Matplotlib

Luisa Gomez
luisa.gomez@pucp.edu.pe
https://github.com/4591526

PUCP

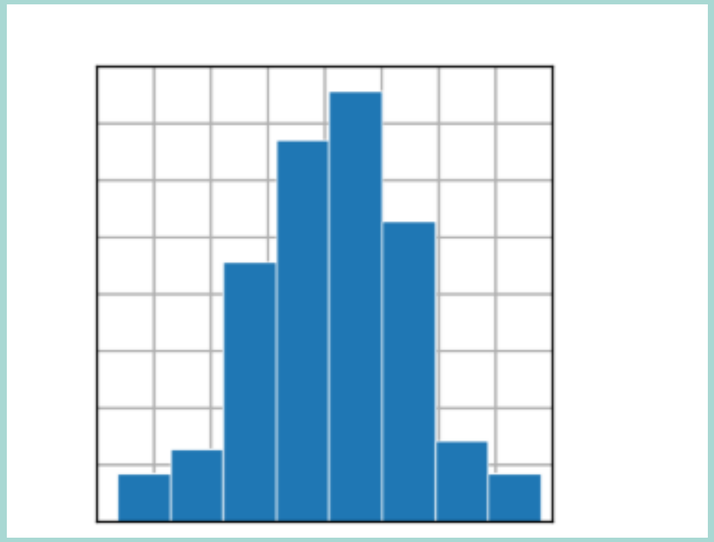
¿Qué es Matplotlib?

- Matplotlib es una biblioteca completa para crear visualizaciones estáticas en Python.
- Ofrece herramientas potentes que permiten desde la creación de gráficos simples hasta visualizaciones altamente complejas.
- Facilita la creación de gráficos con calidad de publicación.
- Ofrece una amplia personalización del estilo visual y del diseño general de las figuras.
- Permite exportar gráficos en múltiples formatos de archivo, como PNG, SVG, PDF, entre otros.

¿Cómo se importa la librería Matplotlib?

- **pyplot** es un módulo dentro de la librería **matplotlib** en Python que proporciona una interfaz basada en funciones para crear gráficos y visualizaciones de manera sencilla
- **plt** es el renombre de pyplot para que sea más rápido de escribir

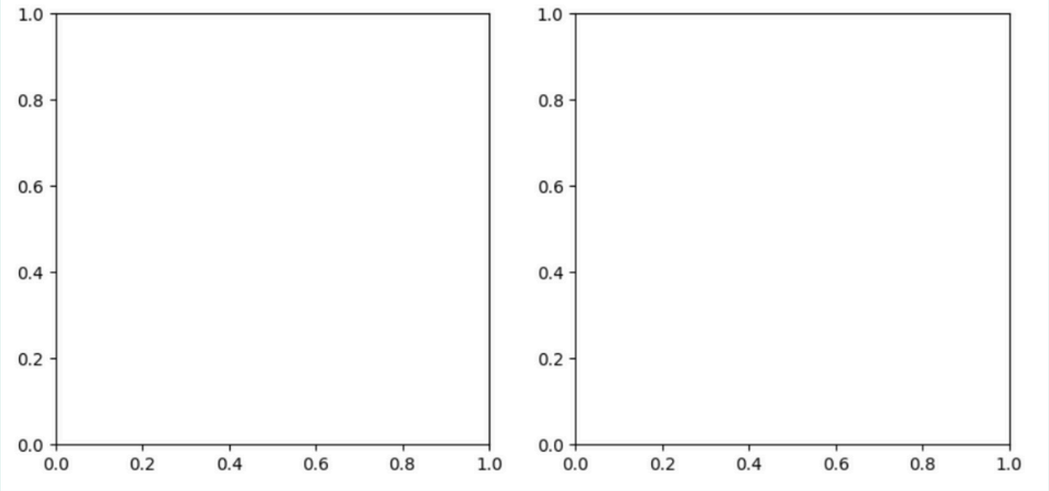
Histograma
plt.hist(X)



- Un histograma divide un conjunto de datos en intervalos (llamados bins) y cuenta cuántos valores caen en cada intervalo.
- Es muy útil para visualizar la distribución de frecuencias de una variable.

Estructura de un histograma

```
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 1)
plt.subplot(2, 2, 2)
plt.show()
```



- **figsize=(10, 10)** significa que la figura ocupará un área de 10x10 pulgadas
- **plt.subplot(2, 2, 1)**

Activa el primer espacio de una cuadrícula de 2 filas y 2 columnas, o sea un total de 4 subgráficos y **1** indica que estamos trabajando con la posición 1 (arriba a la izquierda)

