



Tecnológico  
de Monterrey

| Educación  
Continua

# Plataformas de visualización

Programas LIVE

SS1 | Aprender

# Bienvenida y presentación

# Bienvenida



**“El auténtico genio consiste  
en la capacidad para  
evaluar información incierta,  
aleatoria y contradictoria”**

-Winston Churchill

# Objetivo principal del módulo



- Explotarás las **ventajas** que tienen **Python** y las plataformas de visualización **Matplotlib**, **Pandas** y **Seaborn**, para generar diversos tipos de gráficos que permitan analizar e interpretar **resúmenes de datos** del mundo real, cumpliendo con los requerimientos de la **interfaz de la visualización** requerida

# Objetivos particulares de la sesión



- Analizarás los **datos** provenientes de varios archivos para comprender su estructura y distribución
- Resumirás la información usando **representaciones gráficas** para dar **significado a los datos e identificar tendencias y relaciones** entre variables
- Crearás **gráficas superpuestas** que te permitan hacer análisis comparativos y de variabilidad

# Agenda

1

**Sesión Síncrona 1**

Aprender

2

**Trabajo asíncrono 1**

Profundizar | Ruta de Aprendizaje

3

**Sesión Síncrona 2**

Preparar para Aplicar

4

**Trabajo asíncrono 2**

Aplicar en el trabajo | Reto



2



3



4



# Dinámica de participación y convivencia

## Sesión grupal:

- Las dudas durante la sesión se manejarán a través del chat. Éstas serán respondidas en los momentos definidos para ello
- Participar en las dinámicas propuestas durante la sesión de acuerdo con las instrucciones dadas para cada una

## Rooms:

- Los equipos se harán de manera aleatoria y serán sólo para las actividades de práctica
- Estar atentos a la notificación de zoom para unirse al room
- Seguir al pie de la letra las instrucciones para la dinámica en rooms
- Informar al moderador si te encuentras sólo en el room para reasignarte
- Te pedimos no salir del room que tienes asignado. Cualquier reasignación será realizada por parte del tutor o staff



# Introducción

- El análisis de grandes cantidades de información en formato **tabular dificulta su comprensión**. Hay tendencias y cambios que pueden pasar desapercibidos
- La **visualización de datos** aporta significado a los números y contribuye a la identificación de **patrones** para la toma de decisiones



Tecnológico  
de Monterrey

Educación  
Continua

# Panorámica de la sesión

1

Sesión Síncrona 1  
Aprender

Tema 1: Exploración y validación de datos

Tema 2: Plataformas de visualización en Python

Tema 3: Gráficas superpuestas

Cierre de sesión

# Panorámica de la sesión

1

Sesión Síncrona 1  
Aprender

Tema 1: Exploración y validación de datos

Tema 2: Plataformas de visualización en Python

Tema 3: Gráficas superpuestas

Cierre de sesión

# Propósito | Tema 1



- La cantidad de información a la que tenemos acceso hoy en día proviene de **múltiples y diversas fuentes** u orígenes: desde archivos de texto, en formato tabular, imágenes, hasta bases de datos con contenido estructurado o no estructurado

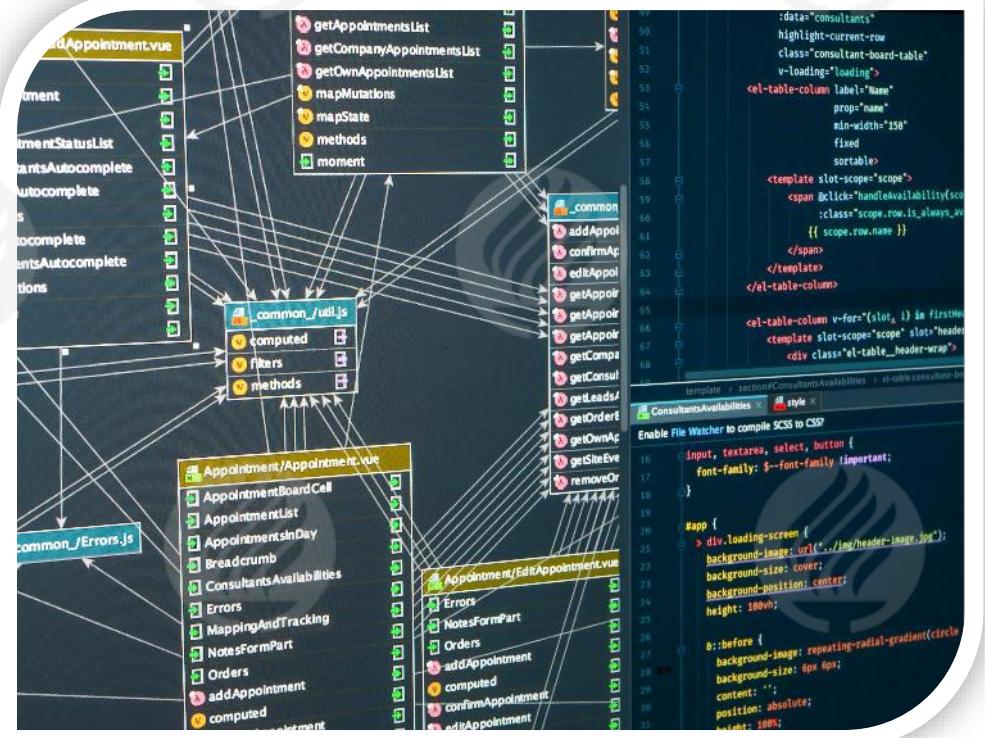
# Propósito | Tema 1



- La etapa de **exploración** nos permite **aprender de los datos**, identificando:
  - ✓ Rangos típicos
  - ✓ Categorías
  - ✓ Distribuciones
  - ✓ Estadísticas entre variables u observaciones
- Es fundamental en un proyecto de ciencia de datos porque puede dar **pistas iniciales** que contribuyan en los modelos de aprendizaje automático

# Rangos típicos, categorías y valores faltantes |

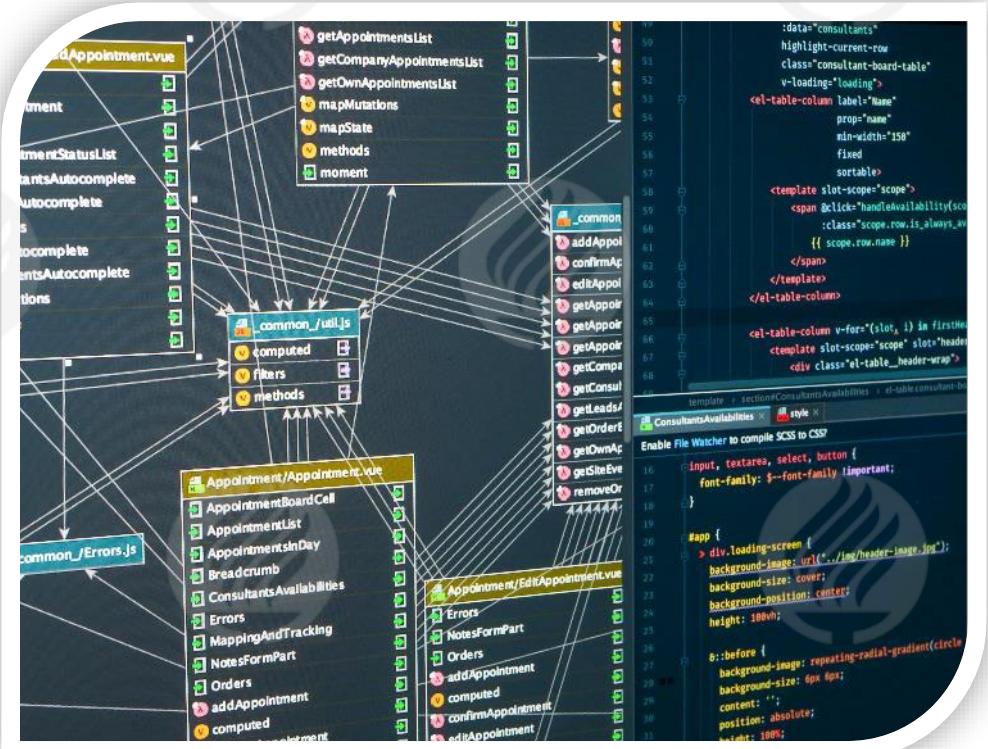
## Tema 1.1



- Cuando se inicia un proyecto de ciencia de datos, la **exploración** inicial tiene como objetivo responder las siguientes preguntas:
- ¿Cómo están **organizados** los datos?
  - ✓ Archivos de texto (delimitados, no estructurados, XML)
  - ✓ Bases de datos (esquema)
  - ✓ Web/Streaming (JSON)

# Rangos típicos, categorías y valores faltantes |

## Tema 1.1



- ¿Qué **tipo de datos** se incluyen?
  - ✓ Texto (forma libre, categóricos)
  - ✓ Numéricos (enteros, flotantes)
  - ✓ Imagen, video, audio
- ¿Qué **problemas** tienen los datos?
  - ✓ Datos faltantes o dañados
  - ✓ Variables y observaciones duplicadas
  - ✓ Dificultades con el acceso a los datos

# Rangos típicos, categorías y valores faltantes |

## Tema 1.1

-487.75391	13905.5459	22688.64063	68905.16406
21547.26563	13647.66895	22752.72852	68509.95313
21448.08203	13737.69727	22740.51953	68529.79688
21567.10156	13655.29883	23025.86328	68868.54688
21133.74609	13580.5293	22630.65625	68072.02344
21316.85352	13640.04102	22908.36914	68232.24219
21623.56055	13589.68457	22737.46875	68628.97656
21234.45508	13606.46973	22720.68359	67975.89063
21304.64648	13545.43359	22734.41602	68285.64844
21756.3125	13690.39355	22783.24609	68813.60938
21397.72656	13588.15918	22752.72852	68352.78906
21599.14453	13415.73242	22711.52734	68166.625
21477.07422	13763.6377	22952.62109	68647.28906
21516.74609	13699.5498	22566.56836	68465.70313
21376.36328	13687.34277	22557.41211	68337.53125
21510.64258	13891.8125	22606.24023	68676.28125
21915.00586	13488.97656	23001.44727	69141.67969
21394.67383	13522.54492	22865.64453	68439.765

- La estructura `dataframe` de **Pandas** tiene muchas similitudes con los formatos tabulares más empleados, por tanto, será el punto de partida para el análisis exploratorio
- Sus atributos más importantes para este fin son:
  - ✓ **shape** - Dimensionalidad del `dataframe`
  - ✓ **columns** - Etiquetas de las columnas
  - ✓ **index** - Etiquetas de las filas
  - ✓ **dtypes** - Tipos de datos

