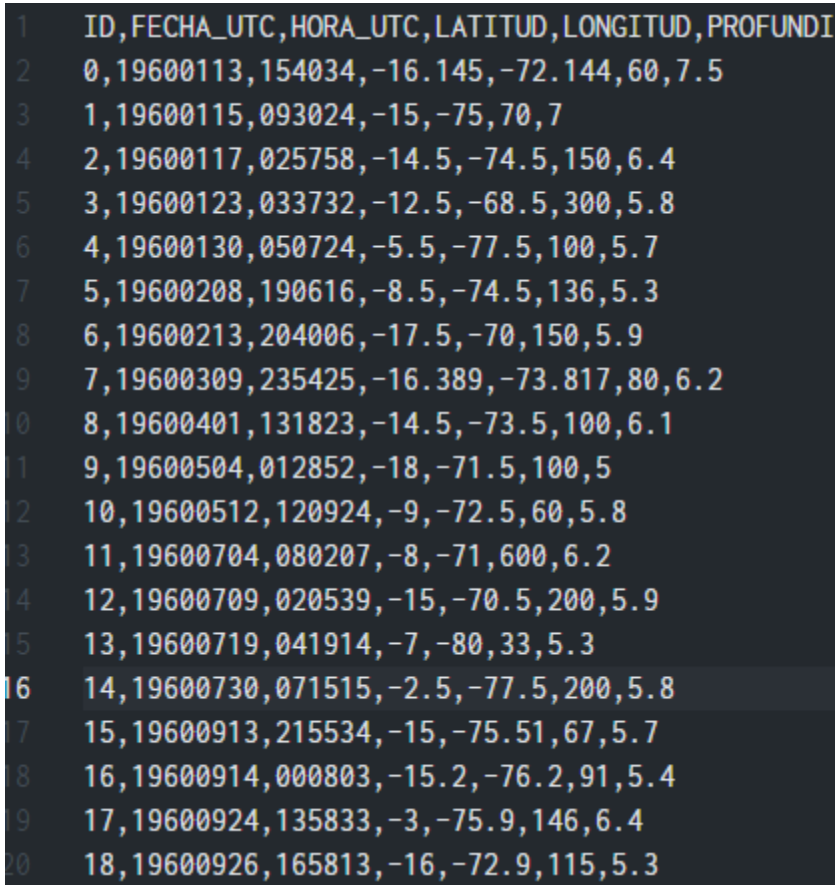
Ejercicio propuesto:

Carga el archivo *sismos.csv* en un DataFrame y escribe el código necesario para satisfacer los siguientes requerimientos:

a. Calcular el número de sismos de al menos 6 grados de magnitud.

b. Listar los sismos de magnitud entre 7 y 8 grados con una profundidad de 60 km o menos.

c. Desafío: Contar el número de sismos por cada año desde el 2015.



```
ID,FECHA_UTC,HORA_UTC,LATITUD,LONGITUD,PROFUNDI
0,19600113,154034,-16.145,-72.144,60,7.5
1,19600115,093024,-15,-75,70,7
2,19600117,025758,-14.5,-74.5,150,6.4
3,19600123,033732,-12.5,-68.5,300,5.8
4,19600130,050724,-5.5,-77.5,100,5.7
5,19600208,190616,-8.5,-74.5,136,5.3
6,19600213,204006,-17.5,-70,150,5.9
7,19600309,235425,-16.389,-73.817,80,6.2
8,19600401,131823,-14.5,-73.5,100,6.1
9,19600504,012852,-18,-71.5,100,5
10,19600512,120924,-9,-72.5,60,5.8
11,19600704,080207,-8,-71,600,6.2
12,19600709,020539,-15,-70.5,200,5.9
13,19600719,041914,-7,-80,33,5.3
14,19600730,071515,-2.5,-77.5,200,5.8
15,19600913,215534,-15,-75.51,67,5.7
16,19600914,000803,-15.2,-76.2,91,5.4
17,19600924,135833,-3,-75.9,146,6.4
18,19600926,165813,-16,-72.9,115,5.3
```

Espacio Práctico (tarea) – Sesión 4

Universidad Tecnológica del Perú

Solución:

a. Calcular el número de sismos de al menos 6 grados de magnitud.

```
print(len(df[df["MAGNITUD"] >= 6.0]))
```

b. Listar los sismos de magnitud entre 7 y 8 grados con una profundidad de 60 km o menos.

```
print(df[(df["MAGNITUD"].between(7, 8)) & (df["PROFUNDIDAD"] <= 60)])
```

c. Desafío: Contar el número de sismos por cada año desde el 2015.

```
# a. Creamos una nueva columna solo con el año del evento sísmico
df['AÑO_SISMO'] = df["FECHA_UTC"].astype(str).str[:4].astype(int)
# b. Agrupamos las filas por año de sismo y obtenemos su tamaño (longitud)
print(df[df["AÑO_SISMO"] >= 2015].groupby("AÑO_SISMO").size())
```

Cierre - Sesión 02

Universidad Tecnológica del Perú

Proporciona ejemplos de creación de arrays con NumPy

Proporciona ejemplos de creación de DataFrames con Pandas

Gracias por su atención