- **plt.subplot(2, 2, 2)**

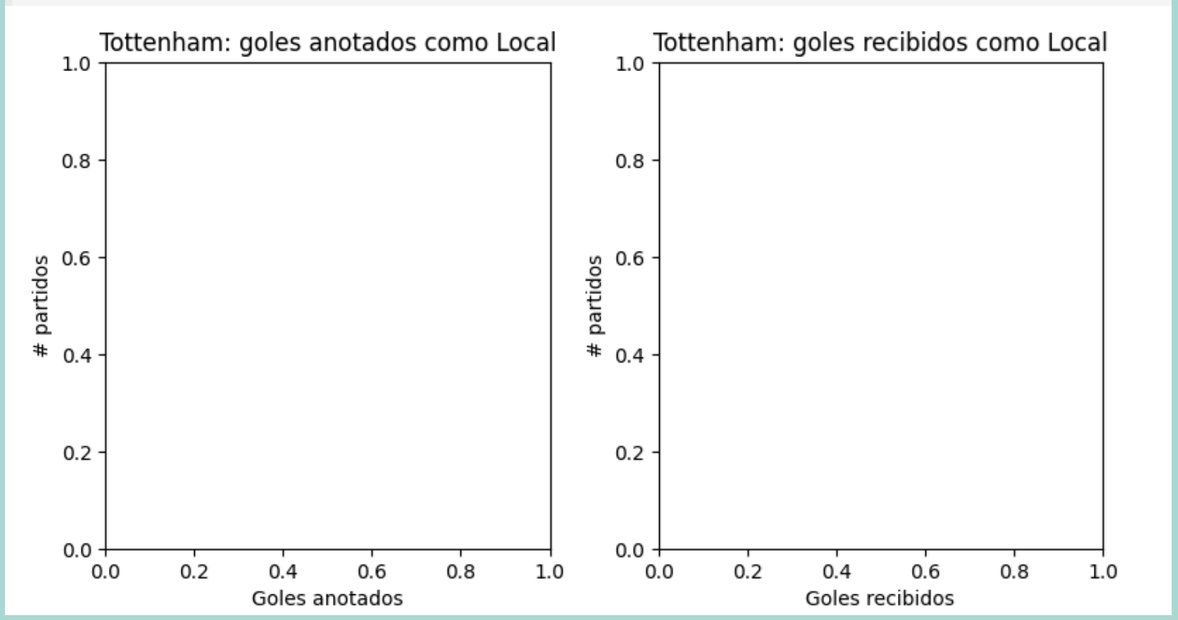
Activa el segundo espacio en la misma grilla de 2x2 → posición 2 (arriba a la derecha)

- **plt.show()**

Muestra la figura completa con los subplots activos hasta ese momento

Estructura de un histograma

```
plt.figure(figsize=(8, 8))
plt.subplot(2, 2, 1)
plt.title('Tottenham: goles anotados como Local')
plt.xlabel('Goles anotados')
plt.ylabel('# partidos')

plt.subplot(2, 2, 2)
plt.title('Tottenham: goles recibidos como Local')
plt.xlabel('Goles recibidos')
plt.ylabel('# partidos')

plt.tight_layout()
plt.show()
```



- **plt.title('Tottenham: goles anotados como Local')**
- **plt.xlabel('Goles anotados')**
- **plt.ylabel('# partidos')**

Estas funciones añaden información al gráfico:

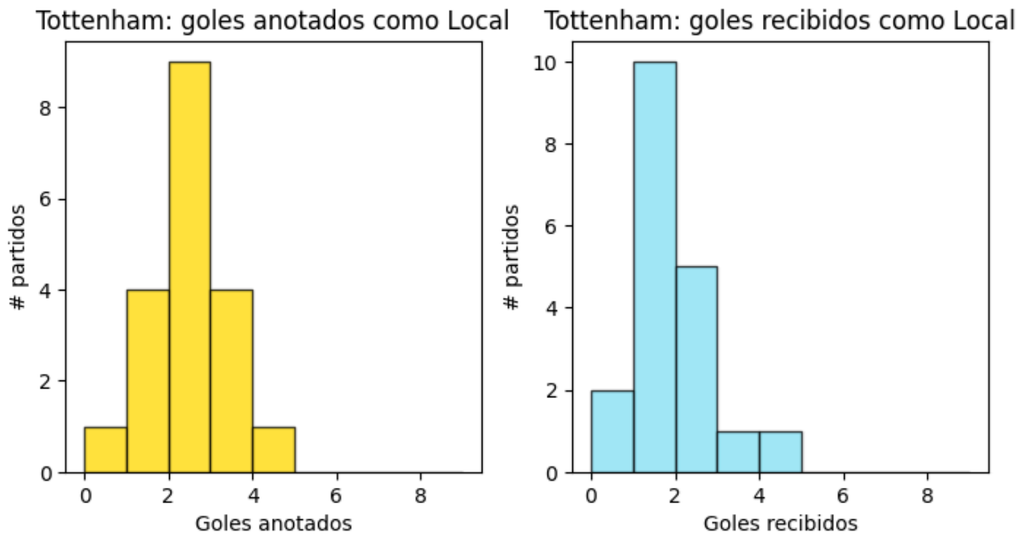
- title: título del gráfico
- xlabel: etiqueta del eje X
- ylabel: etiqueta del eje Y

- **plt.tight_layout()**

Ajusta automáticamente los márgenes y espacios entre los subgráficos para que no se sobrepongan las etiquetas ni títulos.