# Rangos típicos, categorías y valores faltantes |

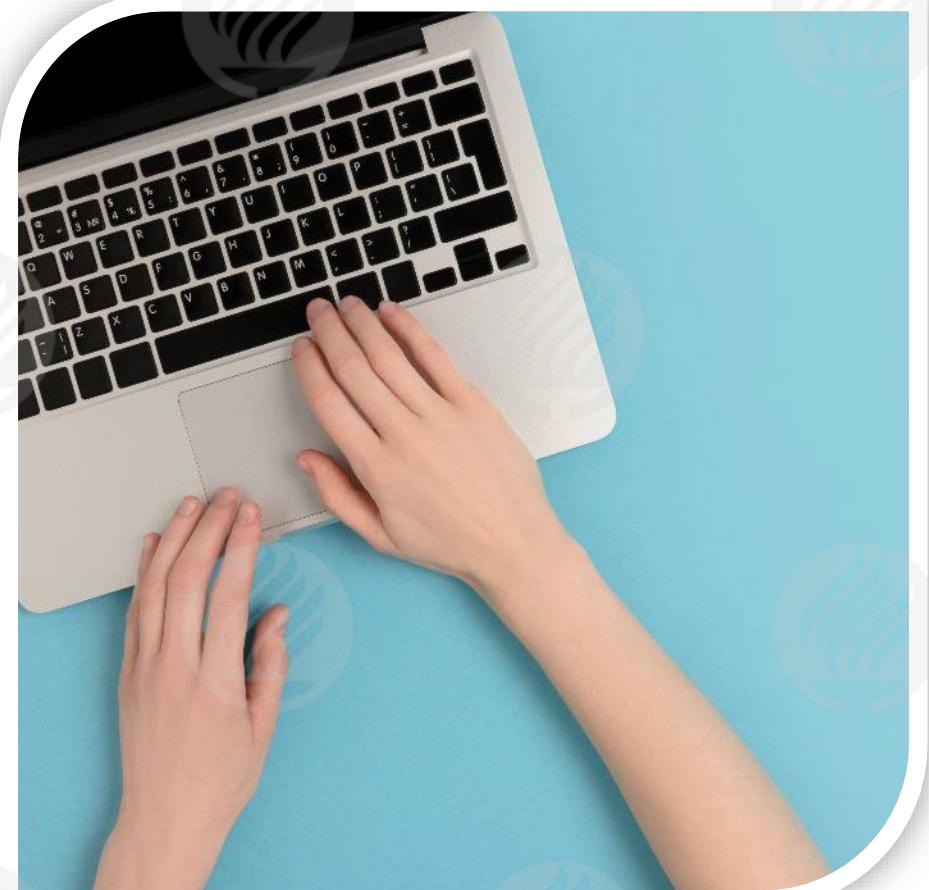
## Tema 1.1



Algunos **métodos de utilidad** para explorar el contenido del dataframe son:

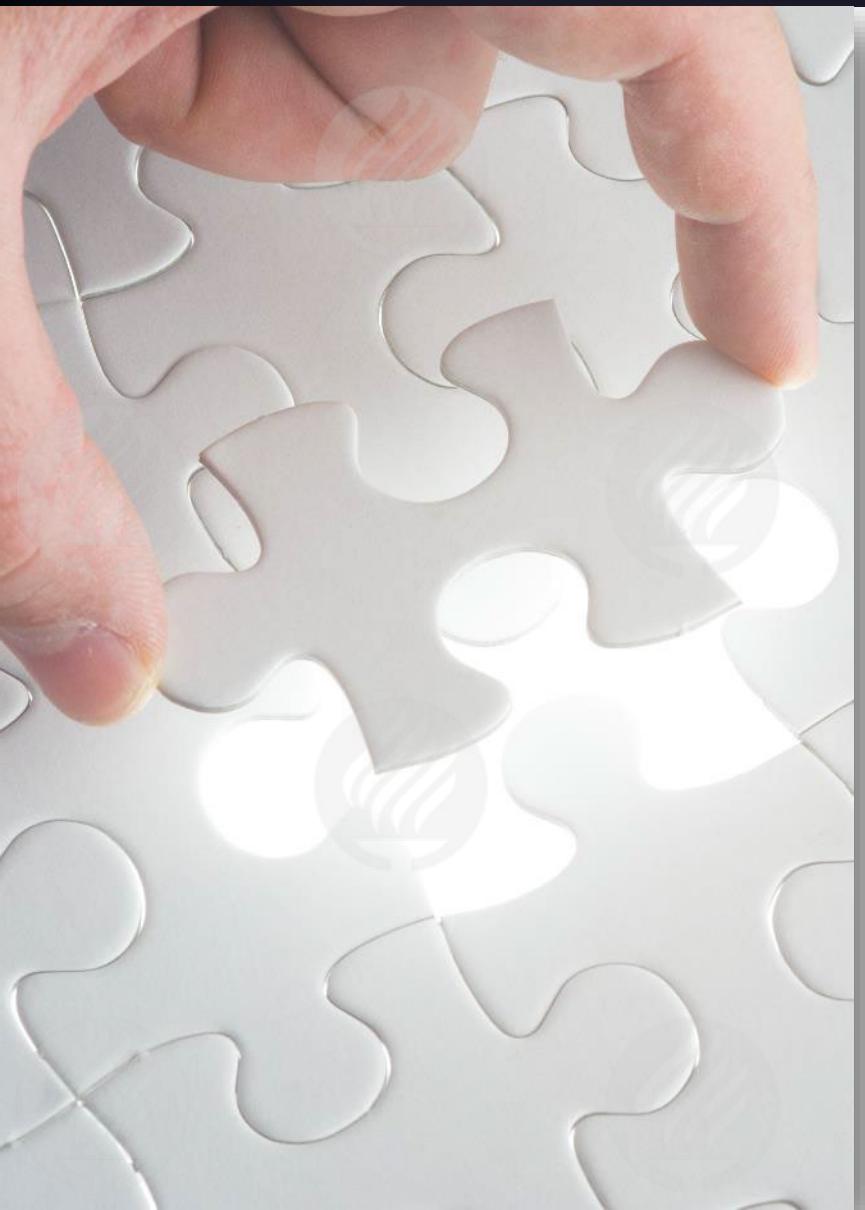
- **head()** - Regresa los primeros registros
- **nunique()** - Cuenta observaciones distintas sobre un eje
- **value\_counts()** - Devuelve una serie que contiene recuentos de filas únicas
- **isna()** - Detecta valores faltantes

# Actividad | Aprendizaje activo



Sube a tu drive los dos archivos csv que el instructor te proporcionará y crea una nueva libreta en **Colab** para:

- Leer dichos archivos en dos *dataframes*: **countries** y **cities**
- Realizar un análisis de la estructura de countries
- Explorar el contenido de countries
- Borrar los registros que tengan NaN en una columna **Capital**
- Ejecutar el mismo análisis con cities
- Combinar ambos dataframes



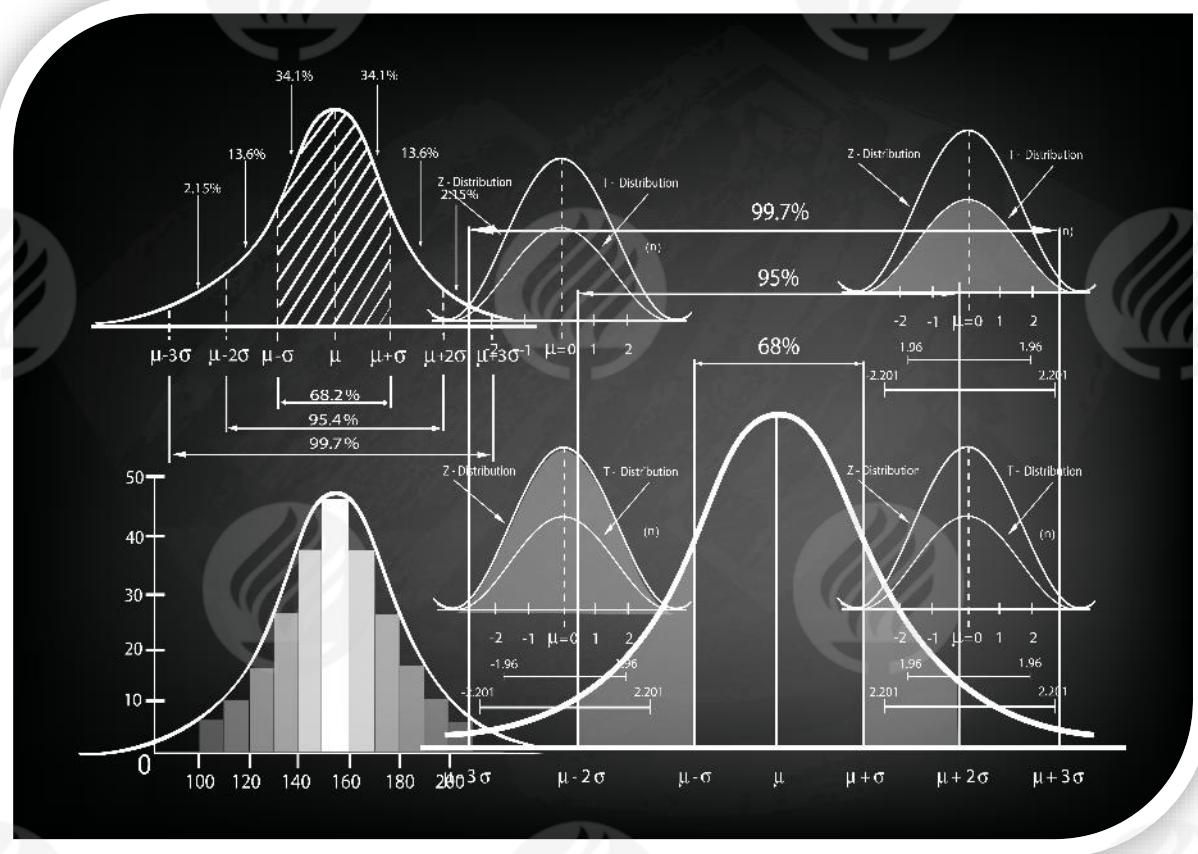
- El conocimiento de la **estructura y contenido** de la información que gestionaste es de vital importancia para su manipulación; permite establecer **conexiones** entre diversas fuentes y fijar una estrategia para tratar con **valores faltantes** o variables que **no aportan información** útil en un dataframe
- Ahora que ya sabes cómo hacer un análisis preliminar de la información, **¿harás una valoración similar con los datos que manejas en tu quehacer diario?**

# Distribuciones y estadísticas | Tema 1.2



- En un módulo previo aprendiste a efectuar **cálculos estadísticos** en una o varias columnas, utilizando funciones como:
  - ✓ **mean()**
  - ✓ **min()**
  - ✓ **max()**
  - ✓ **median()**
- La estructura `dataframe` de Pandas agrupa éstas y otras medidas en la función **describe()**, que genera estadísticas descriptivas para todas las columnas numéricas

