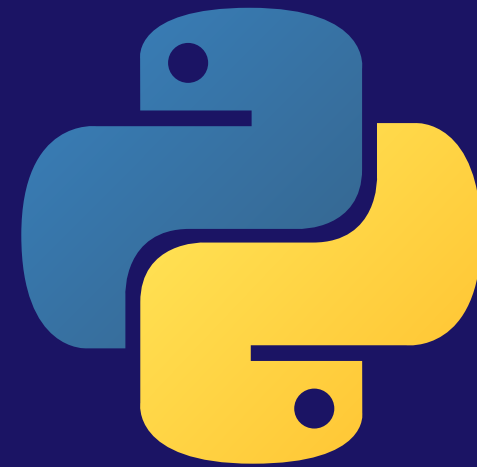
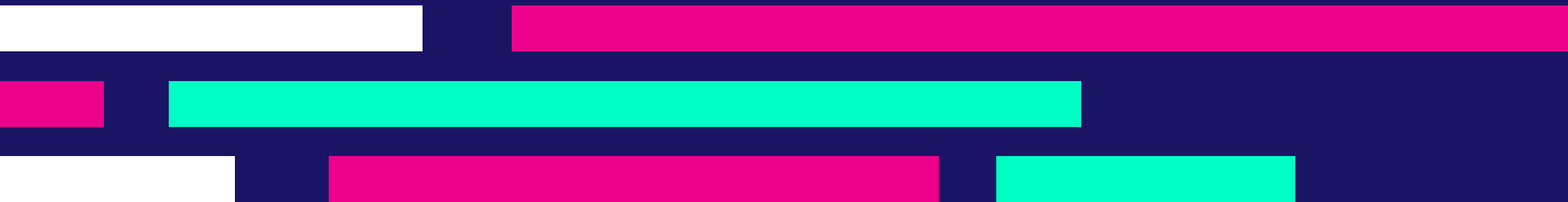




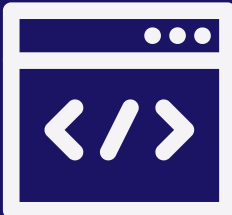
# ANÁLISIS DE DATOS CON PYTHON EN 7 DIAS



Profesor: Arturo Lorenzo Hernández



# RECURSOS DEL CURSO

- MATERIAL DE VÍDEO 
- DIAPOSITIVAS CON TEORIA 
- FICHEROS DE CÓDIGO CON EJEMPLOS Y EJERCICIOS 

# ORGANIZACIÓN DEL CURSO

- Preparación de entorno y herramientas necesarias
- Día 1: Fundamentos de Python
- Día 2: Conceptos Básicos de Análisis de Datos
- Día 3: Introducción Manipulación de Datos con Pandas
- Día 4: Análisis Exploratorio de Datos (EDA) - Parte 1
- Día 5: Análisis Exploratorio de Datos (EDA) - Parte 2
- Día 6: Visualización de Datos con Seaborn
- Día 7: Proyecto Final: Análisis de artistas en Spotify y Youtube

# Preparación de entorno y herramientas necesarias

- Instalación de Python
- Instalación de Pip
- Instalación de Visual Studio Code
- Instalación de Jupyter Notebook en Visual Studio Code

# INSTALACIÓN DE PYTHON

Hay muchas versiones de Python, nosotros vamos a usar la versión 3.9.9, ya que es una versión estable y sin errores.

Lo podemos descargar en este enlace:  
<https://www.python.org/downloads/windows/>

## PASOS EN LA INSTALACIÓN IMPORTANTES

1. Marcar la casilla de Add Python 3.9 to PATH

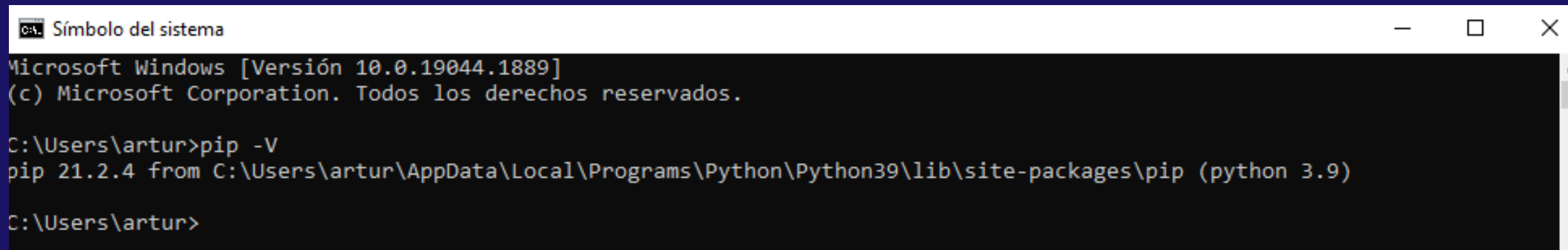
2. Instalar con la opción "Install Now"



# PIP, TU MEJOR AMIGO

Pip es el gestor de paquetes de código de Python, un paquete de código sirve para ayudar a los programadores a realizar tareas difíciles con menos código. Por ejemplo, si queremos leer el contenido de un fichero hay paquetes de código que ofrecen funcionalidades para hacerlo.

## COMPROBACIÓN DE QUE PIP ESTÁ INSTALADO



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19044.1889]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\artur>pip -V
pip 21.2.4 from C:\Users\artur\AppData\Local\Programs\Python\Python39\lib\site-packages\pip (python 3.9)

C:\Users\artur>
```

# ¿QUE ES UN EDITOR DE CÓDIGO?

Al igual que para hacer un documento de texto con imágenes y gráficos podemos utilizar Word de Microsoft o si queremos escribir cualquier nota dentro de nuestro ordenador tenemos la aplicación del bloc de notas, un programador debe escribir código en una aplicación específica para eso se inventaron los editores de código o IDE (Entorno de Desarrollo Integrado).

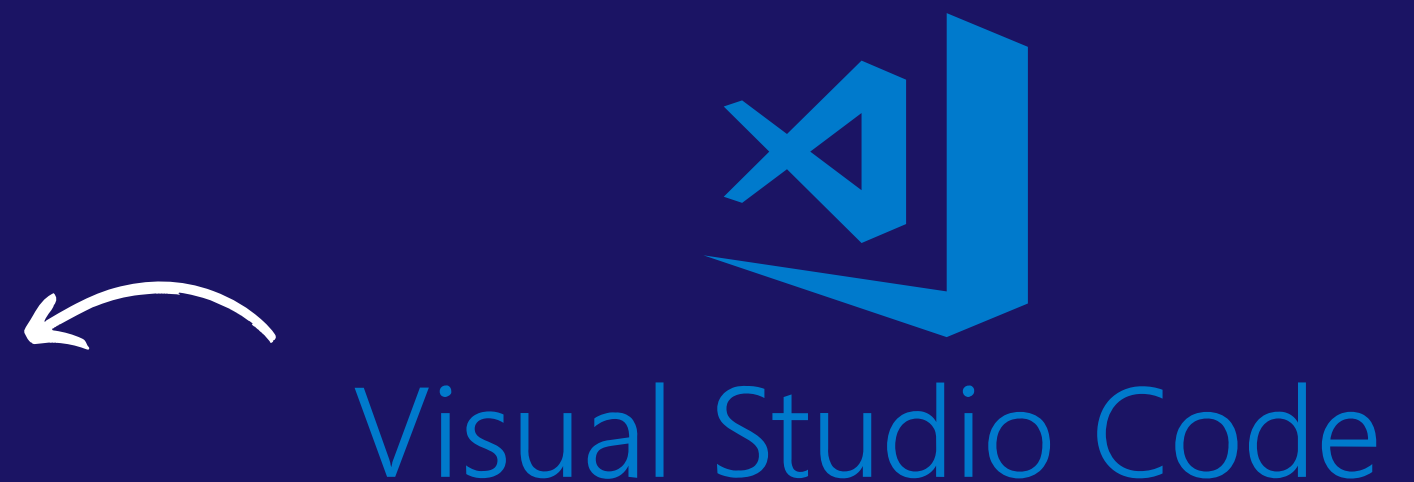
```
# 1. Longitud de un string
|
saludo = "Hola me llamo Arturo"

longitud_saludo = len(saludo)
print(longitud_saludo)

# 2. Quitar espacios en blanco de al principio y final de un string

string_espacios = "          Esto es un string con espacios feos"
print(string_espacios)