Estructura de un histograma

```
plt.figure(figsize=(7, 7))
plt.subplot(2, 2, 1)
plt.hist(goles_anotados_local, bins=range(0, 10), alpha=0.75, color='gold', edgecolor='black')
plt.title('Tottenham: goles anotados como Local')
plt.xlabel('Goles anotados')
plt.ylabel('# partidos')

plt.subplot(2, 2, 2)
plt.hist(goles_recibidos_local, bins=range(0, 10), alpha=0.75, color='#80e2f6', edgecolor='black')
plt.title('Tottenham: goles recibidos como Local')
plt.xlabel('Goles recibidos')
plt.ylabel('# partidos')

plt.tight_layout()
plt.show()
```

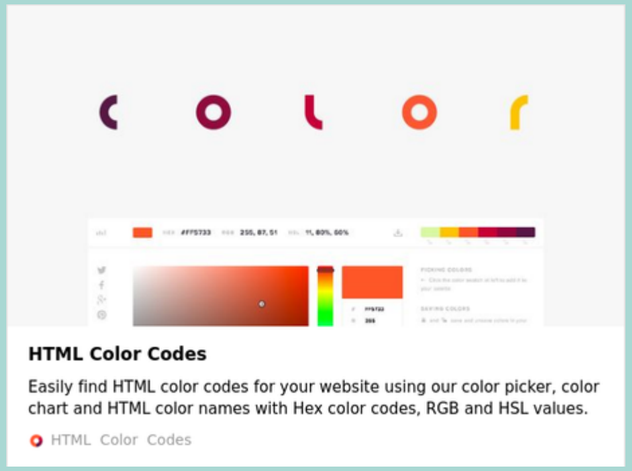


- **plt.hist(goles_anotados_local, bins=range(0, 10), alpha=0.75, color='gold', edgecolor='black')**
  - goles_anotados_local: lista con los goles anotados en partidos como local
  - bins=range(0, 10): los datos se agrupan en intervalos (bins) que van del 0 al 9
  - alpha=0.75: da una ligera transparencia al color
  - color='gold': color dorado para las barras
  - edgecolor='black': bordes negros para que las barras sean más visibles

Definir colores en un gráfico

Se puede definir colores en matplotlib de varias formas, y combinar tanto los nombres estándar de colores en inglés, como los códigos HTML (hexadecimales).

https://matplotlib.org/stable/gallery/color/named_colors.html



¿Cómo guardar mis gráficos?

- **plt.savefig(...)** → Guarda la figura actual (la que hayas construido con plt.figure() y plt.plot()/plt.hist(), etc.) en un archivo de imagen.
- 'goles_Tottenham.png' → Nombre del archivo de salida.
- El formato lo determina la extensión:
  - .png → imagen tipo PNG (sin pérdida, fondo transparente si se quiere)
  - .jpg o .jpeg → formato con compresión
- **dpi=300** → Significa dots per inch (puntos por pulgada)
  - Define la resolución de la imagen
  - 300 dpi es ideal para impresión de alta calidad
  - Para uso en web o presentaciones, 100–150 dpi suele ser suficiente

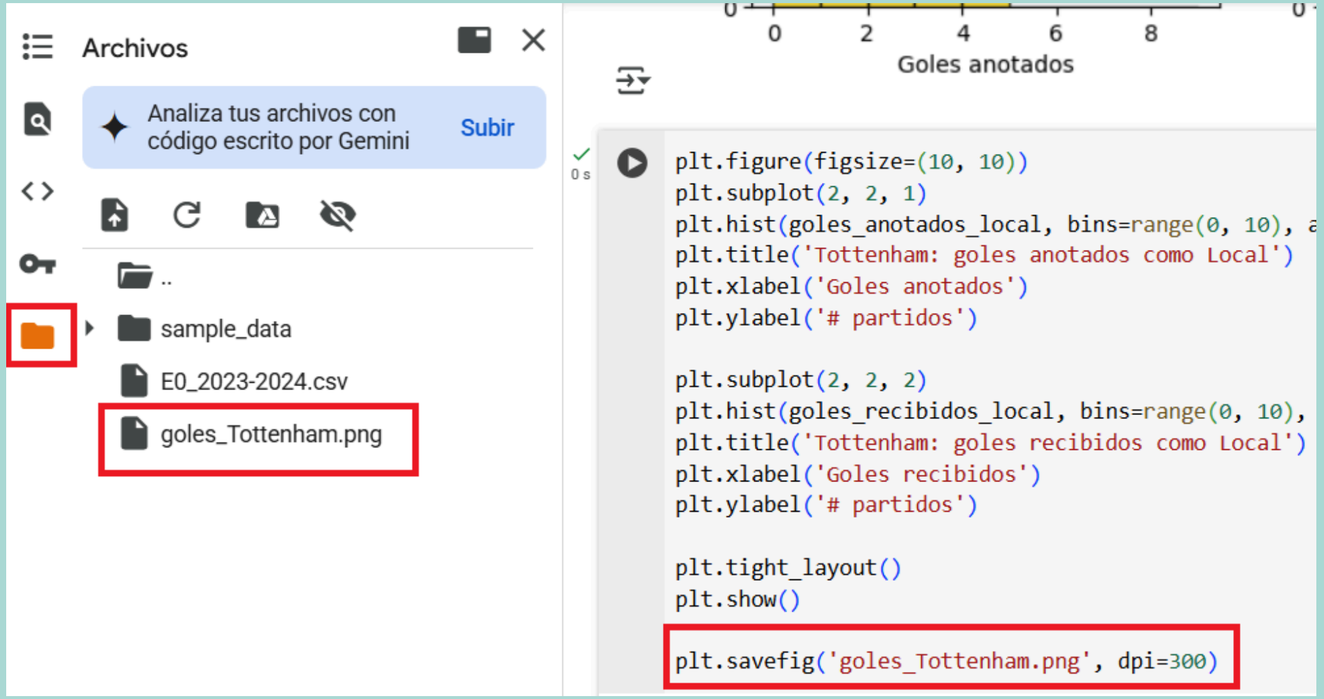
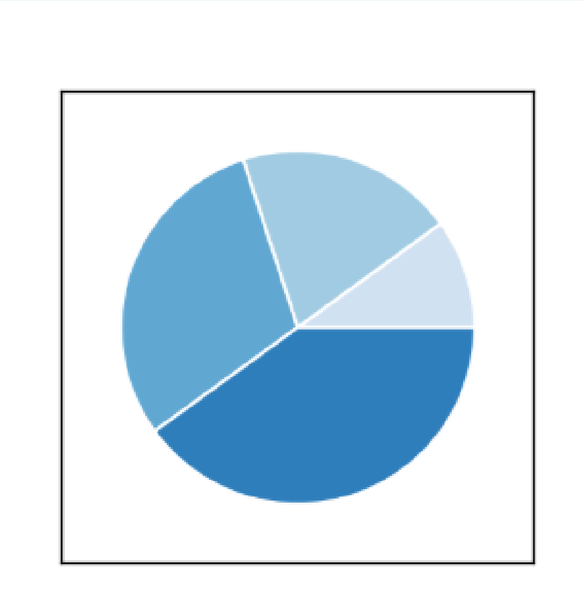