# Distribuciones y estadísticas | Tema 1.2

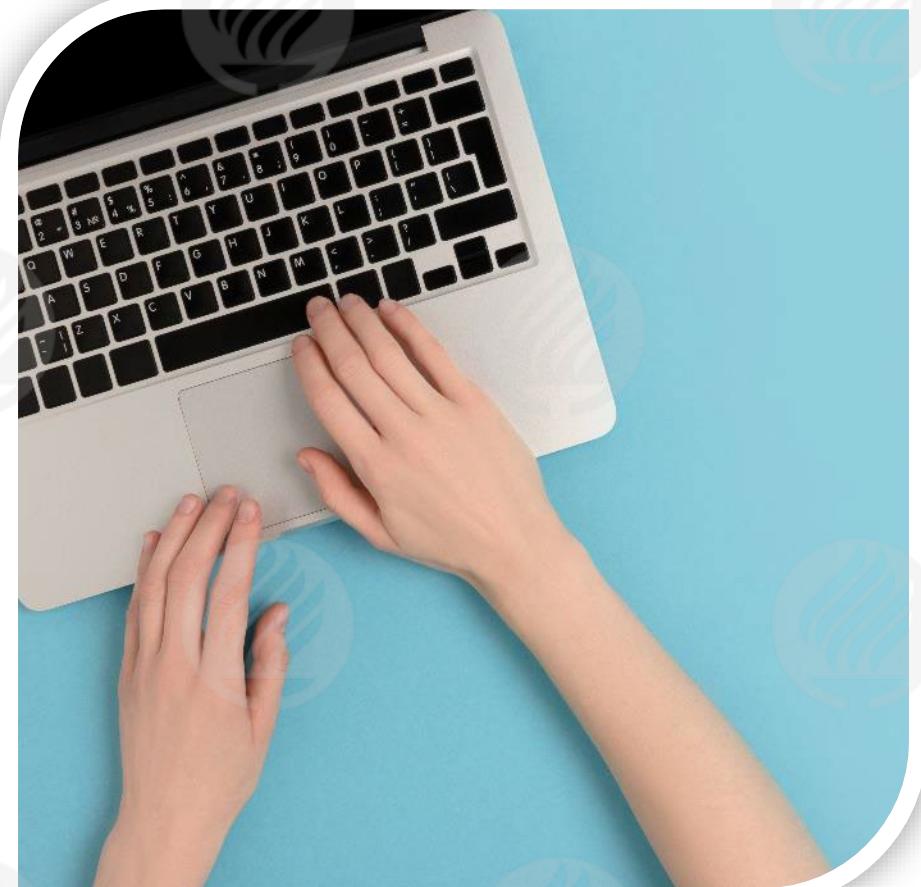


Las **estadísticas descriptivas** incluyen aquellas que, de un conjunto de datos, resumen:

- La **tendencia central** (promedio y mediana)
- La **dispersión** (desviación estándar y cuartiles) y
- La forma de la **distribución** (conteo, mínimo, máximo)

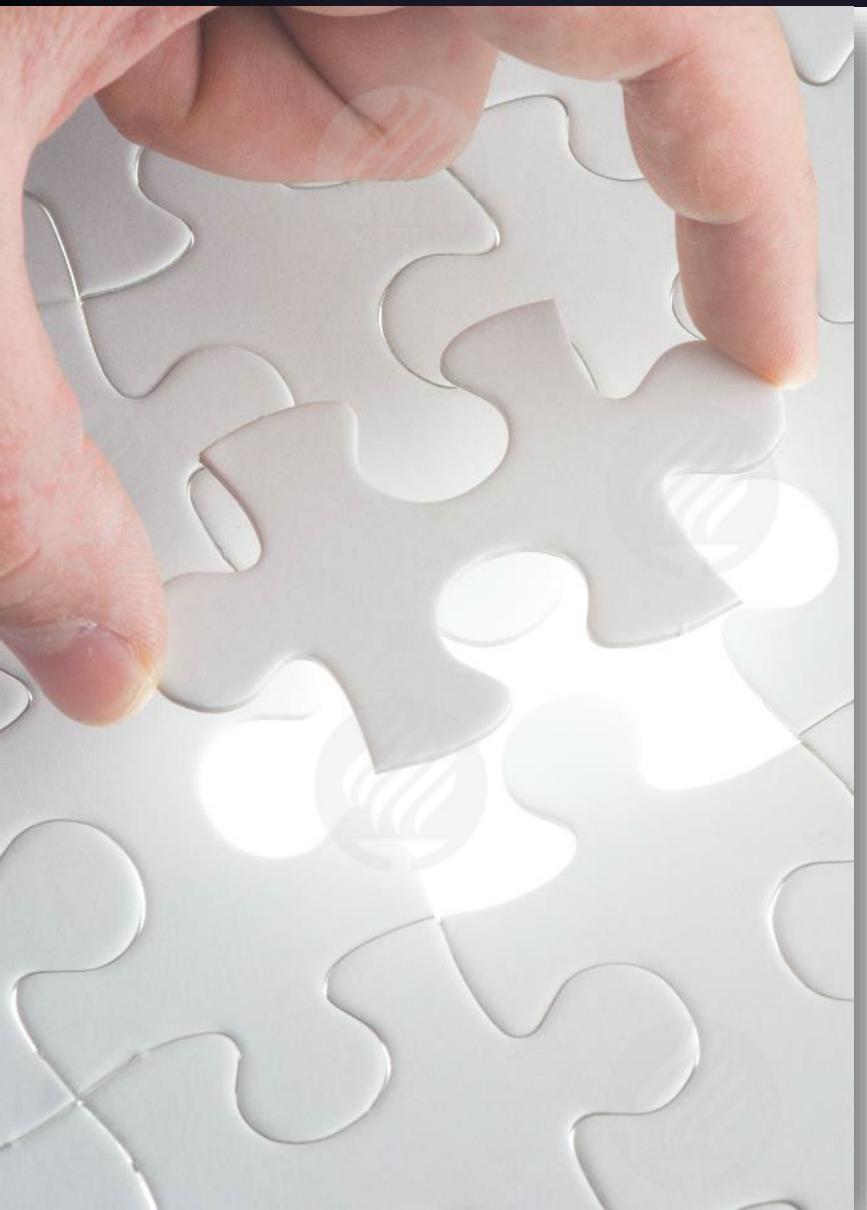
\* Todas las anteriores **excluyen** los valores **Nan**

# Actividad | Aprendizaje activo



En la misma libreta de **Colab** obtén las medidas estadísticas descriptivas de **countries** para responder:

- ¿Cuál es el **promedio de población** para el conjunto de países analizados?
- ¿Qué **superficie territorial** tiene el país más pequeño? ¿Y el más grande?
- ¿Cuál es la **desviación estándar** de la expectativa de vida?
- ¿Qué significa que para el año de independencia está **1974** en el **3er cuartil (75%)**?



- Obtener un conjunto reducido de **valores descriptivos**, como el promedio, el valor máximo y mínimo, la desviación estándar y los cuartiles, te muestran las características o **propiedades principales** de los datos observados
- Piensa en la información que ocupas a diario, **¿conoces estos valores?**
- ¿Qué otra **medida relevante** podrías incorporar al conjunto anterior?

# Cierre: Concepto clave | Tema 1

La fase de **exploración de datos** permite la **caracterización** de la **información**, el análisis de su **comportamiento** y el establecimiento de las estrategias para el manejo de **valores faltantes o irrelevantes**

# Panorámica de la sesión

1

Sesión Síncrona 1  
Aprender

Tema 1: Exploración y validación de datos

Tema 2: Plataformas de visualización en Python

Tema 3: Gráficas superpuestas

Cierre de sesión

# Propósito | Tema 2



- La incapacidad de analizar grandes cantidades de información de manera rápida puede ocasionar **pérdidas económicas** y un **rezago competitivo**, pues los problemas no pueden abordarse oportunamente, ni se reconocen tendencias emergentes
- La visualización de datos permite ver la información de forma **comprendible** y **cohesiva**, facilitando la **identificación de patrones**, la experimentación con **escenarios diversos** y la **comunicación asertiva** de las conclusiones o descubrimientos

# Trazado básico y personalización de la estética |

## Tema 2.1



- El panorama de **visualización** de **Python** puede parecer abrumador al principio
- Incluso se ha creado **PyViz.org**, un sitio para ayudar a los usuarios a decidir cuáles son las mejores herramientas de visualización de código abierto de Python para sus propósitos

<https://pyviz.org/overviews/index.html>

# Trazado básico y personalización de la estética |

## Tema 2.1



- **Matplotlib** es una de las denominadas **bibliotecas núcleo** (core), sobre la que se construyen varias plataformas de nivel superior.
  - Tiene una API completa y potente, que permite personalizar cualquier atributo de la figura
- <https://matplotlib.org>
- Las plataformas de **alto nivel** que ocupan Matplotlib, proporcionan una API más simple para cubrir las tareas más comunes de manera concisa y conveniente

# Trazado básico y personalización de la estética |

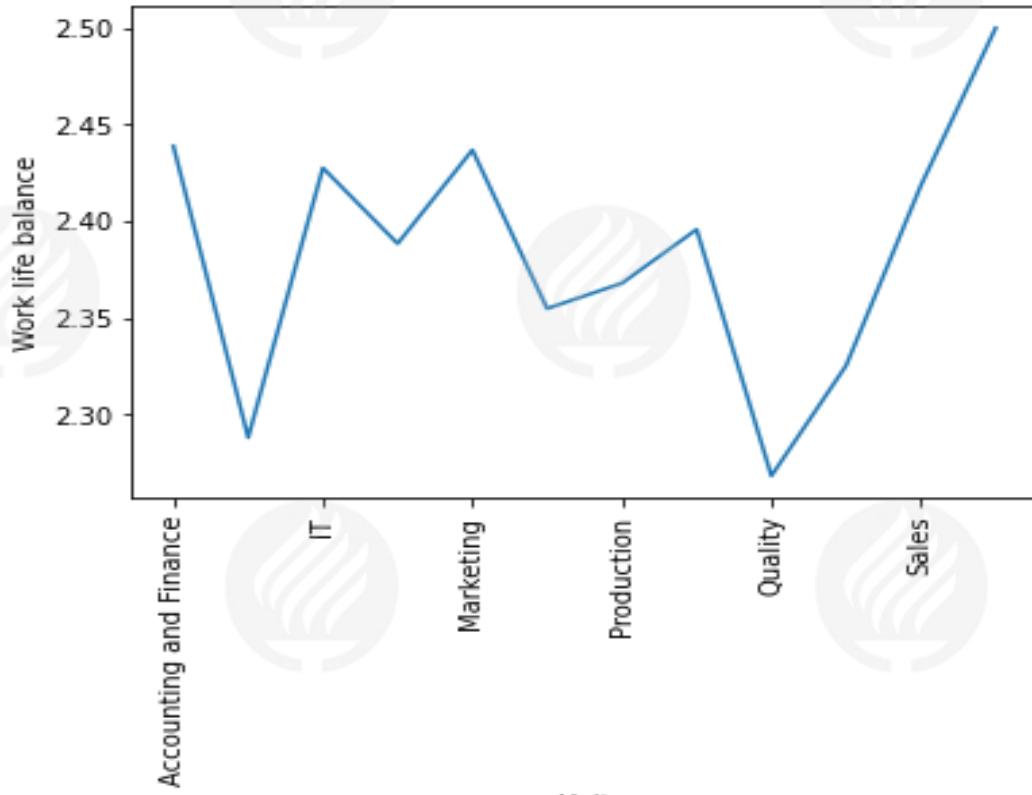
## Tema 2.1



- Una de las más antiguas es la API `.plot ()` de **Pandas**. Esta interfaz renderiza gráficos estáticos en libretas de Jupyter o para exportar desde Python, con un comando que puede ser tan simple como `df.plot ()` para un **dataframe** con dos columnas  
[https://pandas.pydata.org/docs/user\\_guide/visualization.html](https://pandas.pydata.org/docs/user_guide/visualization.html)
- **Seaborn** es otra plataforma basada en matplotlib, para dibujar **gráficos estadísticos** atractivos e informativos. Ésta se integra estrechamente con las estructuras de datos de Pandas.  
<https://seaborn.pydata.org>