string_sin_espacio = string_espacios.strip()
print(string_sin_espacio)
```



# INSTALACIÓN DE VISUAL STUDIO CODE

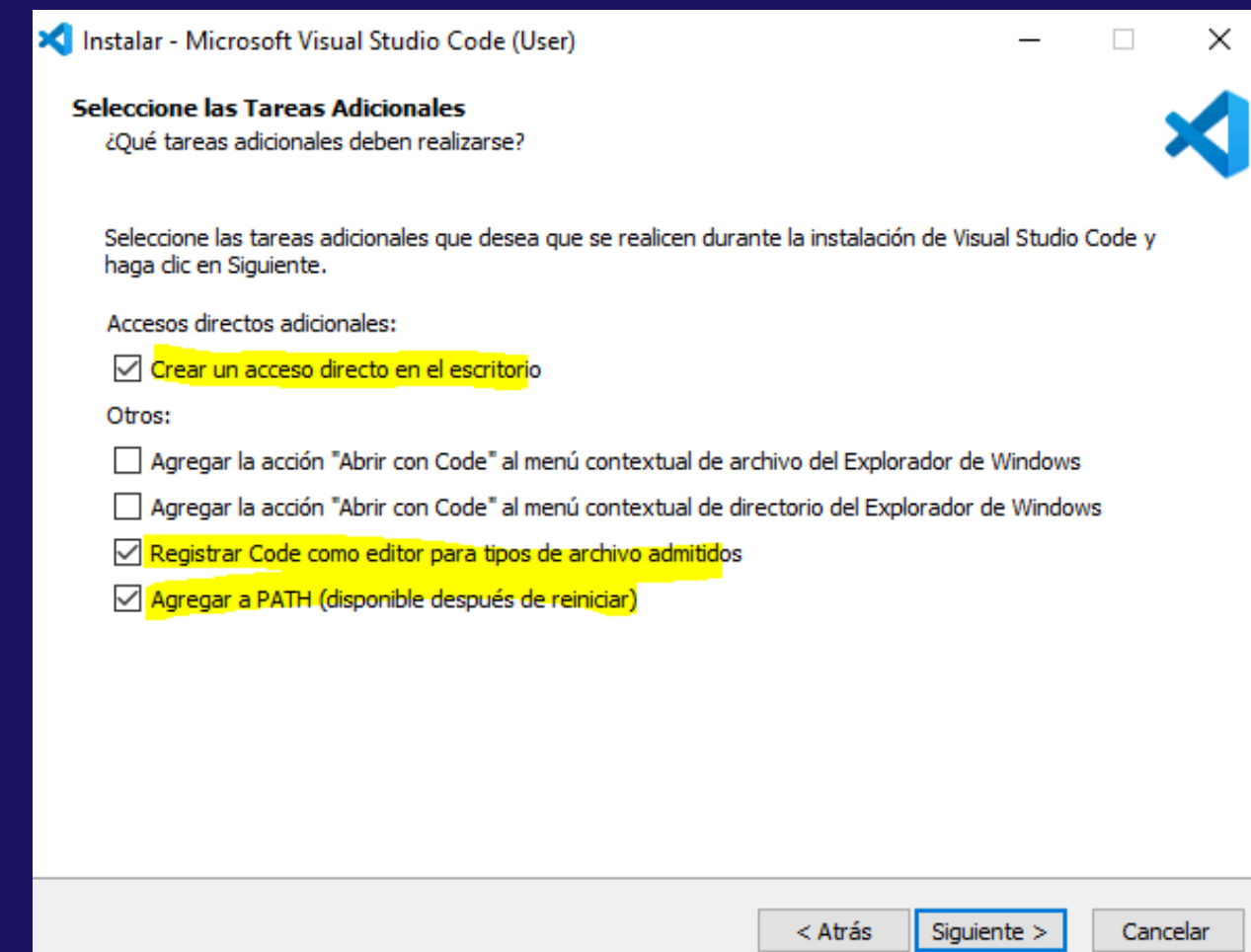
Existen muchos editores de código para programar, vamos a utilizar visual studio code ya que es de los más intuitivos y fáciles de usar.

En el siguiente enlace puedes descargar el programa:

<https://code.visualstudio.com/download>

## PASOS EN LA INSTALACIÓN IMPORTANTES

1. Marcar la casilla de Agregar a PATH
2. Registrar Code como editor para tipo de archivos admitidos





# ¿QUÉ SON LOS MARKDOWNS?

Markdown es un lenguaje que se utiliza para dar formato a texto de manera sencilla utilizando una sintaxis simple y legible. Con Markdown, se puede agregar encabezados, listas, enlaces, imágenes y otros elementos de formato básico al texto.

Las celdas Markdown en un notebook se utilizan para proporcionar explicaciones, documentación, comentarios o cualquier otro contenido textual que complemente el código y los resultados presentados en las celdas de código.

Esto son apuntes de la sintaxis de uso para los casos más típicos:  
<https://www.markdownguide.org/cheat-sheet/>

```
# HOLA ESTA ES UN H1
## HOLA ESTO ES UN H2

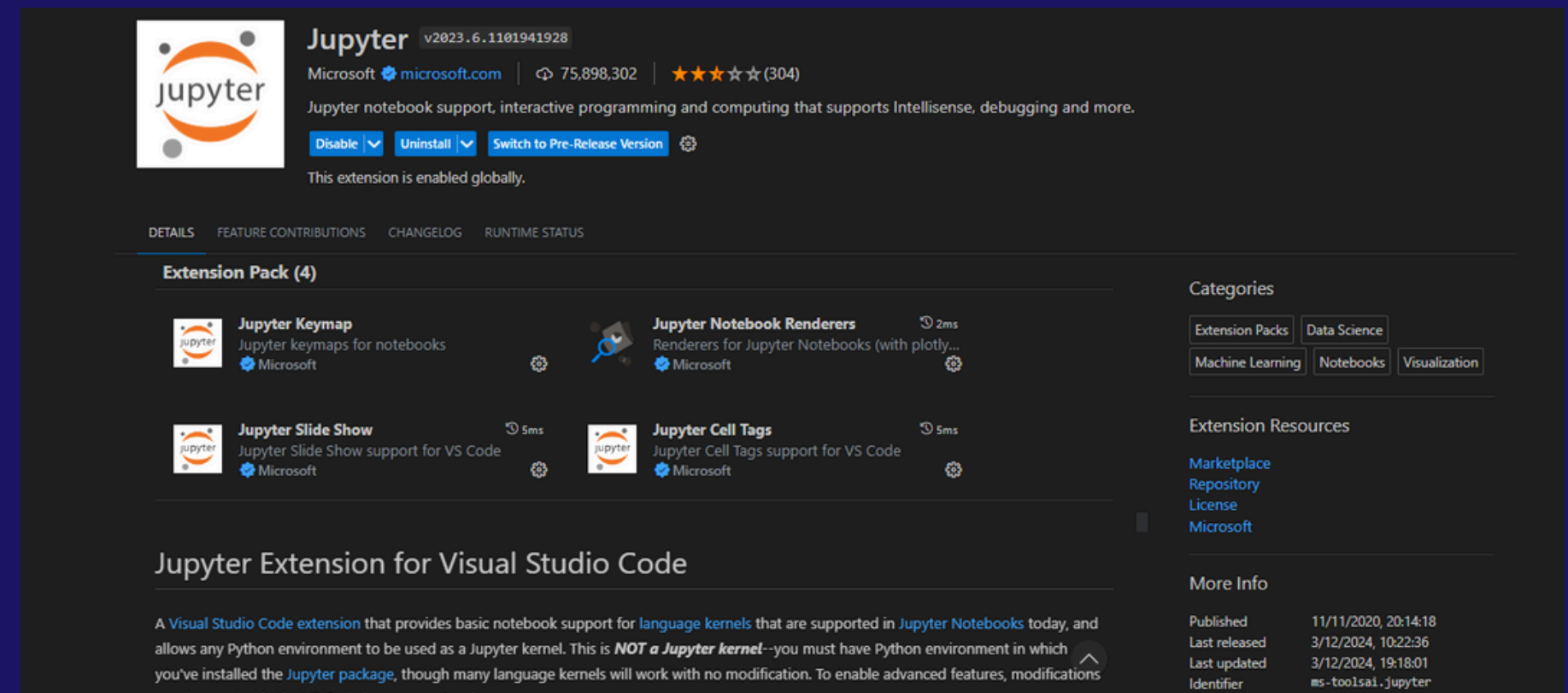
**TEXTO EN NEGRITA**
```

# INSTALACIÓN DE JUPYTER NOTEBOOKS

Jupyter Notebooks es una interfaz para desarrollar dentro de Visual Studio Code orientada al análisis de datos. La podemos descargar como extensión de Visual Studio Code.

## PASOS EN LA INSTALACIÓN IMPORTANTES

1. Instalar módulo jupyter para ejecutar celdas de código
2. Instalar módulo nbconvert para exportar notebooks a PDF o HTML



# DIA 1: Fundamentos de python

- Introducción a Python: Características del lenguaje y usos
- Tipos de datos básicos y operadores
- Estructuras de datos
- Estructuras de control: Condicionales
- Estructuras de control: Bucles
- Funciones
- Programación orientada a objetos

# Introducción a Python: Características del lenguaje.

## Principales características de python:

1. Lenguaje de alto nivel
2. Interpretado
3. No es necesario establecer el tipo de datos de las variables (si es recomendable)
4. Fácil de leer y escribir
5. Multiplataforma, se puede ejecutar en muchos sistemas operativos como Windows, Linux y Mac

# Introducción a Python: Usos del lenguaje.

Python es un lenguaje de programación muy versátil y fácil de aprender, con una comunidad de desarrolladores muy activa, esto ha hecho que se desarrollen librerías de código para diversos ámbitos de la informática, donde más se usa es en:

## 1. Ciencias de datos y análisis



## 2. Machine learning



## 3. Desarrollo web



# Tipos de datos básicos y operadores

Cada variable en programación tiene un tipo de dato y con cada tipo de datos se pueden realizar ciertas operaciones, estos son los más utilizados en Python:

- Números (enteros, flotantes, complejos), cadenas de texto (str), booleanos (True, False), listas, tuplas, conjuntos y diccionarios.
- Operadores Principales:
  - Aritméticos: +, -, \*, /
  - Comparación: ==, !=, <, >
  - Lógicos: and, or, not
  - Asignación: =, +=, -=

Estos elementos son esenciales para cálculos, comparaciones y manipulación de datos.

# Estructuras de datos

Las estructuras de datos nos permiten almacenar la información de una forma más eficiente y ordenada, además permiten un acceso sencillo, en Python existen las siguientes estructuras de datos:

- Listas: Colecciones ordenadas y mutables que permiten elementos duplicados. Ejemplo: [1, 2, 3].
- Tuplas: Colecciones ordenadas e inmutables. Ejemplo: (1, 2, 3).
- Conjuntos: Colecciones desordenadas y sin duplicados. Ejemplo: {1, 2, 3}.
- Diccionarios: Colecciones de pares clave-valor, ordenadas desde Python 3.7. Ejemplo: {"a": 1, "b": 2}.

# Estructuras de control: Condicionales

Un condicional en programación es muy parecido a la vida real, todos los días tomamos decisiones en base a condiciones, cogemos el coche para ir al trabajo si está lloviendo, tomamos café si tenemos sueño o estamos cansados, bebemos agua si tenemos sed, etc...

Los condicionales son estructuras que permiten ejecutar diferentes bloques de código según si una condición es verdadera o falsa.