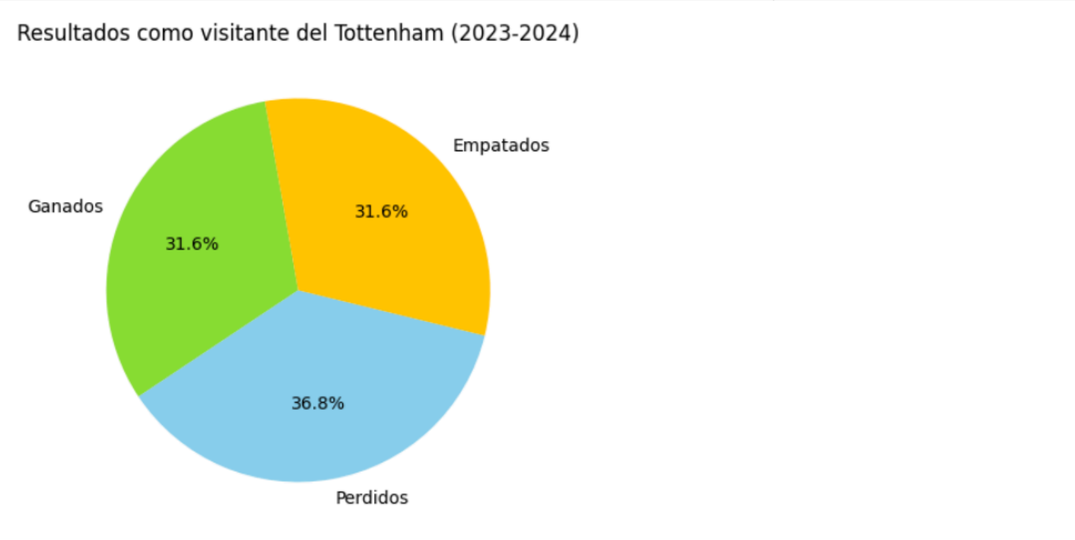
Gráfico de pastel
plt.pie(X)



- La función **plt.pie(X)** de matplotlib.pyplot se utiliza para crear un gráfico de pastel (pie chart) a partir de una lista de valores numéricos.
- Es útil cuando quieres mostrar proporciones o porcentajes de un todo.

Estructura de un pie chart

```
labels = ['Ganados', 'Perdidos', 'Empatados']
sizes = [ganados, perdidos, empatados]
colors = ['#88dc33', 'skyblue', '#FFC300']
explode = (0.0, 0.0, 0.0)

plt.figure(figsize=(5, 5))
plt.title('Resultados como visitante del Tottenham (2023-2024)')
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False, startangle=100)
plt.show()
```