# Trazado básico y personalización de la estética |

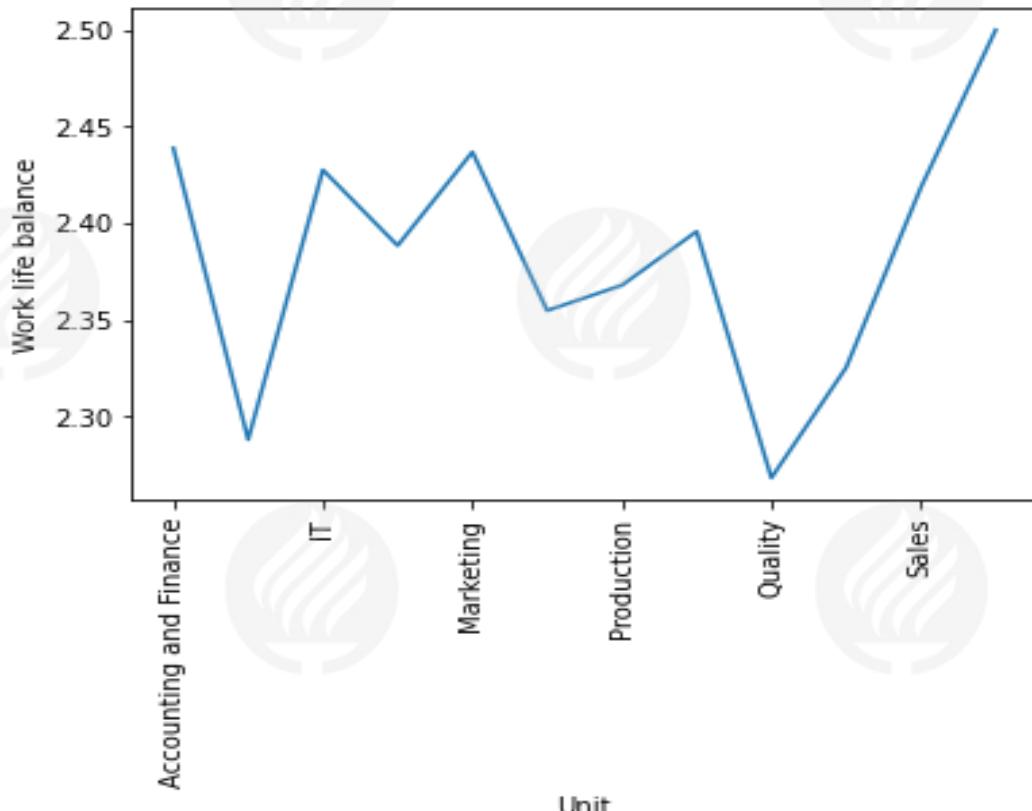
## Tema 2.1



- Para utilizar estas plataformas debes **importarlas** en los scripts de Python
  - ✓ `import matplotlib.pyplot as plt`
  - ✓ `import pandas as pd`
  - ✓ `import seaborn as sns`
- Y a través de ellas llamar a los **métodos** o atributos

# Trazado básico y personalización de la estética |

## Tema 2.1



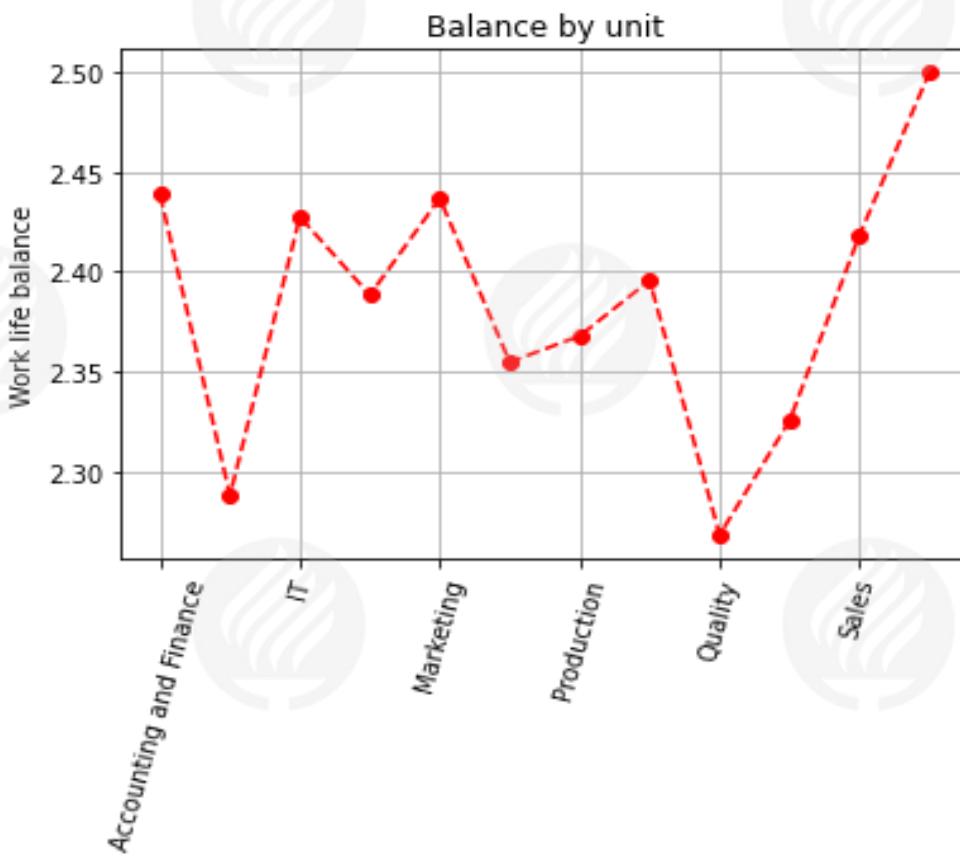
Por ejemplo, la función de graficado básica es **plot(x,y)**, que grafica valores de y contra x como líneas y/o marcadores

- **Con Matplotlib:** `plt.plot(df.index, df.values)`
- **Con Pandas:** `df.plot()`
- **Con Seaborn:**  
`sns.lineplot(x=df.index,y=df[column])`

\* Considerando un dataframe df de sólo **dos columnas**: una de índice y otra de valores

# Trazado básico y personalización de la estética |

## Tema 2.1



- La **personalización** de la estética puede hacerse a través de los argumentos provistos por cada plataforma
  - ✓ **Con Pandas:** kind, figsize, title, grid, legend, xlabel, ylabel, rot, fontsize,...
  - ✓ **Con Seaborn:** hue, size, style,...
- Pero como cualquier gráfico creado por Pandas y Seaborn es un **objeto Matplotlib**, sus métodos plot son sólo un envoltorio (wrapper) de plt.plot() y pueden usarse todos los argumentos de matplotlib
  - ✓ **Con Matplotlib:** color, marker, linestyle,...

# Actividad | Aprendizaje activo



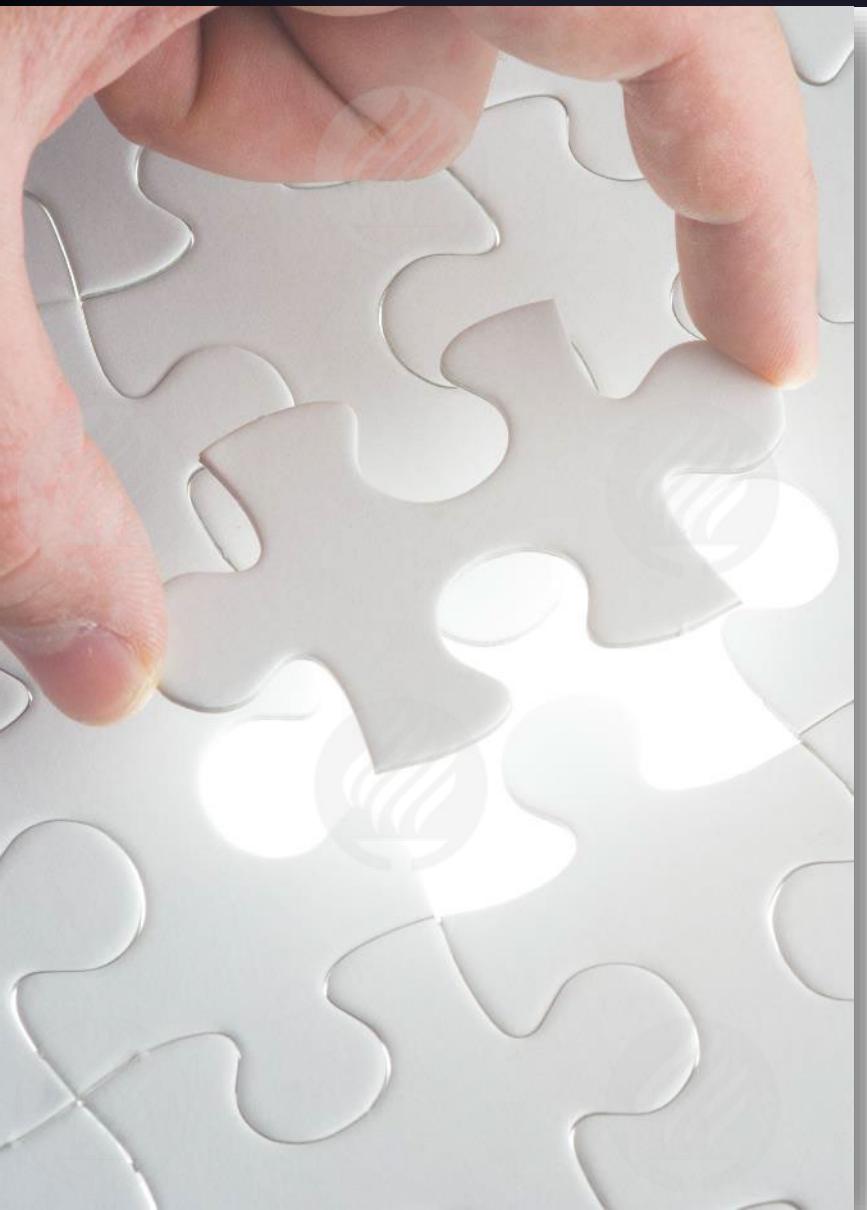
Utiliza la misma libreta de **Colab** para:

- Generar un nuevo dataframe (**sample**) con los 6 primeros registros de countries y dejando únicamente las columnas **Name** y **Population**
- Hacer que **Name** sea el índice
- Visualizar **sample** usando las tres plataformas de trazado
- Cambiar las propiedades: *title*, *grid*, *legend*, *xlabel*, *ylabel*, *rot*, *fontsize*, *color*, *marker* y *linestyle* de la gráfica generada con **Pandas**

