

guayerd

# Fundamentos IA

## Análisis con Python

Clase 6

En colaboración con  
**IBM SkillsBuild**





# ¡Bienvenidos!

¿Nos presentamos?

- ¿Qué recuerdan de la clase anterior?
- ¿Qué esperan aprender?
- ¿Tienen alguna pregunta?

# Contenidos

Por temas

05

- Copilot Chat y prompts
- Demo asincrónica

07

- Estadística aplicada

06

- Limpieza y transformación

08

- Visualización

# Objetivos de la clase



- Pandas
- Lectura de archivos
- Estructuras principales
- Inspección y limpieza

# Análisis con Python

Limpieza y transformación

# Plataforma Skill Build: Python



eLearning


## Data Visualization with Python

3 horas  1.849 ★★★★★ 150



eLearning

## Utilizar la IA generativa para el desarrollo de software

1 hora  34.080 ★★★★★ 2.316

# Limpieza y transformación de datos

## Etapa 3 del ciclo de vida del dato

Proceso técnico para **preparar datos antes del análisis**.

- Elimina errores, inconsistencias y valores irrelevantes
- Estandariza formatos y estructuras
- Mejora la calidad y utilidad de los datos

Estandarizar = criterios

# Formatos comunes

- **CSV:** datos tabulares separados por comas
- **JSON:** objetos anidados y flexibles
- **Excel:** hojas de cálculo
- **Bases de datos:** estructuras relacionales



**Pandas integra el trabajo** con todos estos formatos



## ¿Qué es?

Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos.



# Principales características

- Estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente archivos en formato CSV, Excel y bases de datos SQL.
- 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.
- Realiza todas estas operaciones de manera muy eficiente.

# Pandas

(Panel Data)

## Comandos

- **Instalación:** `pip install pandas`
- **Uso:** `import pandas as pd`

# Estructuras principales

## Series (.s)

- Arreglo unidimensional
- Compuesta por índices y valores
- Similar a una columna

## DataFrame (.df)

- Tabla bidimensional
- Colección de Series alineadas por índice
- Base para análisis y transformación

The diagram illustrates the relationship between Series and DataFrame. It shows two Series being added together to form a DataFrame. The first Series, labeled 'Apples', has values [0, 1, 2] for indices [0, 1, 2]. The second Series, labeled 'Bananas', has values [2, 6, 3] for indices [0, 1, 2]. The resulting DataFrame has two columns, 'Apples' and 'Bananas', with the values from the two Series aligned by index.

Apples	
0	1
1	3
2	7
Series	

+

Bananas	
0	2
1	6
2	3
Series	

=

Apples	Bananas
1	2
3	6
7	3
DataFrame	

# Lectura de archivos

## CSV

- `df_csv = pd.read_csv("nombre_archivo.csv")`
- Útil: `encoding='utf-8', sep=';`

## JSON

- `df_json = pd.read_json("nombre_archivo.json")`
- Para listas: `orient='records'`

## Excel

- `df_xls = pd.read_excel("nombre_archivo.xlsx")`
- Especificar hoja: `sheet_name="nombre_hoja"`

Siempre devuelve un DataFrame listo para trabajar

# Inspección inicial

Aspecto	Comando	Descripción
Estructura	<code>df.shape</code>	Dimensiones del conjunto
Tipos	<code>df.dtypes</code>	Variables numéricas y categóricas
Complejidad	<code>df.isnull().sum()</code>	Valores faltantes
Muestra	<code>df.head()</code>	Primeros registros
Resumen	<code>df.info()</code>	Información general

# Desafíos comunes en datos

- **Valores faltantes:** celdas vacías o NaN
- **Duplicados:** registros repetidos exactos
- **Inconsistencias:** formatos diferentes para mismo dato
- **Valores atípicos:** datos extremos que sesgan análisis
- **Tipos incorrectos:** números como texto, fechas mal formateadas



## Qué son las Series en Pandas ?

- Estructuras similares a los arrays de una dimensión.
- Sus elementos tienen que ser del mismo tipo, y su tamaño es inmutable, es decir, no se puede cambiar, aunque sí su contenido.
- Dispone de un índice que asocia un nombre a cada elemento de la serie, a través de la cuál se accede al elemento.



# Crear series

```
>>> import pandas as pd
>>> s = pd.Series(['Matemáticas', 'Historia', 'Economía', 'Programación', 'Inglés'], dtype='string')
>>> print(s)
0    Matemáticas
1      Historia
2     Economía
3  Programación
4       Inglés
dtype: string
```

A partir de  
una lista

A partir de un  
diccionario

```
>>> import pandas as pd
>>> s = pd.Series({'Matemáticas': 6.0, 'Economía': 4.5, 'Programación': 8.5})
>>> print(s)
Matemáticas    6.0
Economía       4.5
Programación   8.5
dtype: float64
```

## DataFrame en Pandas

- Define un conjunto de datos estructurado en forma de tabla donde cada columna es un objeto de tipo Series, 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.
- Se puede acceder a sus elementos mediante los nombres de las filas y las columnas.

# Ejemplo:

The diagram illustrates a table structure with labels and indices. The table has 4 rows and 4 columns. The first row contains headers: 'Nombre', 'Edad', 'Grado', and 'Correo'. The first column contains row indices: '1', '2', '3', and '4'. Red arrows point from the labels 'Nombres Filas' and 'Columnas' to the first column and first row respectively. Another set of red arrows points from 'Nombres Columnas' and 'Filas' to the first row and first column respectively. The entire table is enclosed in a red dashed border.