```
if ahorros >= precio_entrada and dia_libre:  
    print("VOY AL CONCIERTO!")  
else:  
    print("ME QUEDO EN CASA :(")
```



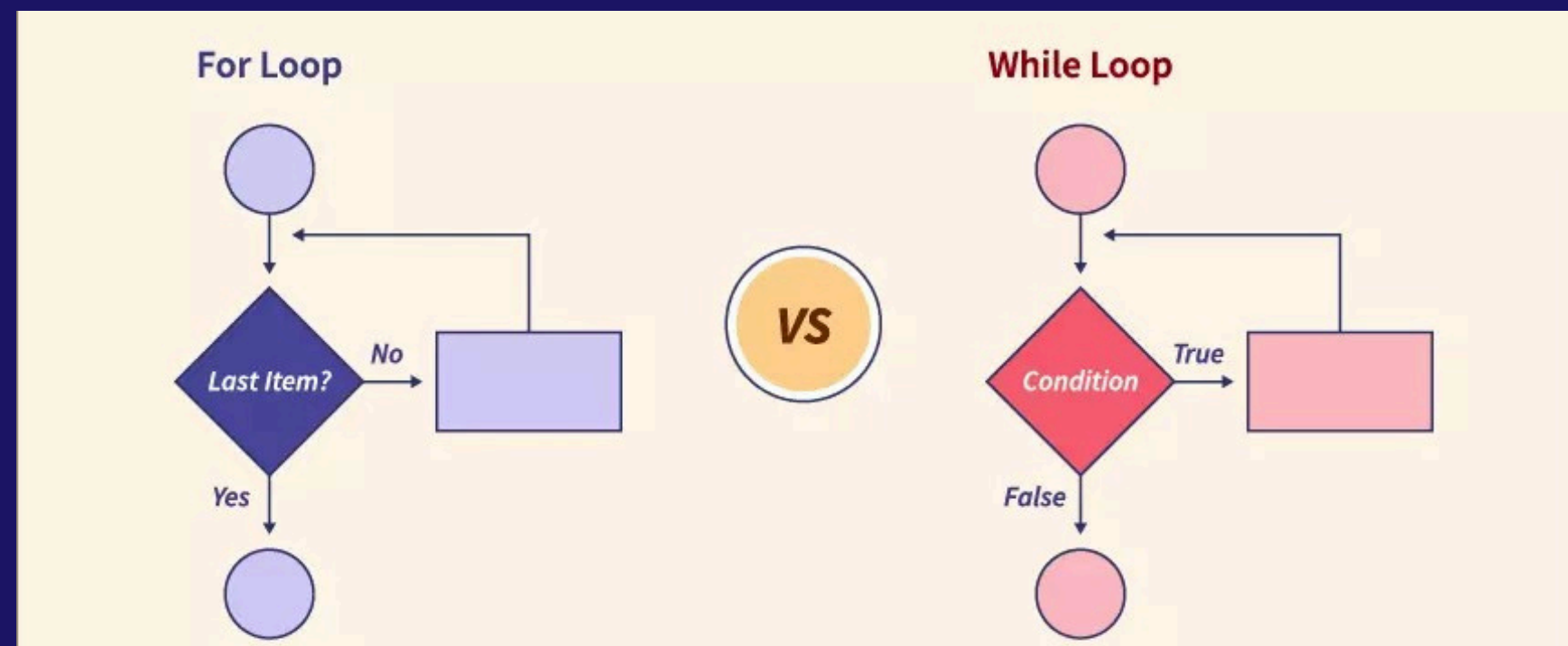
# Estructuras de control: Bucles

Los bucles en programación nos permiten ejecutar una acción de forma repetida, en Python tenemos dos tipos:

- Bucle While: se repite una acción hasta que no se cumpla cierta condición.
- Bucle for: sirve para recorrer cada uno de los elementos de una estructura de datos como puede ser una lista.

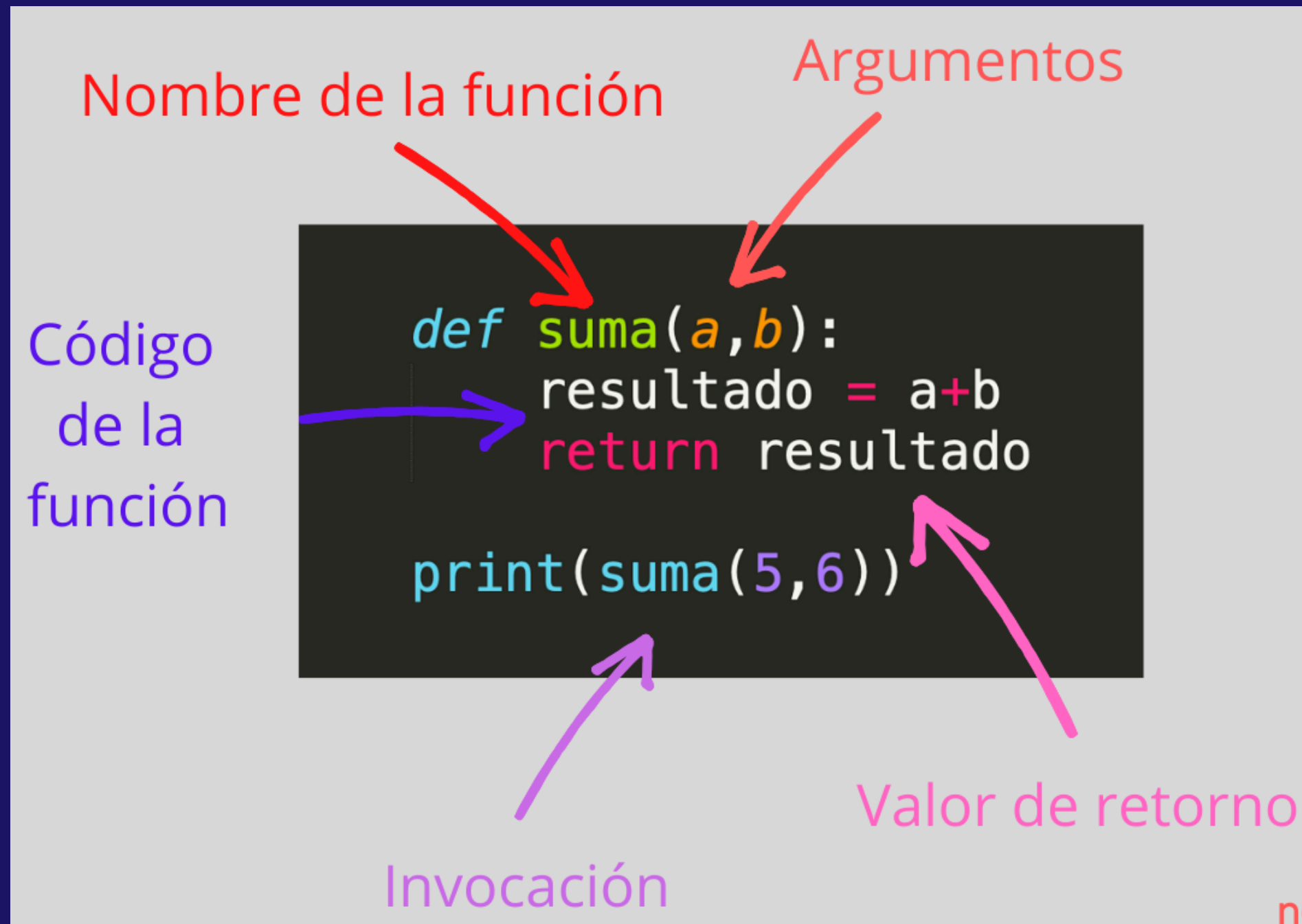
Un ejemplo de bucle while puede ser pedirle al usuario su nombre y hasta que el usuario no lo introduzca de forma correcta volver a pedirle que lo introduzca.

Un ejemplo de bucle for podría ser recorrer una lista de strings para encontrar el string con mayor número de letras.



# Funciones

Una función es un conjunto de instrucciones que se agrupan bajo un nombre y se pueden llamar en cualquier momento desde cualquier parte del programa.



# Programación orientada a objetos

La programación orientada a objetos es el siguiente paso a la construcción de módulos de funciones. Lo que vamos a encontrar son clases y objetos, una clase es parecida a un módulo de funciones pero con un nombre específico, unos atributos que definen las características de esa clase y unos métodos que son las funciones que definiríamos en un módulo. Un objeto es una instancia de una clase con unos atributos concretos.

## Atributos

- Marca
- Color
- Año

## Métodos

- Acelerar()
- Frenar()
- Encender()
- Apagar()



# DIA 2: Conceptos básicos del análisis de datos

- Introducción al análisis de datos: ¿Por dónde empezamos?
- Introducción a Pandas: ¿Qué es pandas?
- Estructuras de datos: Dataframe y Series
- Tipos de ficheros de datos más relevantes en análisis de datos

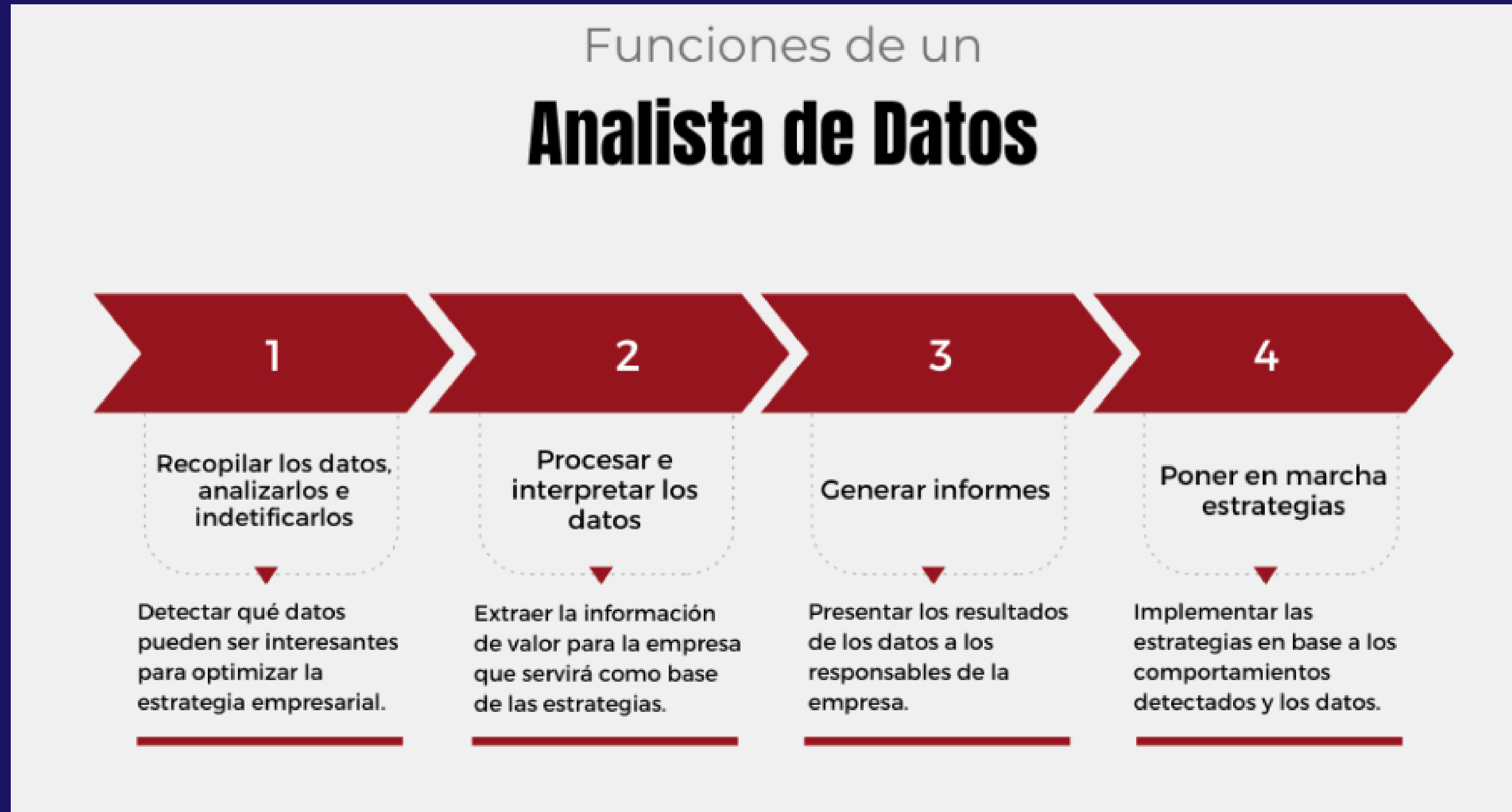
# Introducción al análisis de datos: ¿Por dónde empezamos?

El trabajo de un analista de datos se puede resumir en una persona que se dedica a resolver preguntas enfocadas en mejorar los procesos de un negocio.

Su trabajo implica recopilar, limpiar, analizar y visualizar datos con el objetivo de identificar tendencias, patrones y relaciones que puedan ser relevantes para una empresa u organización.

Cuando encuentras algo valioso, lo presentas de manera atractiva, ya sea en un informe detallado o con gráficos coloridos. Es como contar una historia con los datos, ayudando a las empresas a entender lo que está sucediendo y tomar decisiones inteligentes para el futuro.

# Introducción al análisis de datos: ¿Por dónde empezamos?





# Introducción a Pandas: ¿Qué es Pandas?

Pandas es una librería de Python que proporciona estructuras de datos y funciones de análisis de datos eficientes y flexibles. Es ampliamente utilizada en el campo del análisis de datos y la ciencia de datos debido a su capacidad para trabajar con datos en formato de tabla y etiquetados de manera intuitiva.

En el día a día de un analista de datos puede llegar a trabajar con muchos datos diferentes y con distintos formatos, columnas, tamaños, etc...

Por eso es necesario tener una librería que nos abstraiga para poder realizar nuestro trabajo con relativa sencillez.



# Estructura de datos: Dataframes y Series

Los dataframes y las series son las dos estructuras de datos más importantes en Pandas.

Un DataFrame es como una tabla de Excel: tiene filas y columnas donde puedes almacenar datos organizados, además tiene una columna de índice que indica el valor que hay en cada fila.

En cambio, una Series es como una sola columna de esa tabla, equivalente a una lista de valores con una etiqueta o índice, como una lista de frutas de una tienda.

Mientras el DataFrame es ideal para manejar conjuntos de datos complejos, la Series es perfecta para trabajar con un solo tipo de dato ordenado.