# Actividad | Aprendizaje activo



- ¿Cómo **cambian** los códigos anteriores si no se filtran las columnas de Name y Population?
- Probar con otro dataframe (**sample2**) que almacene los 6 primeros registros de countries, pero con todas sus columnas
- Cambiar el tipo de gráfico a barras horizontales usando el argumento **kind**



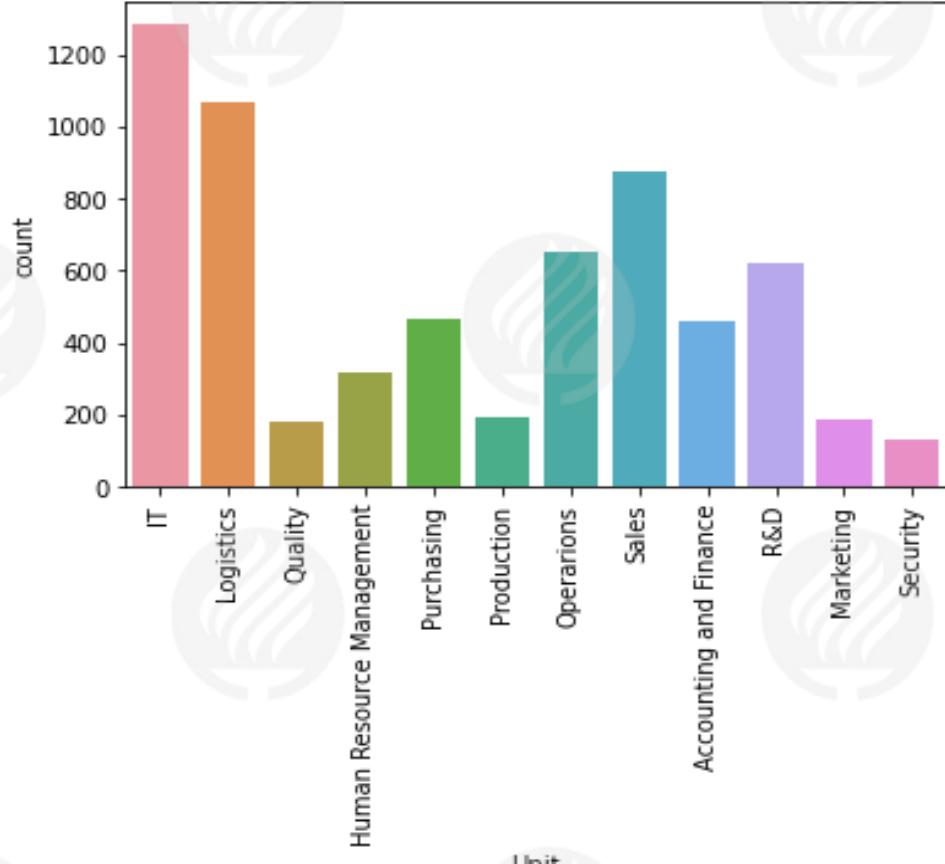
- Hay muchas plataformas que permiten trazar datos en el ecosistema de Python. **Matplotlib** fue la librería inicial de visualización y es considerada una de las más poderosas, pero ello incrementa su complejidad
- **Pandas** y **Seaborn** son interfaces de trazado superior o también conocidas como envolturas delgadas de Matplotlib
- ¿Habías utilizado alguna de estas **plataformas** anteriormente?

# Agrupamiento y filtrado | Tema 2.2



- Cuando se usan dataframes de gran volumen, o se combinan varios de ellos, es común que la visualización se lleve a cabo sobre el resultado de **agrupar** y/o **filtrar** registros
- Por ello, es muy común ocupar la función de **groupby** previo al trazado, con alguna operación sobre los datos (*count, sum, mean,...*)
- La salida agregada de groupby produce como **índices** las etiquetas de grupo, lo que impacta en la graficación

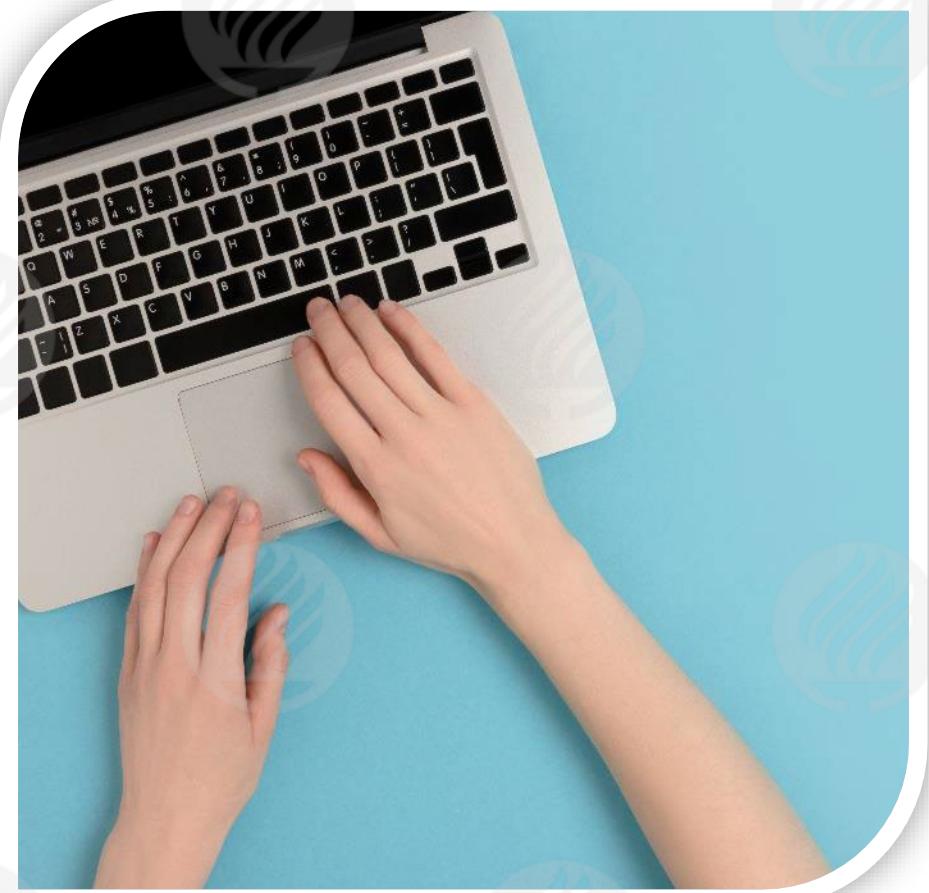
# Agrupamiento y filtrado | Tema 2.2



- **Seaborn** ofrece además un gráfico de recuento, con variables categóricas, que permite realizar el agrupamiento para *count* de manera automática
- Para ello, es necesario indicar en la función, la columna de la cual se obtendrán las etiquetas de grupo

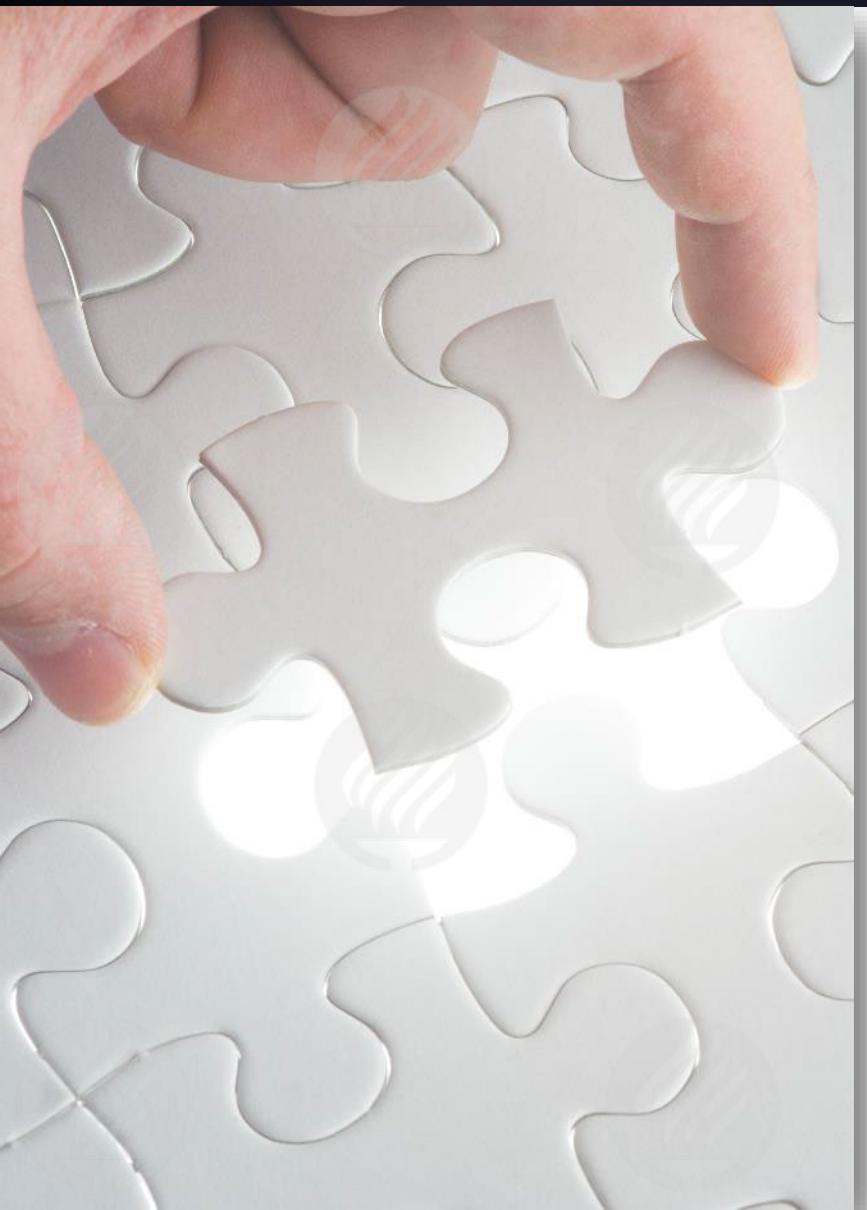
`sns.countplot(columna)`

# Actividad | Aprendizaje activo



Utiliza la misma libreta de **Colab** para:

- Generar un nuevo dataframe (**population\_by\_continent**) con el resultado de agrupar la columna **Population** con el total por continente
- Graficar los resultados usando un gráfico pie de **Pandas**
- Graficar el número de países por continente usando la función `countplot()` de **Seaborn**
- Cambiar la **paleta** de colores y el **tema** con Seaborn



- Cuando se manipulan dataframes con miles o millones de registros, los datos no son graficados de manera independiente, sino que se **combinan** y aplican operaciones sobre los mismos para obtener resultados en función de ciertas **variables o categorías**
- Ahora ya sabes cómo influye en la **visualización** el manejo de **datos agrupados** y/o **filtrados**
- ¿Puedes identificar las **ventajas** que ofrece cada una de las plataformas de trazado revisadas para este tema?