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

# Crear DataFrames

```
>>> import pandas as pd
>>> df = pd.DataFrame(['María', 18], ['Luis', 22], ['Carmen', 20]), columns=['Nombre', 'Edad'])
>>> print(df)
```

	Nombre	Edad
0	María	18
1	Luis	22
2	Carmen	20

**A partir de  
una lista**

**A partir de una lista de diccionarios**

```
>>> import pandas as pd
>>> df = pd.DataFrame([{'Nombre':'María', 'Edad':18}, {'Nombre':'Luis', 'Edad':22}, {'Nombre':'Carmen'}])
>>> print(df)
```

0	María	18.0
1	Luis	22.0
2	Carmen	NaN

# Crear DataFrames a partir de un array:

```
>>> import pandas as pd
>>> df = pd.DataFrame(np.random.randn(4, 3), columns=['a', 'b', 'c'])
>>> print(df)
```

	a	b	c
0	-1.408238	0.644706	1.077434
1	-0.279264	-0.249229	1.019137
2	-0.805470	-0.629498	0.935066
3	0.236936	-0.431673	-0.177379

## Crear DataFrames a partir de un archivo .csv o Excel

```
>>> df = pd.read_csv('colesterol.csv', sep=';', decimal=',')  
>>> print(df.head())
```

	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	1.70	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0

- `read_csv(fichero.csv, sep=separador, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal)`
- `read_excel(fichero.xlsx, sheet_name=hoja, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal)`

# ¿Qué harías en cada situación?



- Dataset con 10% de valores faltantes distribuidos aleatoriamente
- Registros de clientes con emails duplicados pero datos diferentes
- Precios con valores negativos en sistema de inventario
- Fechas en formatos: "2024-01-15", "15/01/2024", "Jan 15, 2024"

# Tratamiento de valores faltantes

Estrategia	Comando(s)	Descripción
Detección	<code>df.isnull(), df.isna()</code>	Identificar valores faltantes
Eliminación	<code>df.dropna()</code>	Eliminar registros con pocos casos
Valor constante	<code>df.fillna(0)</code>	Rellenar con cero o texto
Valor estadístico	<code>df.fillna(df['columna'].median())</code>	Rellenar con promedio o mediana



# Eliminación de duplicados

Estrategia	Comando(s)	Descripción
Detección	<code>df.duplicated()</code>	Marca filas duplicadas
Eliminación completa	<code>df.drop_duplicates()</code>	Remueve todos los duplicados
Por columnas específicas	<code>df.drop_duplicates(subset=['col1', 'col2'])</code>	Detecta duplicados según columnas elegidas
Conservar primera/última	Parámetro <code>keep</code>	Mantiene el primer o último registro

# Inconsistencias de formato

Tipo de dato	Comando(s) / Acción	Descripción
Texto	<code>.str.lower()</code> , <code>.str.strip()</code>	Normalizar mayúsculas/minúsculas y espacios
Fechas	<code>pd.to_datetime()</code>	Convertir a formato fecha uniforme
Números	<code>pd.to_numeric()</code>	Ajustar separadores decimales
Categorías	—	Estandarizar variaciones del mismo valor

# Manejo de valores atípicos

**Valores extremos que se alejan significativamente** del resto de los datos.

- **Detección visual:** boxplots muestran valores extremos
- **Filtrado por rango:** remover valores fuera de límites lógicos
- **Criterio de dominio:** usar conocimiento del área
- **Verificación manual:** confirmar si son errores o valores reales

# Tipos de datos incorrectos

Estrategia	Comando(s)	Descripción
Verificar tipos	<code>df.dtypes</code>	Identificar tipos actuales
Conversión manual	<code>df.astype()</code>	Cambiar tipo de columna
Fechas	<code>pd.to_datetime()</code>	Convertir a formato fecha
Numéricos	<code>pd.to_numeric()</code>	Forzar conversión numérica

# Transformaciones básicas

Operación	Comando(s)	Descripción
Filtrado	<code>df[condición]</code>	Seleccionar subconjuntos específicos
Agrupación	<code>df.groupby()</code>	Calcular estadísticas por categorías
Ordenamiento	<code>df.sort_values()</code>	Organizar registros por columnas
Selección	<code>df[['col1', 'col2']]</code>	Elegir columnas específicas

# Café de barrio



1. Calcular correlación entre temperatura y ventas
2. Identificar el mes con mejor retorno publicitario
3. Analizar relación personal vs satisfacción cliente
4. Proponer estrategia basada en datos

Mes	Ventas (\$)	Temp (°C)	Publicidad (\$)	Personal	Satisfacción
Ene	15,000	18	800	4	4.2
Feb	22,000	25	1,200	5	4.5
Mar	18,000	22	900	4	4.1
Abr	28,000	28	1,500	6	4.8
May	25,000	30	1,300	5	4.6

# Proyecto

## Tienda Aurelion

- **Documentación:** notebook Markdown
- **Desarrollo técnico:** programa Python
- **Visualización de datos:** dashboard en Power BI
- **Presentación oral:** problema, solución y hallazgos



# Limpieza de datos

## Trabajo en equipo



1. Usar **Copilot para analizar problemas** con dataset
2. **Limpiar la base** de datos
3. **Documentar con Copilot** cada paso aplicado





# Retro

¿Cómo nos vamos?

- ¿Qué fue lo más útil de la clase?
- ¿Qué parte te costó más?
- ¿Qué te gustaría repasar o reforzar?