```
Serie de frutas:
```

```
0    Manzana
```

```
1    Plátano
```

```
2      Fresa
```

```
3    Naranja
```

```
4      Piña
```

```
dtype: object
```

```
DataFrame:
```

```
Nombre  Edad  Ciudad
```

```
0    Juan   25   Madrid
```

```
1  María   30  Barcelona
```

```
2    Luis   35   Sevilla
```

```
3     Ana   28   Valencia
```



# Tipos de ficheros de datos más relevantes en análisis de datos

Los datos que analizamos se pueden extraer de bases de datos o ficheros, existen diversos tipos de ficheros para almacenar datos de forma eficiente:

- CSV: Archivo de texto plano donde los datos están separados por comas, ideal para datos simples.
- JSON: Formato estructurado basado en texto que organiza datos en pares clave-valor, útil para datos jerárquicos o anidados.
- XLSX: Formato de hojas de cálculo de Excel que permite almacenar datos tabulares con fórmulas, gráficos y formato.
- Parquet: Formato binario optimizado para grandes volúmenes de datos, diseñado para almacenamiento eficiente y consultas rápidas.



# Día 3: Introducción Manipulación de Datos con Pandas

- ¿Qué es un índice en un dataframe?
- ¿Cómo manipulamos los índices de un dataframe?
- Selección condicional de datos
- Asignación de datos

# ¿Qué es un índice en un dataframe?

Un índice en un DataFrame de Pandas es como las etiquetas en un cuaderno: identifica y organiza cada fila para facilitar el acceso a los datos.

Puede ser numérico (como un conteo del 0 en adelante) o personalizado (por ejemplo, nombres o fechas). Sirve para realizar búsquedas rápidas, ordenar información o alinear datos entre diferentes estructuras.

The diagram illustrates a DataFrame with four rows and five columns. The columns are labeled 'Name', 'Age', 'Salary', and 'Department'. The rows are indexed from 0 to 3. Red arrows and labels highlight the indexing structure: 'Rows indexes' points to the index column, 'Column Names' points to the column headers, 'Rows' points to the row indices, and 'Columns' points to the column headers.

	Name	Age	Salary	Department
0	Tom	20	50000	HR
1	Nick	21	60000	Engineering
2	John	19	55000	Marketing
3	Tom	20	70000	HR

# ¿Cómo manipulamos los índices de un dataframe?

Manipular índices en Pandas es muy flexible y permite personalizar la forma en que accedemos a los datos. Puedes cambiar el índice con el método set\_index(), restablecerlo a su forma original con reset\_index() o incluso renombrarlo con rename(). También es posible ordenarlos con sort\_index().

A diferencia de una columna normal, el índice no se considera parte explícita de los datos y se usa principalmente para acceder, filtrar o alinear información

Para muchas operaciones tenemos que tener bien definidos los índices y haberlos tratado previamente.

# Selección condicional de datos

La selección condicional de datos en Pandas permite filtrar filas o columnas según ciertas condiciones, similar a buscar productos en una tienda según su precio o categoría. Usando expresiones lógicas (por ejemplo, `df[df['Edad'] > 18]`), puedes obtener solo las filas que cumplan con el criterio, como personas mayores de 18 años. También puedes combinar condiciones con operadores como `&` (y), `|` (o) y `~` (no), para crear filtros más complejos.

	Year	Gender	Rank	Name	Count
0	1982	Female	5	MICHELLE	2446
1	2000	Female	24	DESTINY	1188
2	1992	Female	6	AMANDA	2776
3	1991	Female	23	JASMINE	1482
4	1967	Male	19	THOMAS	2149
5	2013	Female	1	SOPHIA	3447
6	1980	Male	14	BRIAN	2737
7	1983	Female	16	VANESSA	1517
8	1995	Male	16	CHRISTIAN	2761
9	1960	Male	21	PAUL	2136