## Cierre: Concepto clave | Tema 2

Hay muchas opciones disponibles para la **visualización** de datos en Python. La mejor siempre estará en función de la **necesidad** que se intenta cubrir y la **estructura** subyacente de los datos

# Panorámica de la sesión

1

Sesión Síncrona 1  
Aprender

Tema 1: Exploración y validación de datos

Tema 2: Plataformas de visualización en Python

Tema 3: Gráficas superpuestas

Cierre de sesión

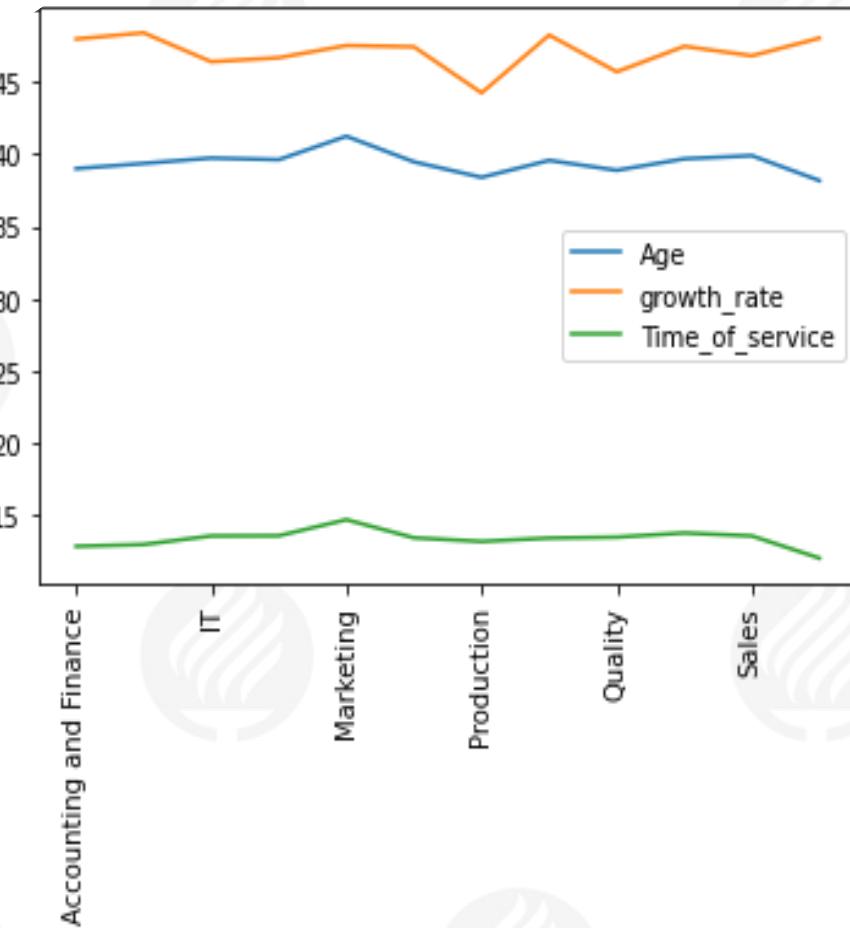
# Propósito | Tema 3



- La **visualización de series individuales** en gráficas independientes **no** será pertinente cuando se requieran:
  - ✓ Análisis comparativos
  - ✓ Mostrar relaciones entre dos o más variables
  - ✓ Contrastar con valores de referencia
- Generar gráficos con **ejes compartidos** enfatizará las **semejanzas o diferencias** entre los conjuntos de datos incluidos

# Compartiendo ambos ejes (horizontal y vertical)

## Tema 3.1



- **Con Matplotlib:** La función `plot()` grafica **todas** las series del `dataframe`, si éste se pasa como parámetro.

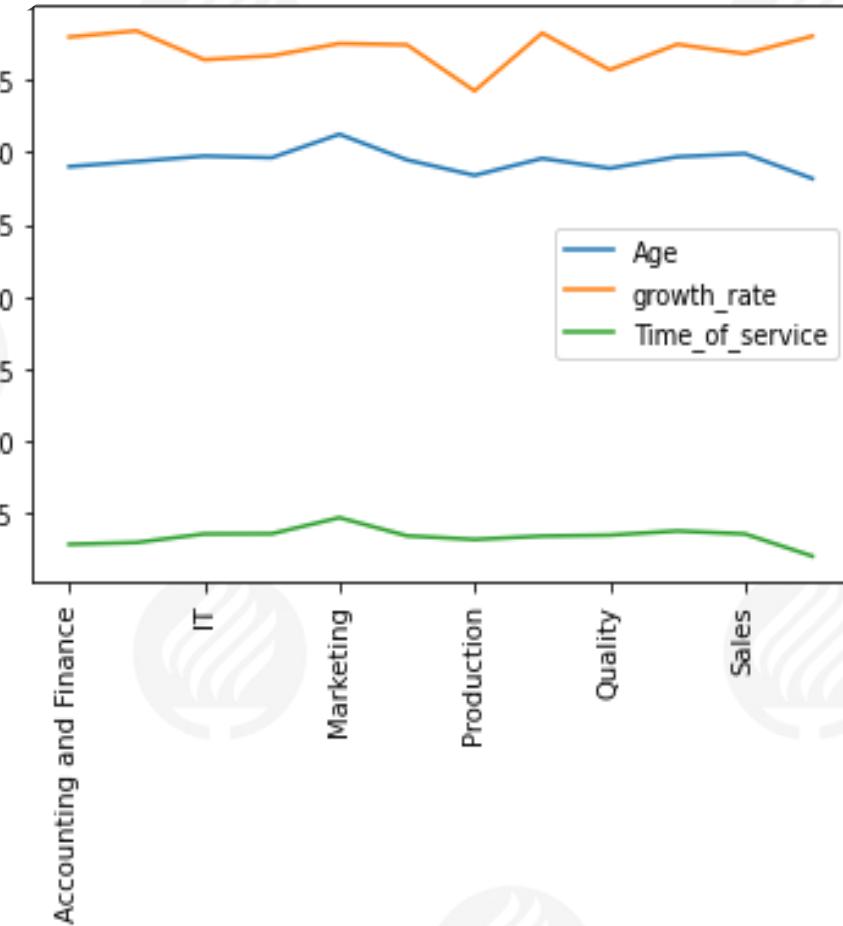
`plt.plot(df)`

Para mostrar en la leyenda los encabezados de cada serie, se utiliza el atributo **columns**

`plt.legend(df.columns)`

# Compartiendo ambos ejes (horizontal y vertical)

## Tema 3.1



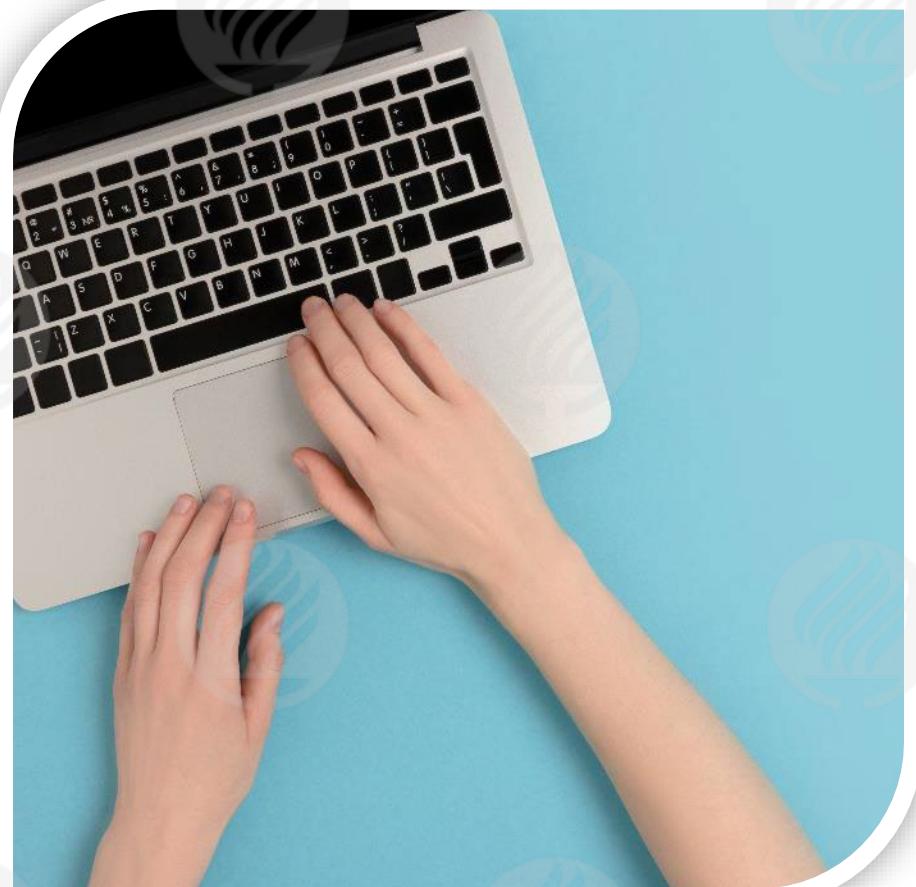
- **Con Pandas:** La función `plot()` graficará **todas** las series que contenga el `dataframe`, así que basta con una llamada. La leyenda se muestra por defecto en el gráfico

`df.plot()`

- **Con Seaborn:** La función `lineplot` hace el trazado de **todas** las series que contenga el `dataframe` que recibe en el **argumento data**