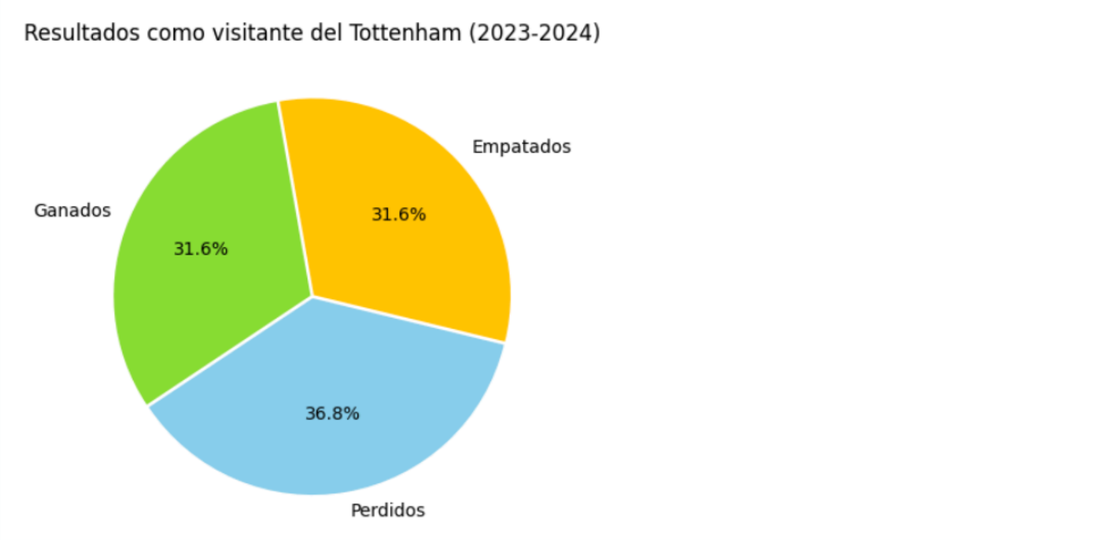


- labels = ['Ganados', 'Perdidos', 'Empatados'] → Lista de etiquetas para cada sector del gráfico (lo que aparecerá al lado de cada porción del pastel)
- sizes = [ganados, perdidos, empatados] → Lista con los valores numéricos reales
  - Estos datos deben estar definidos previamente (por ejemplo, ganados = 5, perdidos = 7, etc.).
- colors = ['#88dc33', 'skyblue', '#FFC300'] → Colores personalizados para cada sector del pastel
- **explode = (0.0, 0.0, 0.0)** → Controla si quieres separar alguno de los sectores del pastel
  - En este caso, todos los sectores están juntos (sin separación).

plt.pie(
sizes, # Datos numéricos
explode=explode, # Separación de sectores
labels=labels, # Etiquetas
colors=colors, # Colores
autopct='%1.1f%%', # Muestra porcentaje con 1 decimal (ej. 33.3%)
shadow=False, # No aplica sombra
startangle=100 # Rota el gráfico para que el primer sector empiece a las 10 en punto
)

¿Cómo separar los sectores del pie chart?

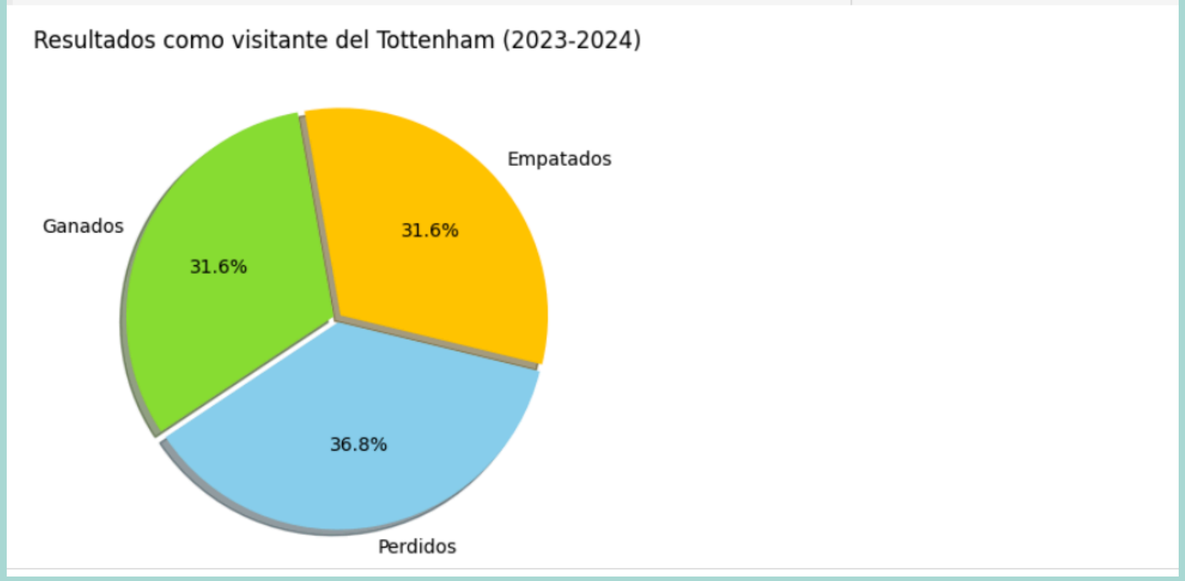
```
labels = ['Ganados', 'Perdidos', 'Empatados']
sizes = [ganados, perdidos, empatados]
colors = ['#88dc33', 'skyblue', '#FFC300']
explode = (0.01, 0.01, 0.01)

plt.figure(figsize=(5, 5))
plt.title('Resultados como visitante del Tottenham (2023-2024)')
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False, startangle=100)
plt.show()
```



- **explode = (0.01, 0.01, 0.01)** → Cada valor indica cuánto se separa ese sector del centro
    - **explode = (0.1, 0, 0)** → solo el primer sector se separa bastante.
    - **explode = (0.01, 0.01, 0.01)** → todos los sectores se separan un poquito.
    - **explode = (0, 0, 0)** → todos quedan juntos (por defecto).

¿Cómo aplicar sombra al pie chart?