`df [ df[ 'Rank' ] > 19 ]`

`df [ df[ 'Rank' ].gt(19)]`

	Year	Gender	Rank	Name	Count
1	2000	Female	24	DESTINY	1188
3	1991	Female	23	JASMINE	1482
9	1960	Male	21	PAUL	2136

# Asignación de datos

La asignación de datos en un DataFrame de Pandas consiste en modificar o agregar valores a las celdas, columnas o filas, como actualizar el precio de un producto en un inventario.

Puedes asignar valores a una columna existente usando el nombre de la columna (por ejemplo, `df['Precio'] = 10`) o crear nuevas columnas basadas en cálculos (como `df['Total'] = df['Precio'] * df['Cantidad']`).

También puedes usar condiciones para cambiar valores específicos, por ejemplo, `df.loc[df['Stock'] == 0, 'Estado'] = 'Agotado'`.

Esto es especialmente útil para realizar transformaciones intermedias en los dataframes que nos permitan sacar insights interesantes.



# Día 4: Análisis Exploratorio de Datos (EDA) - Parte 1

- ¿Qué estadísticas nos interesan de un dataframe?
- Manejo de función de agrupación (group by)
- Manejo de multi-índices en un dataframe
- Manejo de función de ordenación (sort)

# ¿Qué estadísticas nos interesan de un dataframe?

Hay varias métricas estadísticas que es importante conocer su definición:

El promedio representa el valor central calculado como la suma de todos los datos dividida por su cantidad, útil para medir tendencias generales.

La mediana, por otro lado, es el valor que se encuentra justo en el medio cuando los datos están ordenados, lo que la hace más robusta frente a valores extremos.

La desviación estándar evalúa qué tan dispersos o concentrados están los datos alrededor del promedio, reflejando su variabilidad.

Finalmente, un percentil indica el punto por debajo del cual cae un porcentaje específico de los datos, como usar el percentil 90 para identificar los valores más altos en un conjunto.



# ¿Qué estadísticas nos interesan de un dataframe?

Cálculo de la desviación estándar. Se calcula en los siguientes pasos:

1. Encuentra el promedio de los datos.
2. Resta el promedio a cada dato y eleva esas diferencias al cuadrado.
3. Calcula el promedio de esas diferencias al cuadrado (esto se llama varianza).
4. Saca la raíz cuadrada de la varianza.

Por ejemplo, para los datos [2, 4, 6]:

- $\text{Promedio} = (2 + 4 + 6) / 3 = 4$
- $\text{Diferencias al cuadrado} = [(2-4)^2, (4-4)^2, (6-4)^2] = [4, 0, 4]$
- $\text{Varianza} = (4 + 0 + 4) / 3 = 2.67$
- $\text{Desviación estándar} = \sqrt{2.67} \approx 1.63$

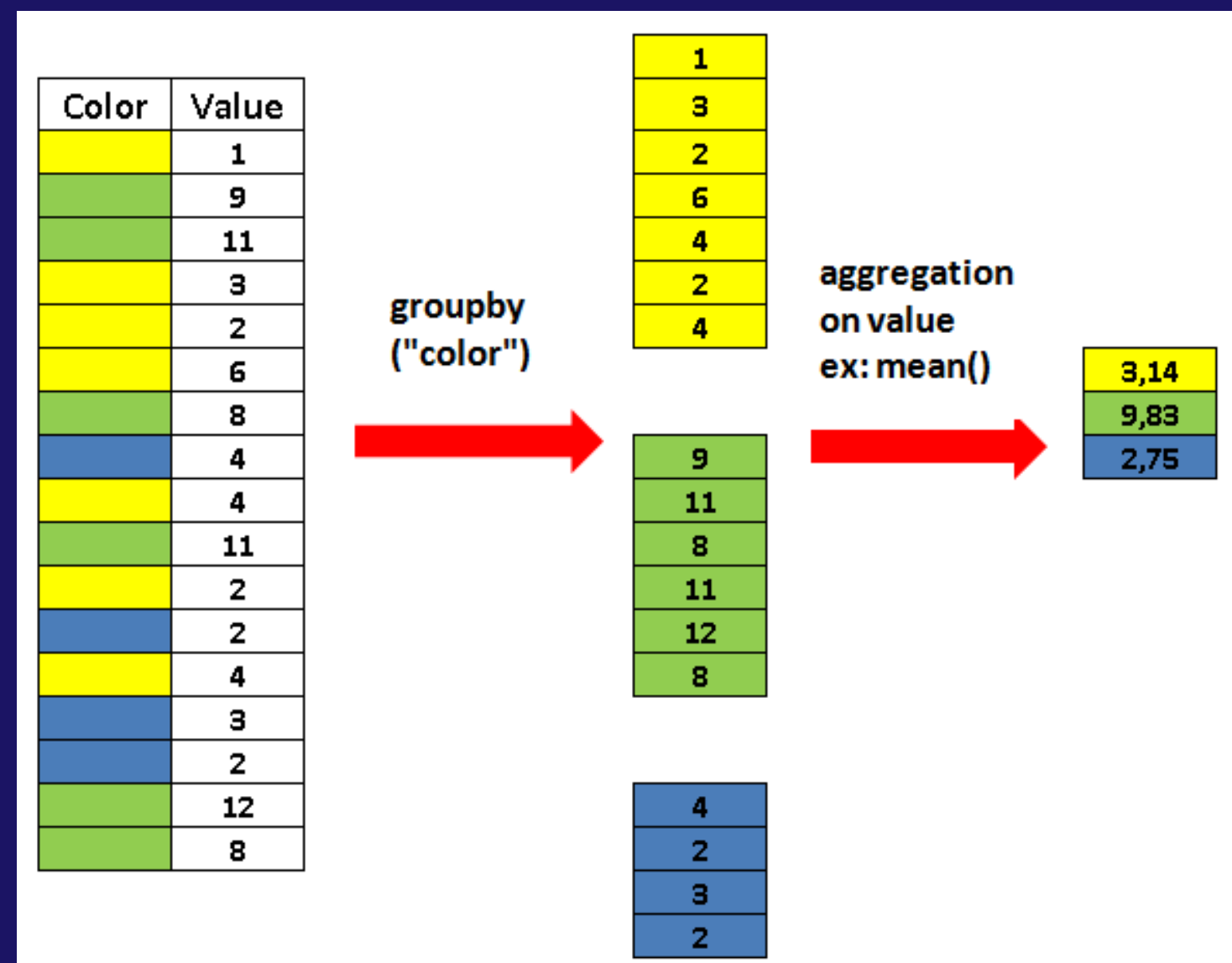
# ¿Qué estadísticas nos interesan de un dataframe?

Cálculo de percentiles. Se realiza en los siguientes pasos:

1. Ordena los datos de menor a mayor.
2. Calcula la posición del percentil usando la fórmula:
3. 
$$P = (p / 100) \times (n - 1) + 1$$
4. donde p es el percentil deseado y n el número de datos.
5. Si la posición no es un entero, interpola entre los datos más cercanos.  
Por ejemplo, para el percentil 75 de los datos [1, 3, 4, 7, 9]:
  - $$P = (75 / 100) \times (5 - 1) + 1 = 4$$
  - El percentil 75 es el 4º valor ordenado: 7.

# Manejo de función de agrupación (group by)

La función `groupby` en Pandas se utiliza para agrupar datos según una o más columnas y aplicar funciones agregadas como sumas, promedios o conteos sobre cada grupo. Es como organizar estudiantes por curso y calcular el promedio de notas por clase. Este enfoque permite analizar y resumir grandes conjuntos de datos de forma más ordenada, facilitando la comparación entre grupos.