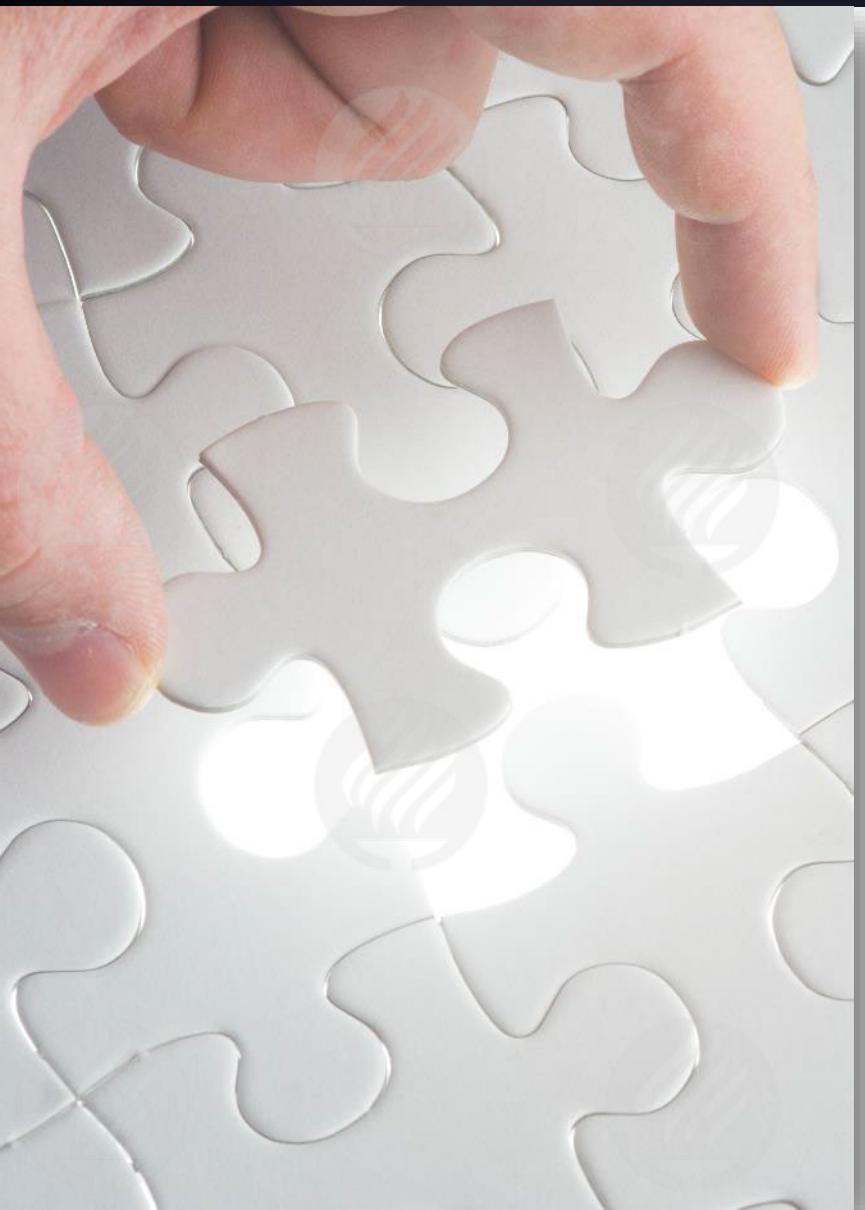
`sns.lineplot(data=df)`

# Actividad | Aprendizaje activo



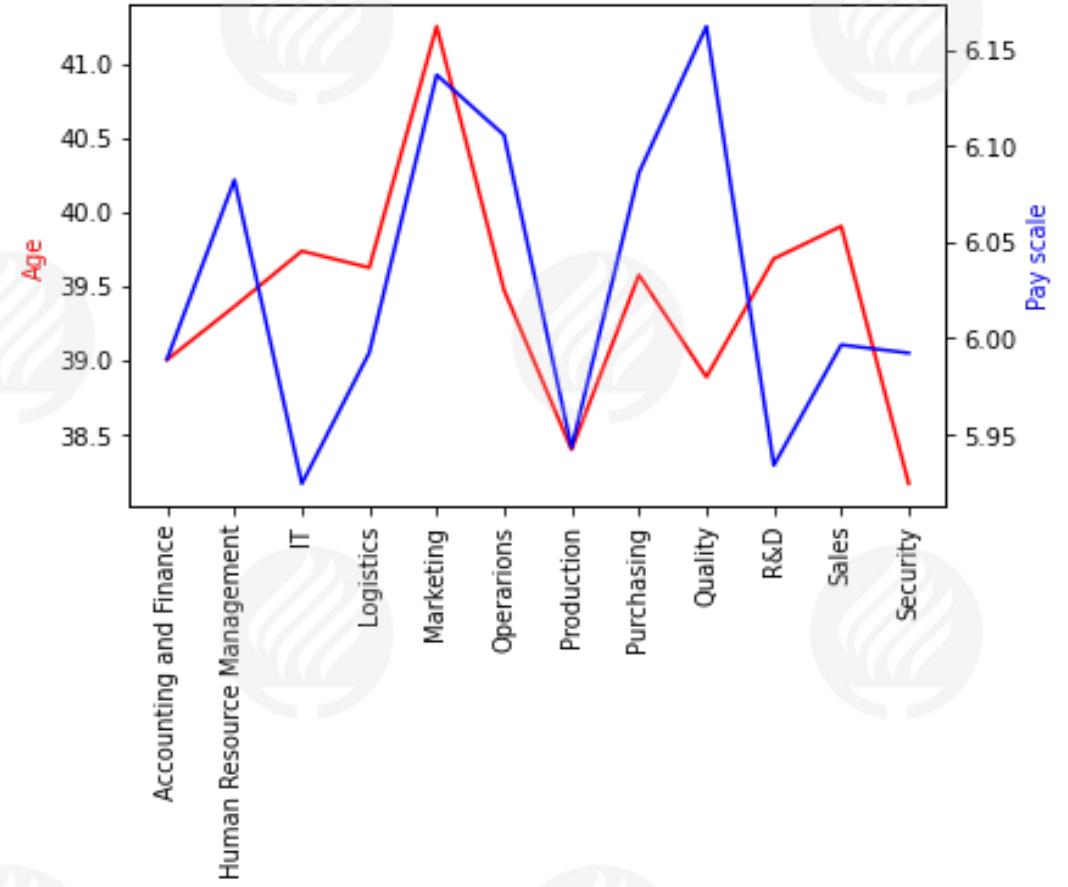
Utiliza la misma libreta de **Colab** para:

- Generar un nuevo dataframe (**gnp\_by\_continent**) con el resultado de agrupar las columnas **GNP** y **GNPOld** con el promedio por continente
- Graficar los resultados usando las **tres plataformas de trazado**



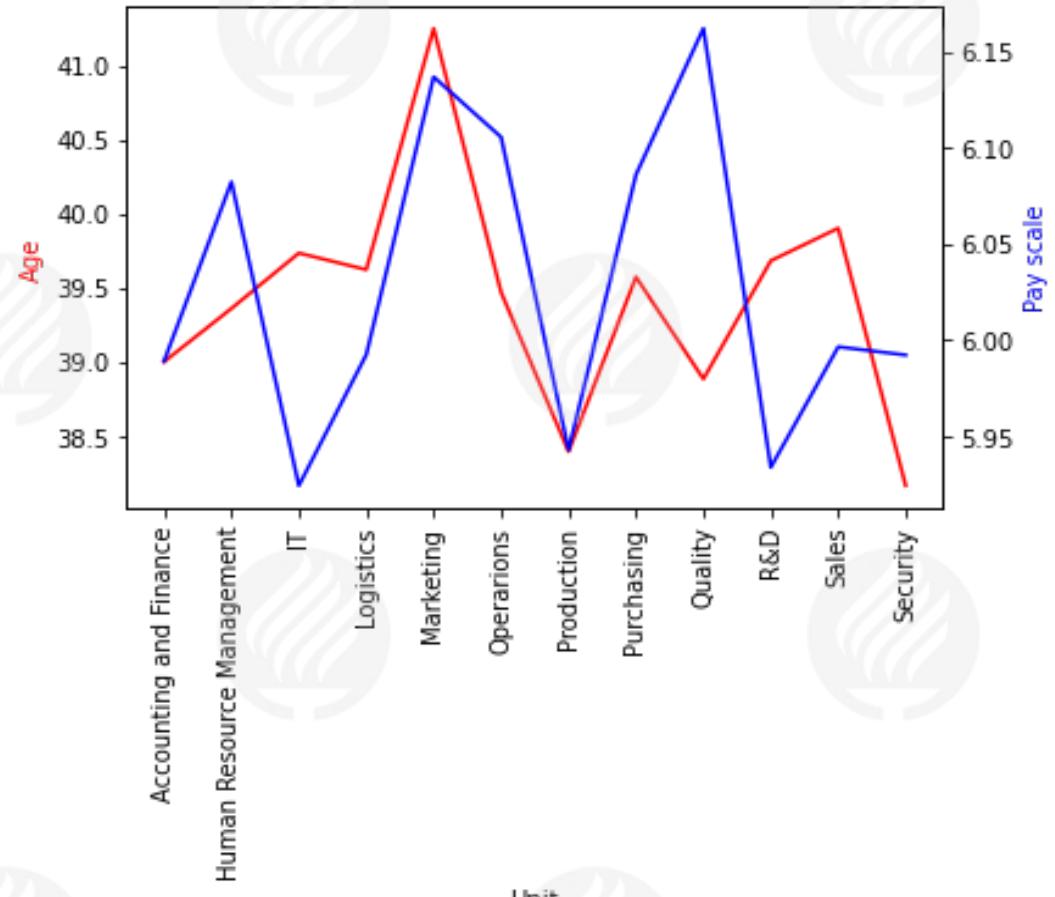
- Cuando dos series o más series de datos poseen la **misma escala**, es posible **comparar** sus magnitudes compartiendo ambos ejes y fijando los valores de uno de ellos para establecer la correlación
- Ahora que ya sabes cómo hacerlo, piensa en los reportes que generas como parte de tus actividades diarias y analiza: **¿en cuáles sería conveniente generar gráficas que compartan ambos ejes?**

# Compartiendo el eje horizontal con ejes verticales independientes | Tema 3.2



- Pero hay casos donde sólo es posible compartir un eje, pues en el otro las series poseen **diferentes escalas**
- Para ello se pueden usar ejes **duales** (*twin*) en una misma figura
- **`twinx()`** - crea un eje x invisible y un eje y independiente posicionado opuesto al original (es decir, a la derecha)

# Compartiendo el eje horizontal con ejes verticales independientes | Tema 3.2



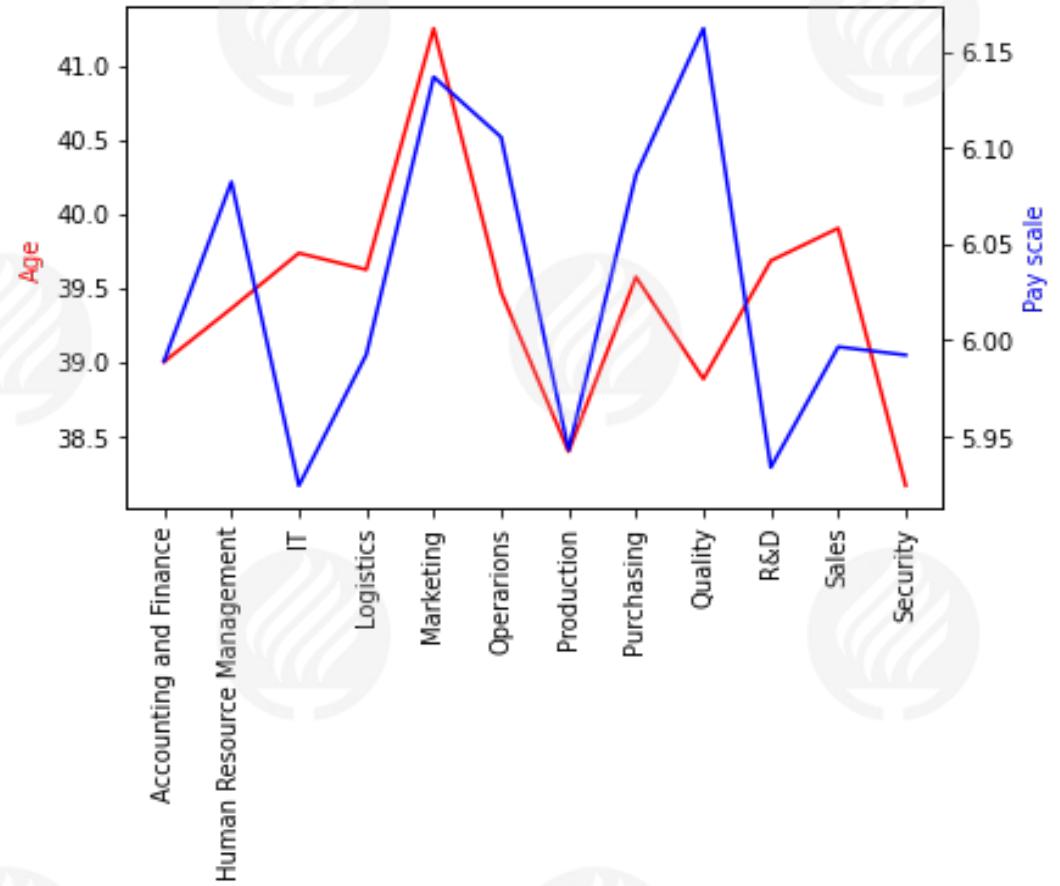
Esta configuración sólo es posible a través la función **subplot()** de matplotlib, que regresa:

- **fig** - El contenedor de nivel superior para todos los elementos de la figura
- **ax** - Un objeto (o arreglo) de gráficas

```
fig, ax1 = plt.subplots()  
ax1.plot(df.index, df.values)
```

```
ax2 = ax1.twinx()  
ax2.plot(df.index, df.values)
```

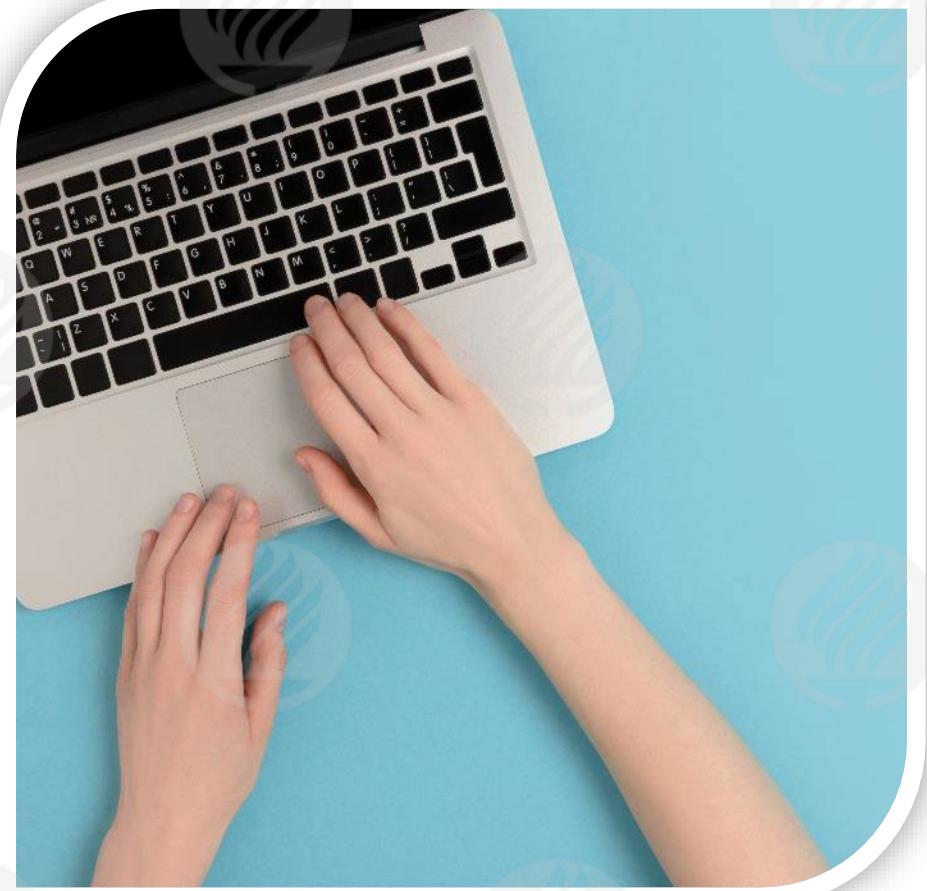
# Compartiendo el eje horizontal con ejes verticales independientes | Tema 3.2



Es recomendable en estos casos:

- Personalizar las etiquetas de cada eje  
`ax1.set_ylabel('Nombre eje y izquierdo') y  
ax2.set_ylabel('Nombre eje y derecho')`
- Dar un color a cada serie tanto para la gráfica en `plot()`, como para la etiqueta en `set_ylabel()` a través del argumento `color`
- Cuando se ejecutan las funciones `plot` a través de las envolturas de Pandas y Seaborn, en ellas hay que especificar la gráfica a través del argumento `ax`

# Actividad | Aprendizaje activo



Utiliza la misma libreta de **Colab** para:

- Generar un nuevo dataframe (**summary\_by\_continent**) con el resultado de agrupar las columnas **LifeExpectancy** y **GNP** con el promedio por continente
- Crear un eje vertical **dual** o secundario
- Graficar **LifeExpectancy** en el eje vertical izquierdo con color **azul**
- Graficar **GNP** en el eje vertical derecho con color **verde**



- Los **gráficos con ejes verticales independientes** son útiles para mostrar series que contienen rangos de valores muy distantes entre sí. De esta forma es posible emplear una **escala conveniente** para cada conjunto de datos
- También resultan de utilidad cuando hay datos de **diversa naturaleza** combinados en el mismo gráfico (por ejemplo, precio y capacidad de producción)

## Cierre: Concepto clave | Tema 3

**Los gráficos con varias métricas  
pueden ser más descriptivos  
visualmente compartiendo uno  
o ambos ejes, en dependencia  
de la variabilidad de la escala**

# Panorámica de la sesión

1

Sesión Síncrona 1  
Aprender

Tema 1: Exploración y validación de datos

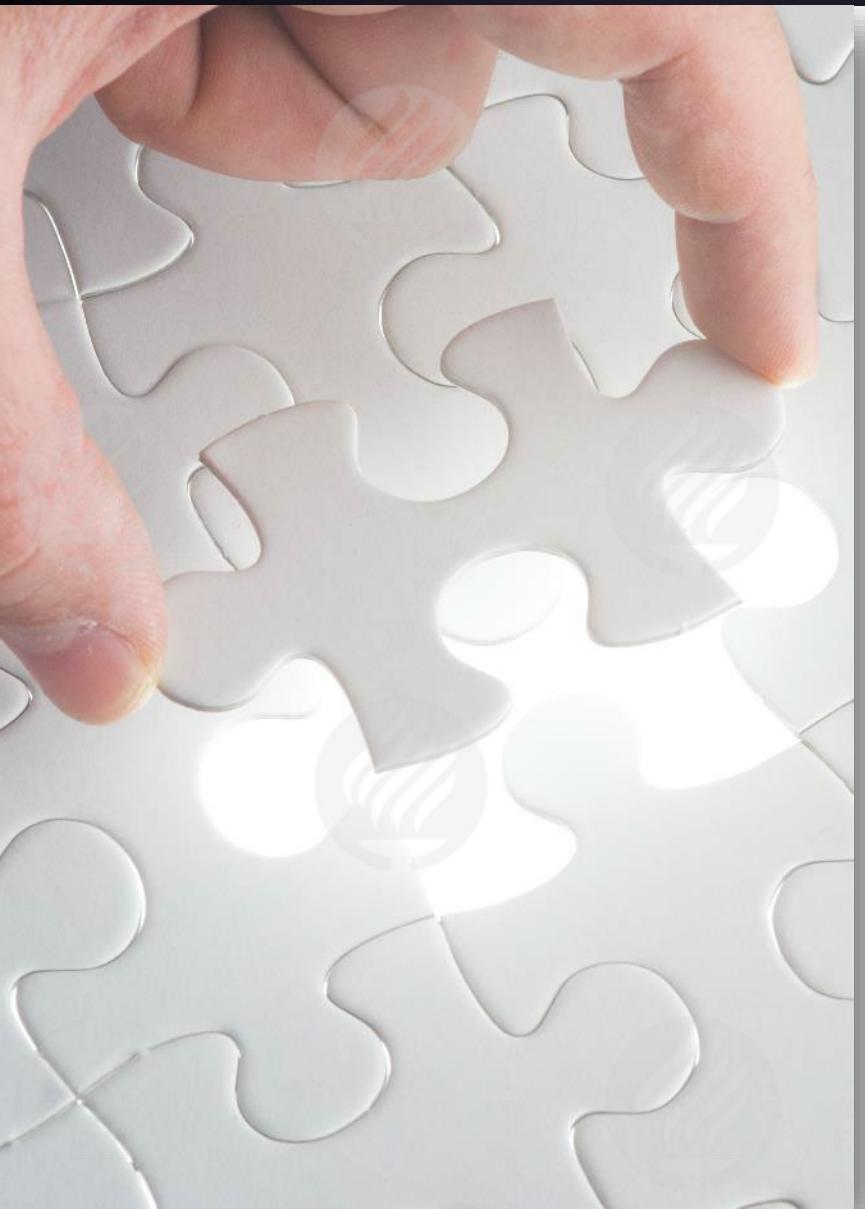
Tema 2: Plataformas de visualización en Python

Tema 3: Gráficas superpuestas

Cierre de sesión

## Cierre de la sesión

# Cierre | Sesión Sincrónica 1 [Aprender]

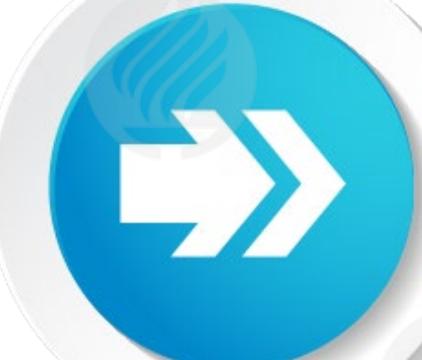


- ¿Eres capaz de analizar la **estructura** y **contenido** de las fuentes de información que utilizas como parte de la fase de exploración de datos?
- ¿Puedes resumir grandes cantidades de información usando **representaciones gráficas** para reconocer tendencias y vínculos entre sus variables?
- ¿En qué situaciones es oportuno el uso de **gráficas superpuestas** para lograr comparaciones concluyentes?

## Cierre: Concepto clave

**La exploración y visualización de los datos te ayuda a maximizar el rendimiento de la información como uno de los activos más importantes de la organización**

# Siguientes pasos | Semana de trabajo asíncrono [Profundizar]



1 Sesión Síncrona 1  
Aprender

2 Trabajo asíncrono 1  
Profundizar | Ruta de Aprendizaje

3 Sesión Síncrona 2  
Preparar para Aplicar

4 Trabajo asíncrono 2  
Aplicar en el trabajo | Reto

- La **compresión** de la distribución y organización de los datos y los conceptos elementales para su **visualización**, te permitirán asimilar los temas de la sesión Profundizar
- Ya estás familiarizado con las funciones básicas de tres plataformas de trazado: **Matplotlib**, **Pandas** y **Seaborn** y continuarás profundizando en ellas para crear **subgráficas**, hacer **anotaciones** de texto e incluir información en formato tabular y conocer las gráficas más empleadas para **exploración de datos**



Tecnológico de Monterrey | 2021

Prohibida la reproducción total o parcial de esta  
obra sin expresa autorización del Tecnológico  
de Monterrey

# Gracias | Programas LIVE