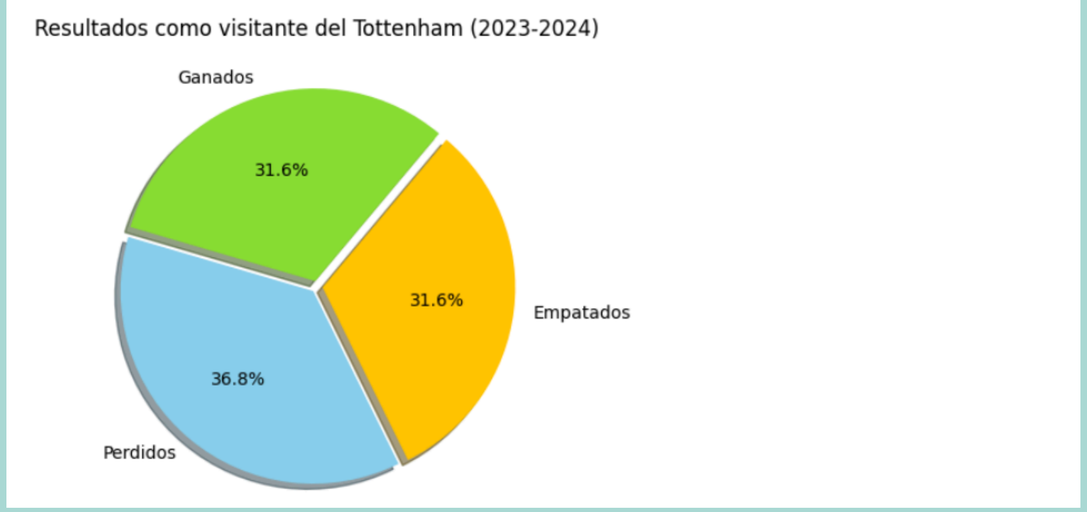
```
labels = ['Ganados', 'Perdidos', 'Empatados']
sizes = [ganados, perdidos, empatados]
colors = ['#88dc33', 'skyblue', '#FFC300']
explode = (0.02, 0.02, 0.02)

plt.figure(figsize=(5, 5))
plt.title('Resultados como visitante del Tottenham (2023-2024)')
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=100)
plt.show()
```



- Si **shadow=False** (valor por defecto), el gráfico se ve plano.
- Si **shadow=True**, se dibuja una sombra proyectada detrás del gráfico, dándole un poco de profundidad.

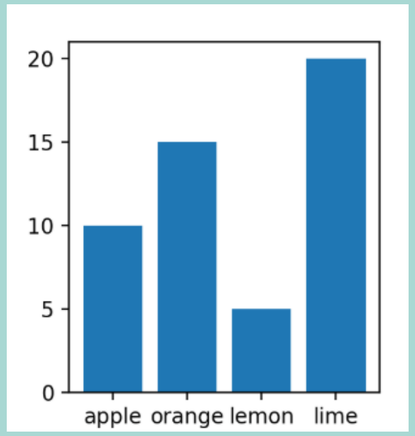
¿Cómo rotamos el pie chart?

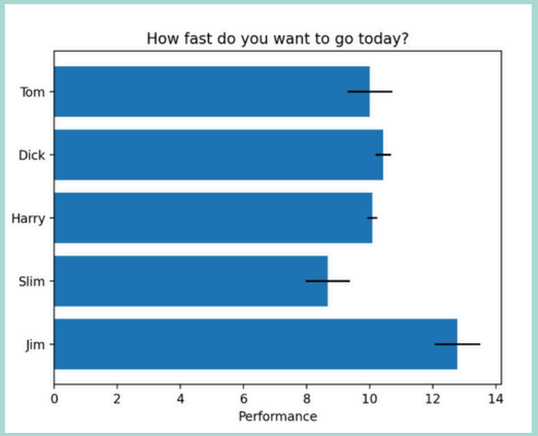
```
labels = ['Ganados', 'Perdidos', 'Empatados']
sizes = [ganados, perdidos, empatados]
colors = ['#88dc33', 'skyblue', '#FFC300']
explode = (0.03, 0.03, 0.03)

plt.figure(figsize=(5, 5))
plt.title('Resultados como visitante del Tottenham (2023-2024)')
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=50)
plt.show()
```



- El valor se mide en grados, en sentido antihorario desde el eje horizontal derecho (0°).
- **startangle=0** → el primer sector comienza en el lado derecho (como las 3 en un reloj).
- **startangle=90** → empieza desde arriba (como las 12 en un reloj).
- **startangle=50** → el primer sector empieza un poco antes de la posición superior derecha, girado 50° en sentido antihorario.