# Manejo de multi-índices en un dataframe

Los multi-índices en Pandas permiten organizar un DataFrame en varios niveles de índices, como si estructuraras datos en jerarquías.


Por ejemplo, podrías clasificar primero por país y luego por ciudad. Esto es útil para trabajar con datos complejos, como ventas agrupadas por región y producto, ya que facilita acceder, filtrar y realizar operaciones sobre subconjuntos específicos de datos en cada nivel de la jerarquía.

		MultiIndex Columns		
		Temperature	Wind	
		°C	mph	
MultiIndex Rows	Oxford	2022-03-01	15	8
		2022-03-02	16	9
	London	2022-03-01	18	7
		2022-03-02	17.5	6

# Manejo de función de ordenación (sort)

La función sort en Pandas permite ordenar un DataFrame por los valores de una o más columnas, o incluso por el índice. Con `sort_values()`, puedes organizar los datos de manera ascendente o descendente según una columna específica, como ordenar productos por precio o fecha.

Con `sort_index()`, puedes reorganizar las filas o columnas según el índice, útil para estructurar datos en un orden lógico o cronológico.



## Sort by Column in Pandas DataFrame

Diagram illustrating sorting a DataFrame by the 'Age' column.

**Original DataFrame:**

	Name	Age	City
0	John	45	New York
1	Emma	30	London
2	Michael	35	Paris

↓

**Sorted DataFrame (by Age):**

	Name	Age	City
1	Emma	30	London
2	Michael	35	Paris
0	John	45	New York

# Día 5: Análisis Exploratorio de Datos (EDA) - Parte 2

- Tipos de datos de los dataframes
- ¿Qué es un valor nulo y por qué son tan importantes?
- Renombramiento de columnas e índices
- ¿Cómo combinamos distintos dataframes? (Operación Join)
- La importancia de las fechas en el análisis de datos



# Tipos de datos en los dataframes

Cada columna de un dataframe puede tener un tipo de dato propio de Pandas, estos son los tipos más importantes:

- int64: Enteros, utilizados para representar números sin decimales. Son ideales para contar elementos o trabajar con valores discretos, como edades o cantidades.
- float64: Números de punto flotante, que incluyen decimales. Son utilizados para representar valores continuos, como precios o medidas.
- bool: Valores booleanos, que pueden ser True o False. Se usan para representaciones lógicas o condiciones, como en filtros o decisiones.
- object: Este tipo es usado principalmente para cadenas de texto (strings), pero también puede almacenar datos mixtos. Es el tipo más flexible, pero menos eficiente en términos de memoria.

# Tipos de datos en los dataframes

- datetime64: Representa fechas y horas. Es útil para realizar operaciones temporales, como calcular la diferencia entre dos fechas o extraer componentes de una fecha (día, mes, año).
- timedelta64: Similar a datetime64, pero se usa para representar diferencias de tiempo o duraciones, como la diferencia entre dos fechas o horas.
- category: Tipo utilizado para datos que toman un número limitado de valores posibles (como categorías). Es más eficiente en términos de memoria cuando los datos tienen muchas repeticiones, como en una columna de "género" o "estado civil".



# ¿Qué es un valor nulo y por qué son tan importantes?

Un valor nulo (o NaN, que significa "Not a Number") representa la ausencia de un valor en un conjunto de datos.

Es importante en el análisis de datos porque la presencia de valores nulos puede afectar la precisión y la validez de los resultados.

Los valores nulos pueden surgir por errores de entrada, datos faltantes o simplemente porque no se aplica un valor en una observación específica.