Gráfico de barras



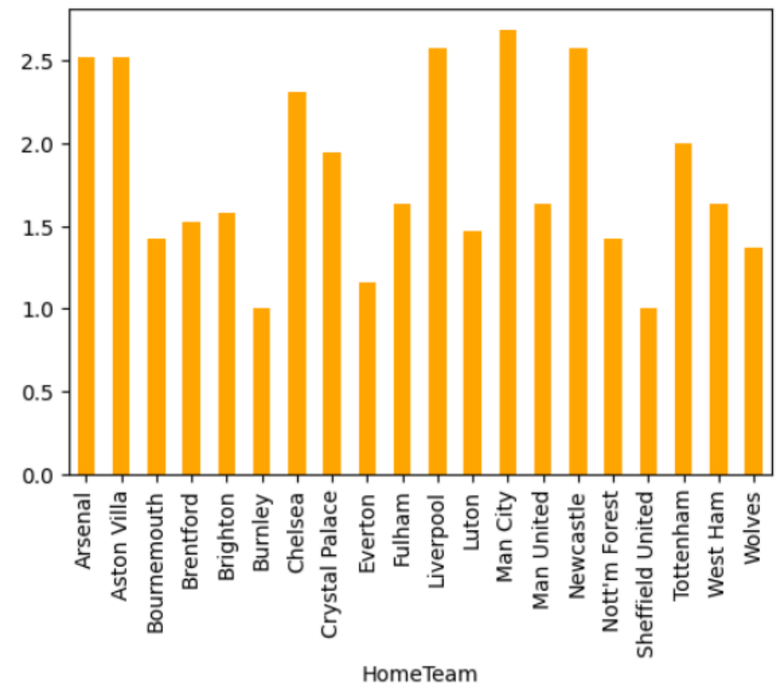


- Las barras verticales en matplotlib.pyplot se crean con gráficos de tipo bar, y son muy útiles para comparar cantidades categóricas de forma visual y clara.
- **kind='bar'** → Cuando usas pandas para graficar, puedes hacer algo como: **df.plot(kind='bar')**
  - **kind='bar'** le dice a pandas que deseas un gráfico de barras verticales.
  - Es equivalente a usar directamente **plt.bar()** con listas o series.

- Comparar valores de diferentes categorías (por ejemplo, goles por equipo, población por país, ventas por producto).
- Mostrar frecuencias o cantidades fácilmente interpretables.

Estructura del gráfico de barras verticales

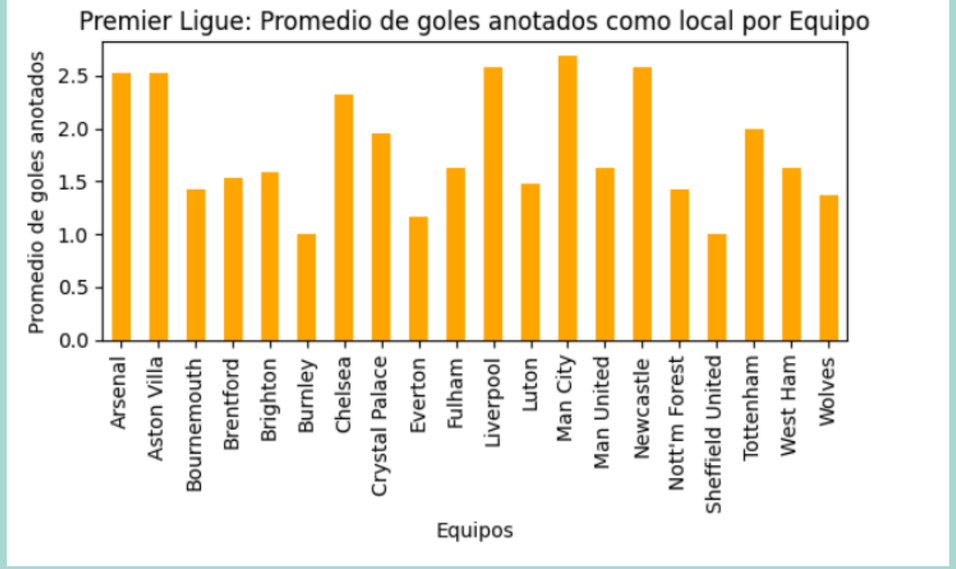
```
promedio_goles_por_equipo.plot(kind='bar', figsize=(6, 4), color='orange')

plt.show()
```



- promedio_goles_por_equipo.**plot(...)**
  - promedio_goles_por_equipo es una Serie o DataFrame de pandas donde las etiquetas (índice) son los nombres de los equipos, y los valores son los promedios de goles.
  - **kind='bar'** → Crea un gráfico de barras verticales (en vez de un gráfico de líneas, que es el valor por defecto)
  - figsize=(6, 4) → Define el tamaño de la figura en pulgadas: 6 de ancho y 4 de alto
  - color='orange' → Da a las barras el color naranja

¿Cómo cambiar las etiquetas del gráfico de barras verticales?