Identificar y tratar adecuadamente los valores nulos es esencial para evitar distorsiones en el análisis, ya sea mediante la imputación de valores, la eliminación de filas o columnas con datos faltantes, o usando técnicas que manejan nulos de manera eficiente.

# Renombramiento de columnas e índices

El renombramiento de columnas e índices en Pandas permite modificar los nombres de las columnas o del índice de un DataFrame para hacerlos más claros o adecuados a las necesidades del análisis.

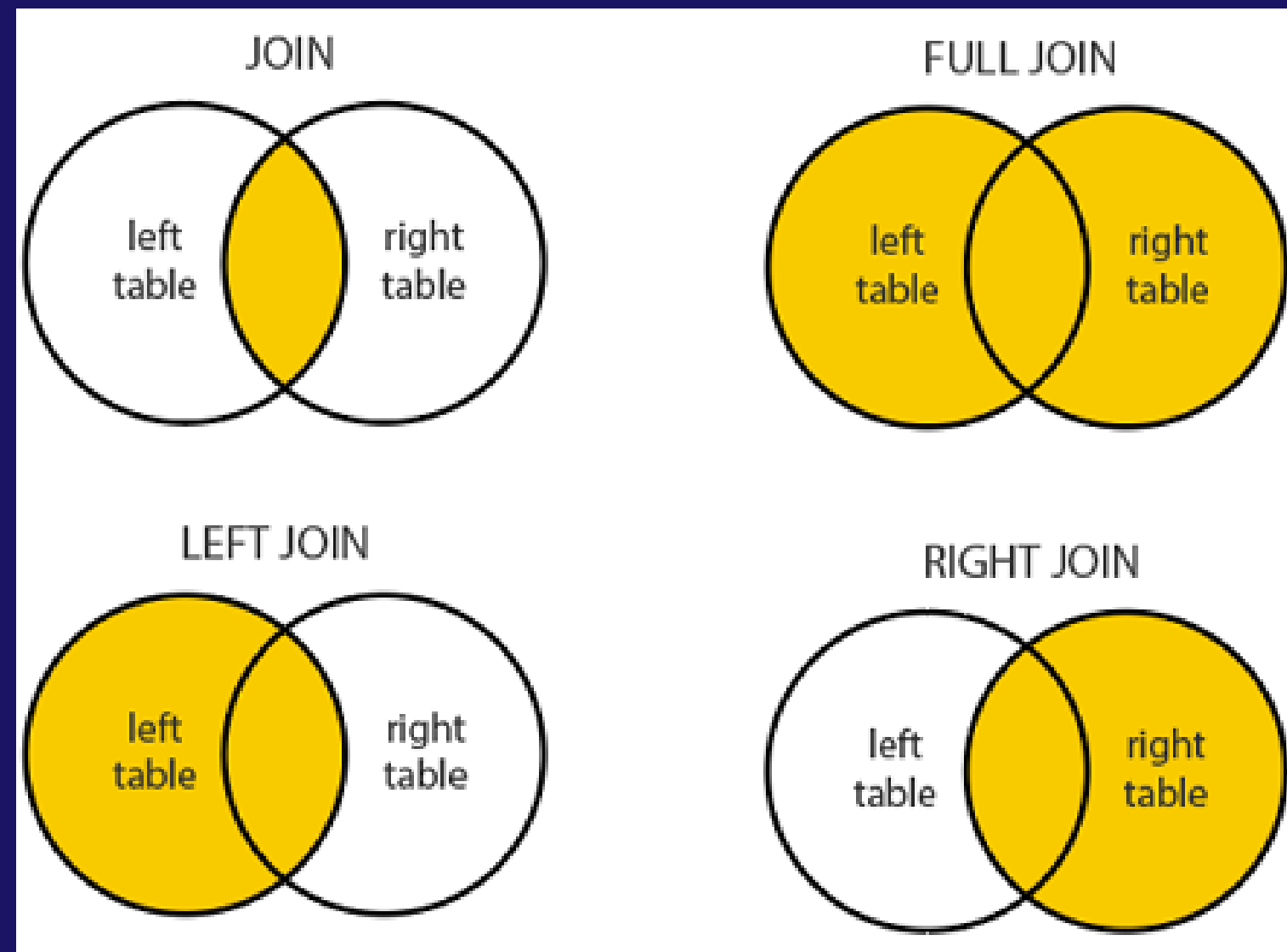
Para renombrar columnas, se utiliza el método rename(), pasando un diccionario donde las claves son los nombres antiguos y los valores los nuevos nombres.

También se puede cambiar el índice de las filas de manera similar, lo que es útil cuando se quiere personalizar las etiquetas o alinear los datos con otros conjuntos de información.

Este proceso facilita la comprensión y mejora la organización de los datos durante el análisis.

# ¿Cómo combinamos distintos dataframes?

Un join es una operación que combina dos DataFrames en función de una o más columnas comunes, como si estuvieras emparejando dos listas según un identificador compartido. Por ejemplo, imagina que tienes un dataframe de estudiantes con sus nombres y DNI, y otro dataframe con las calificaciones usando también los DNIs. Un join te permite unir ambas listas para tener toda la información de cada estudiante en un solo dataframe.



# La importancia de las fechas en el análisis de datos

Las fechas son fundamentales en el análisis de datos, ya que permiten estudiar patrones y tendencias temporales, como ventas diarias, comportamiento estacional o crecimiento mensual.

En Pandas, las fechas se manejan con el tipo `datetime64`, que facilita operaciones como filtrar datos por rango de fechas, calcular diferencias entre días o extraer componentes como años, meses o días.

Esto es útil para responder preguntas como "¿Cuál fue el mes con mayores ventas?" o "¿Cuánto tiempo pasó entre dos eventos?".

Existen otros tipos de datos que manejan unidades de tiempo pero son menos utilizados como son: `Period`, `Interval` y `TimeStamp`.

# Día 6: Visualización de Datos con Seaborn

- ¿Qué es un gráfico y para que sirve?
- Gráficos de tendencia
- Mapas de calor
- Gráficos de distribución

# ¿Qué es un gráfico y para que sirve?

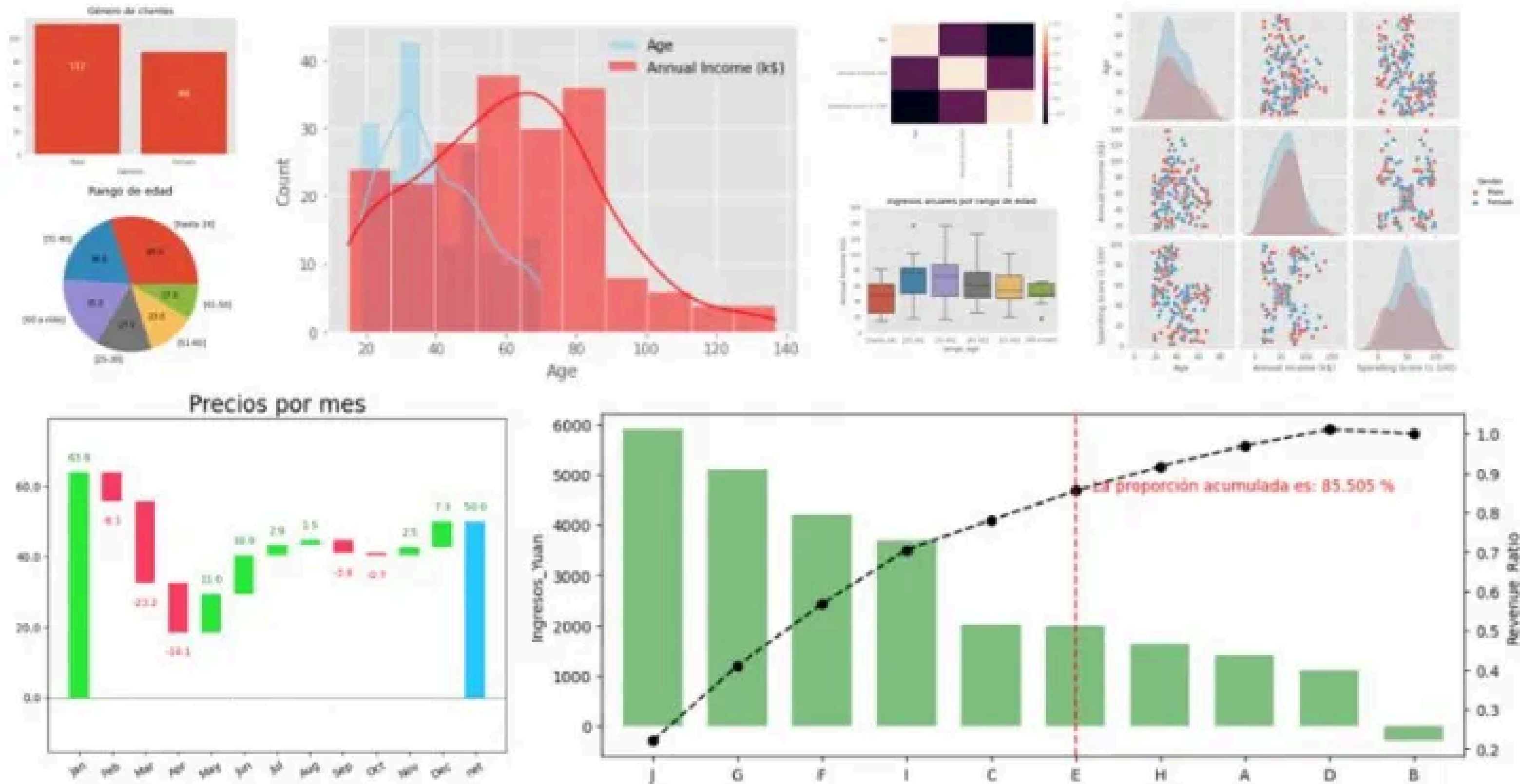
Un gráfico es una representación visual de datos que permite identificar patrones, tendencias y relaciones de manera más clara y rápida que con tablas o números crudos.

Sirve para comunicar información de forma intuitiva, como mostrar el crecimiento de ventas a lo largo del tiempo con una línea o comparar categorías con barras.

Los gráficos ayudan a las empresas porque permiten analizar grandes volúmenes de datos de manera eficiente y descubrir información clave para tomar decisiones estratégicas.

Por ejemplo, un gráfico de ventas puede mostrar qué productos son más populares en ciertos meses o regiones, ayudando a optimizar inventarios y campañas de marketing.

# ¿Qué es un gráfico y para que sirve?





# Gráficos de tendencia

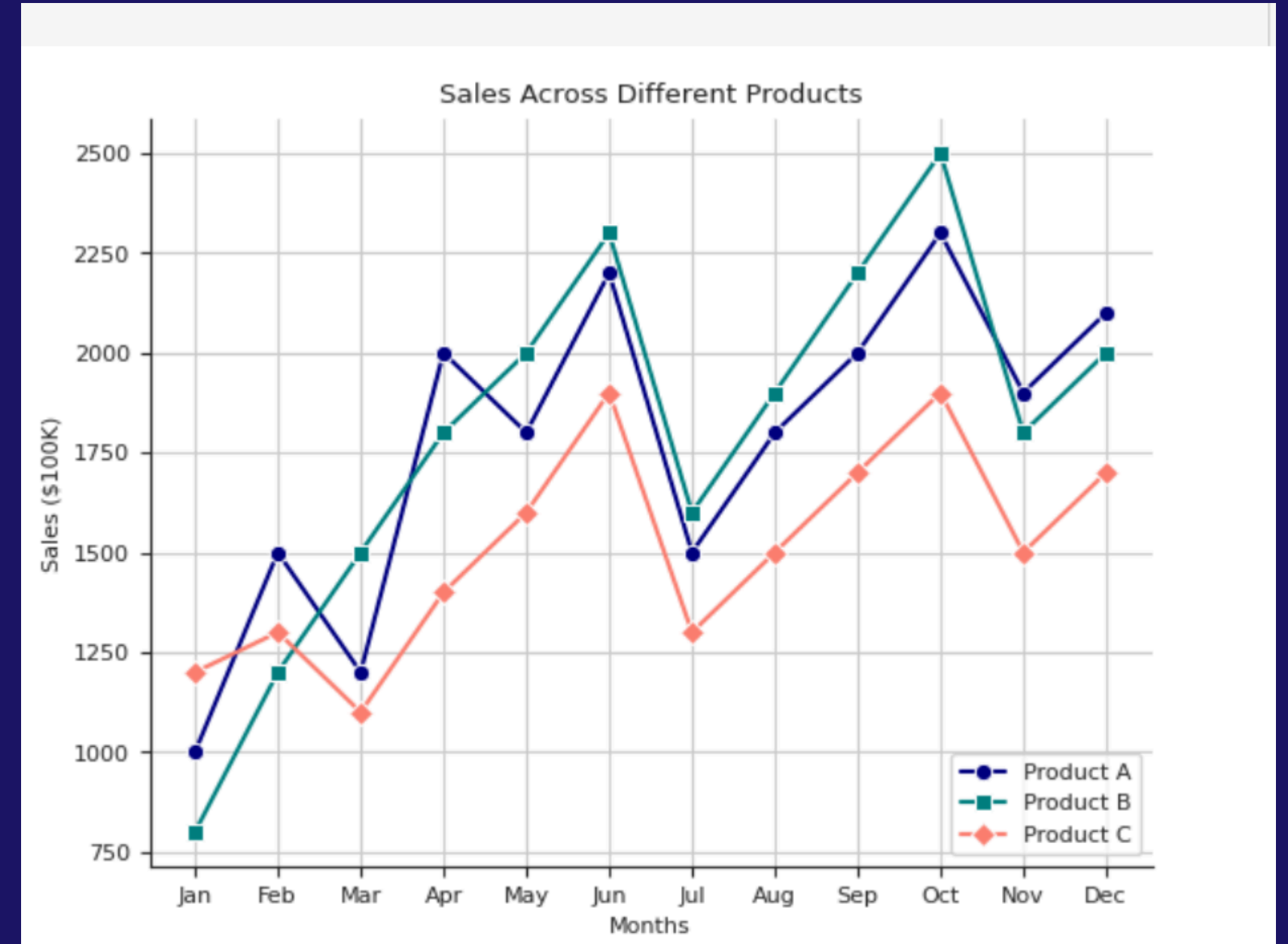
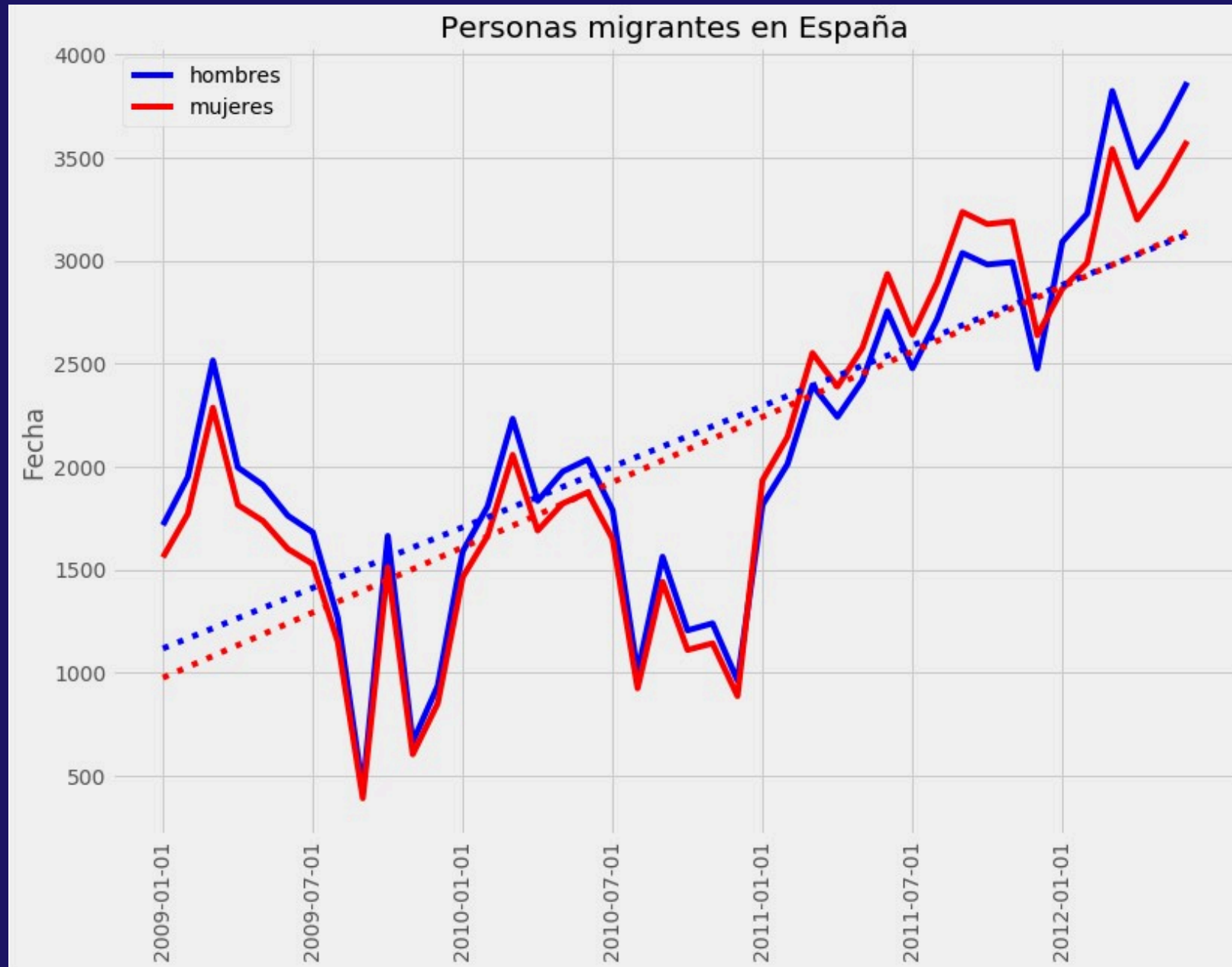
Los gráficos de tendencia muestran cómo cambian los datos a lo largo del tiempo, ayudando a identificar patrones, picos o caídas en series temporales.

Por ejemplo, un gráfico de líneas que registra las ventas mensuales puede revelar si hay un crecimiento constante, estacionalidad o períodos de declive.

Estos gráficos son esenciales para tomar decisiones informadas, ya que permiten prever comportamientos futuros basados en datos históricos.

Son como una brújula que señala la dirección en la que se mueven los datos a lo largo del tiempo.

# Gráficos de tendencia



# Mapas de calor

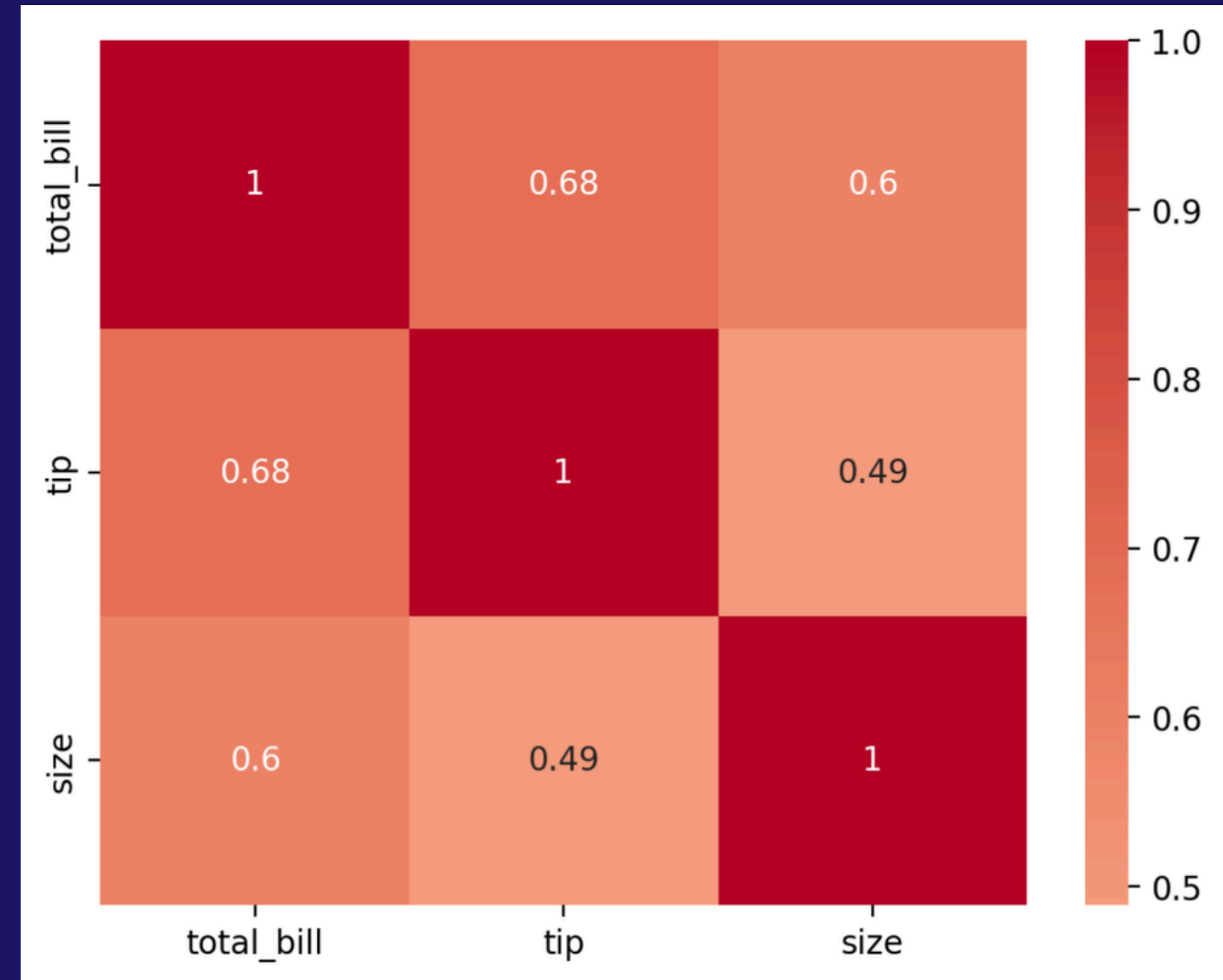
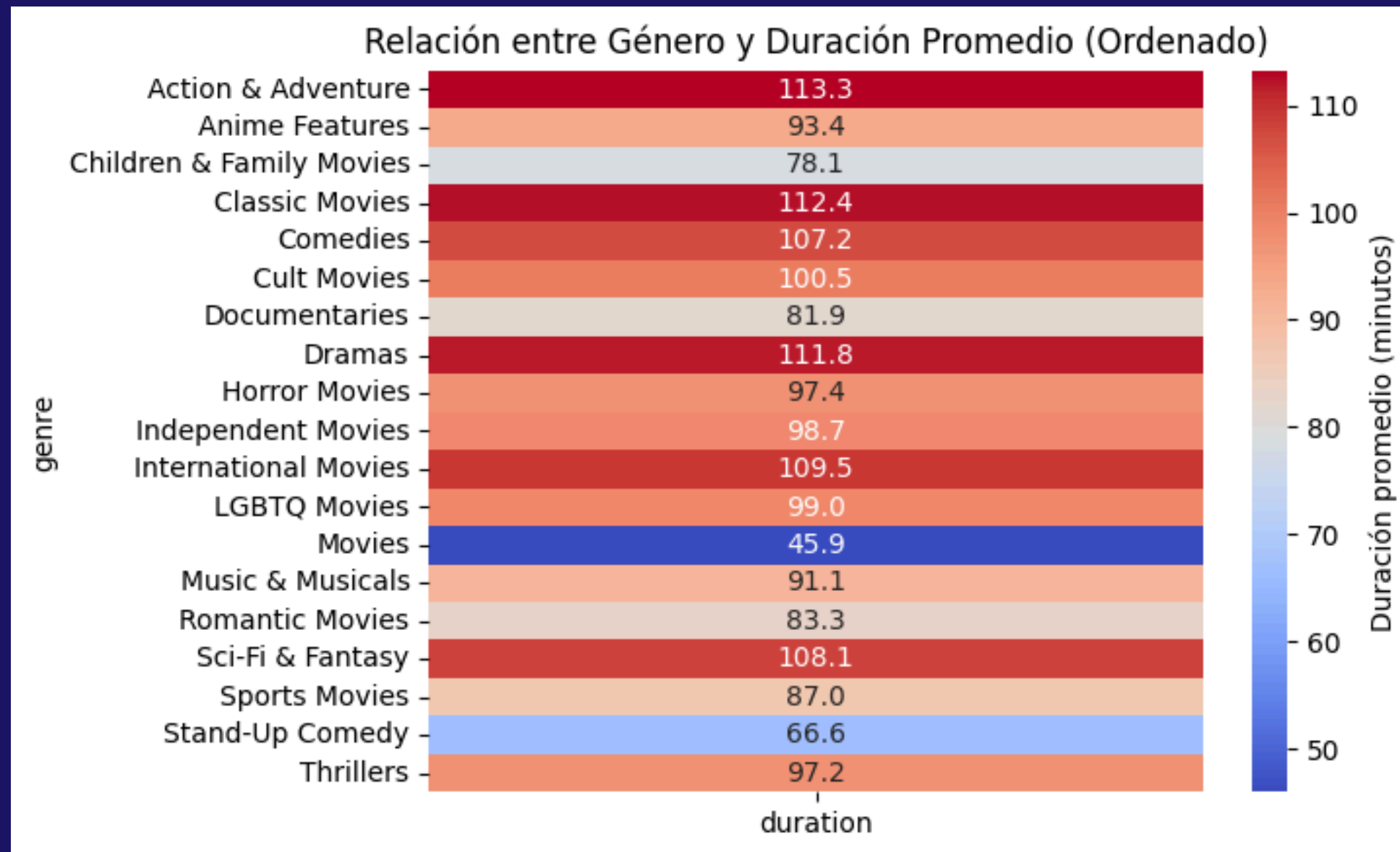
Un mapa de calor es una representación gráfica que muestra la relación entre múltiples variables en una matriz de correlación.

Utiliza una escala de colores para indicar la fuerza y dirección de la correlación entre cada par de variables, también se utiliza para ver la distribución de los valores de una variable frente a otra y ver que rango de valores existen.

Por ejemplo, podríamos comparar en una tienda online el número de visitas a la página con el dato de compras, seguramente veríamos que cuantas más visitas más compras tenemos, en una matriz de correlación.

También podríamos utilizar un mapa de calor para ver los rangos de valores de la duración promedio de películas para cada uno de los géneros.

# Mapas de calor



# Gráficos de distribución

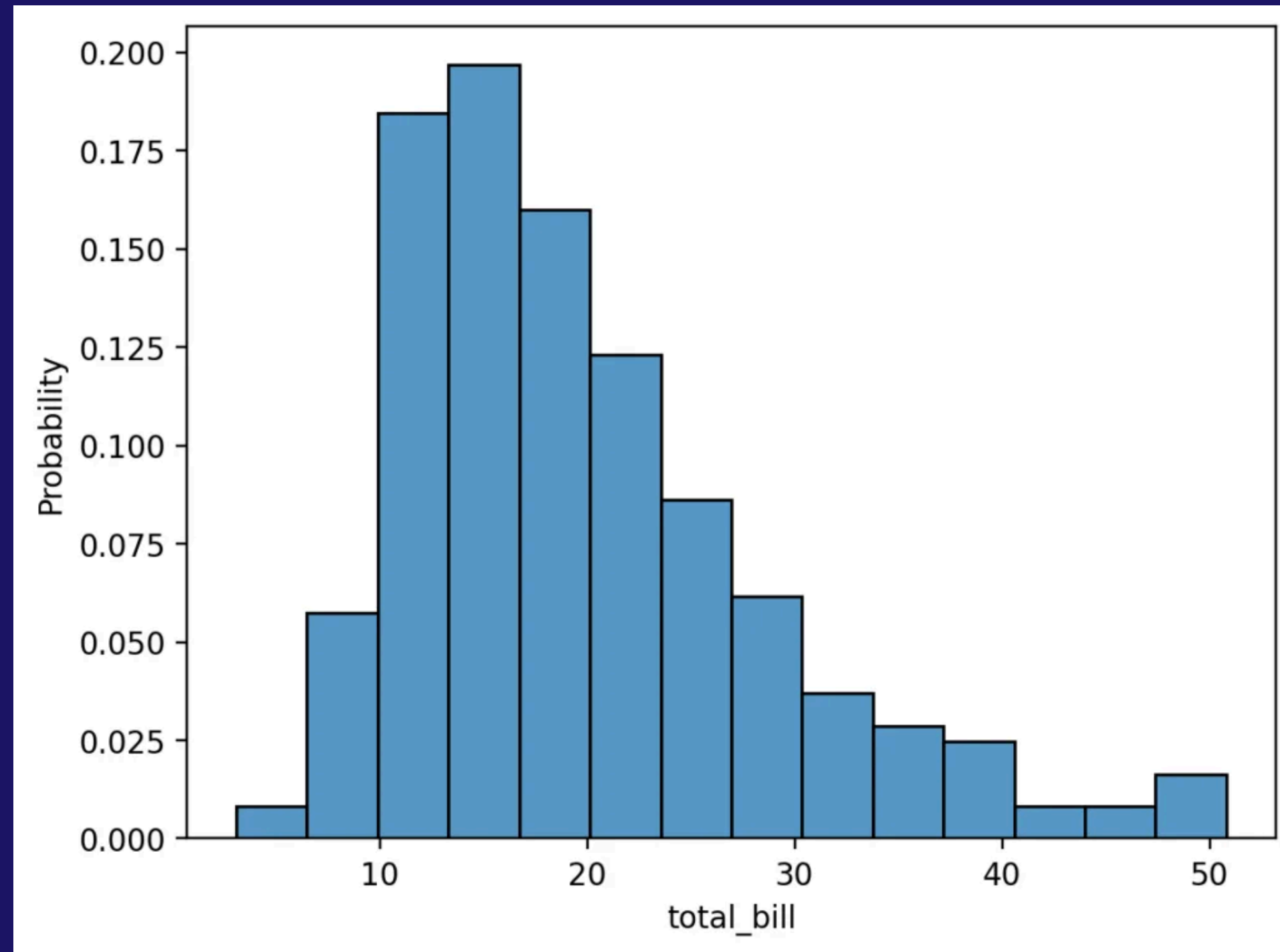
Los gráficos de distribución permiten visualizar cómo se distribuyen los datos en un conjunto, ayudando a identificar patrones, sesgos o anomalías.

Los **histogramas** muestran la frecuencia de los datos dentro de intervalos o "bins", permitiendo observar la distribución de los valores en un conjunto de datos.

Ayudan a entender cómo están distribuidos los valores, si hay sesgo, outliers, y qué patrones pueden encontrarse.

En Pandas, se pueden crear fácilmente utilizando el método `.plot()`, con `kind='hist'` para histogramas y `kind='density'` para gráficos de densidad.

# Gráficos de distribución



Histograma

# Día 7: Proyecto Final: Análisis de artistas en Spotify y Youtube.

- Limpieza de datos
- Análisis descriptivo
- Visualizaciones y conclusiones



# Explicación del proyecto

El proyecto analiza datos de artistas en Spotify y YouTube, explorando métricas como visualizaciones, likes, comentarios y streams.

Incluye tareas de limpieza, análisis descriptivo y visualización para identificar tendencias y relaciones entre plataformas.

Este dataset es muy interesante ya que nos da la visión de un mismo artista en dos plataformas completamente diferentes, Spotify que está dedicada exclusivamente a música y Youtube que es una plataforma de vídeos principalmente.

# Limpieza de Datos

El proyecto comienza con la preparación del dataset.

Se eliminan columnas con más del 3% de valores nulos y se imputan valores faltantes: las columnas numéricas usan la mediana, y las categóricas el valor más frecuente.

Además, se ajustan tipos de datos claves, como convertir "Views" y "Likes" a flotantes y "Key" a entero. Esto asegura que los datos sean consistentes para el análisis.

# Análisis Descriptivo

Se exploran las métricas principales del dataset limpio, calculando estadísticas como media, mediana, y desviación estándar para variables relevantes como "Views" y "Likes".

También se identifican canciones populares con más de 500 millones de visualizaciones, agrupándolas en un subconjunto llamado `top_youtube`, para evaluar tendencias.

# Visualización y Conclusiones

En el tercer punto, se analizan patrones y relaciones entre variables usando visualizaciones como lineplot, heatmap y barplot

Por ejemplo, se evalúa cómo "Views" en YouTube se relaciona con "Streams" en Spotify.

También se exploran diferencias entre categorías como géneros musicales para entender su impacto en métricas como "Likes" y "Comments". Esto permite identificar tendencias clave y formular conclusiones basadas en los datos.