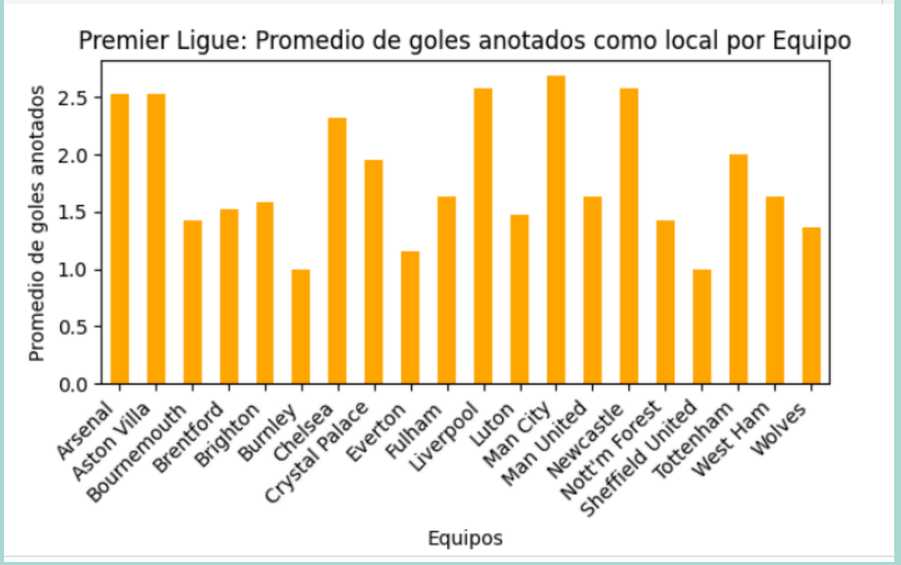
```
promedio_goles_por_equipo.plot(kind='bar', figsize=(6, 4), color='orange')

plt.title('Premier Ligue: Promedio de goles anotados como local por Equipo')
plt.xlabel('Equipos')
plt.ylabel('Promedio de goles anotados')

plt.tight_layout()

plt.show()
```



- **plt.title(...)** → Agrega un título al gráfico, que aparece en la parte superior
- **plt.xlabel(...)** → Agrega una etiqueta al eje X (horizontal)
- **plt.ylabel(...)** → Agrega una etiqueta al eje Y (vertical)
- **plt.tight_layout()** → Ajusta automáticamente los márgenes y espaciado del gráfico para que nada quede cortado y todo el contenido (título, ejes, etiquetas) se vea correctamente

¿Cómo cambiar la orientación de las etiquetas?

```
promedio_goles_por_equipo.plot(kind='bar', figsize=(6, 4), color='orange')

plt.title('Premier Ligue: Promedio de goles anotados como local por Equipo')
plt.xlabel('Equipos')
plt.ylabel('Promedio de goles anotados')

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.show()
```

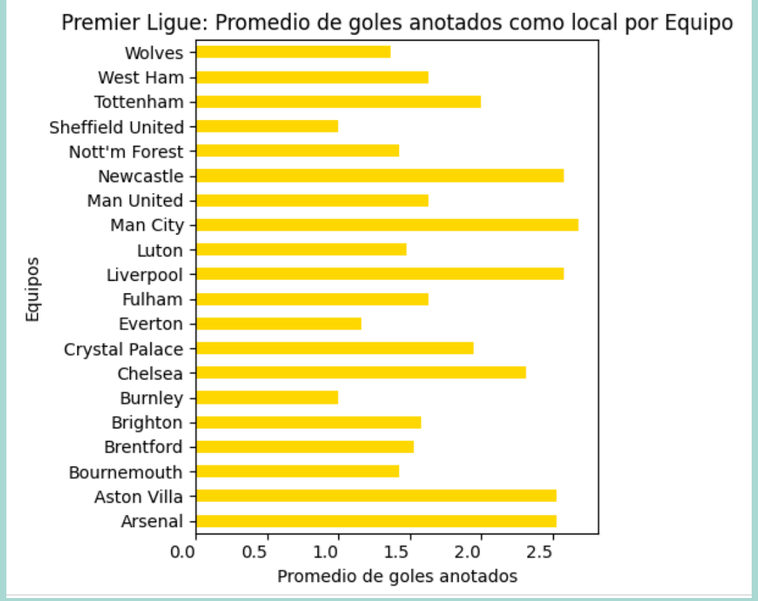


- **plt.xticks(**rotation=45, ha='right'**)** → sirve para modificar la apariencia de las etiquetas del eje X en un gráfico
  - Ajusta la orientación y alineación horizontal de las etiquetas en el eje X

- **rotation=**45 → Rota las etiquetas del eje X a 45 grados en sentido antihorario
  - Esto es útil cuando las etiquetas son largas o están muy juntas, porque evita que se sobrepongan.
- **ha**='right'
  - **ha** significa alineamiento horizontal
  - 'right' alinea el extremo derecho del texto con su posición en el eje X
  - Esto hace que las etiquetas se vean más ordenadas después de rotarlas.

¿Cómo cambiar la orientación de las barras?

```
promedio_goles_por_equipo.plot(kind='barh', figsize=(5, 5), color='gold')

plt.title('Premier Ligue: Promedio de goles anotados como local por Equipo')
plt.ylabel('Equipos')
plt.xlabel('Promedio de goles anotados')

plt.xticks(rotation=0, ha='right')

plt.tight_layout()

plt.show()
```



- promedio_goles_por_equipo.**plot(kind='barh', ...)**
  - **kind='barh'**: Crea barras horizontales en lugar de verticales (bar)
  - Esto es útil cuando los nombres de los equipos son largos, ya que se leen mejor en el eje Y.
- **plt.ylabel**('Equipos') → Etiqueta del eje Y (vertical)
- **plt.xlabel**('Promedio de goles anotados') → Etiqueta del eje X (horizontal)

Revisa el siguiente link si deseas ver otros gráficos que puedes realizar con matplotlib

https://matplotlib.org/stable/gallery/index.html

Revisa el siguiente link si deseas ver el Colab con los ejemplos presentados en esta presentación