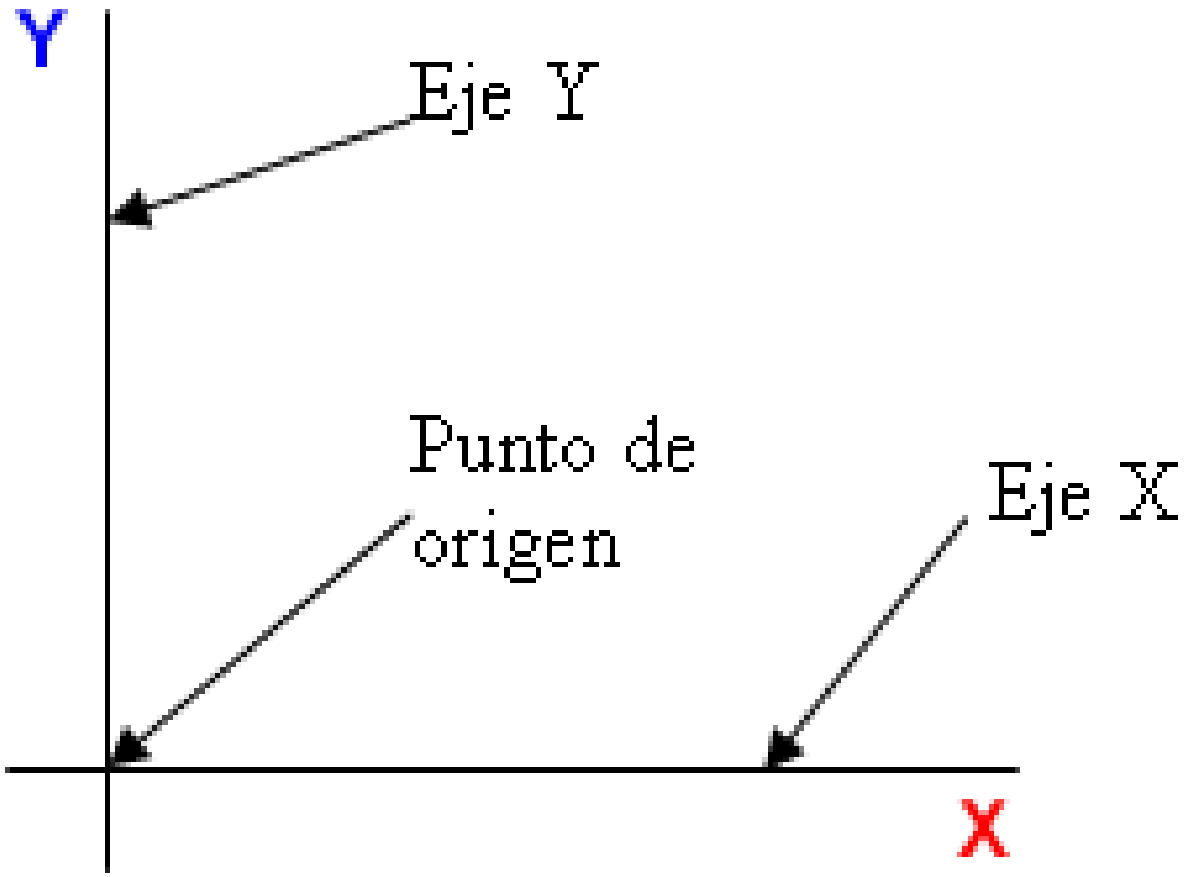
Indice

Graficos



Histograma

Concepto

Un histograma es un resumen de la variación en una variable medida. Muestra el número de ejemplos que ocurren en una categoría. Un histograma es un tipo de distribución de frecuencia.

Pueden crear un histograma usando Matplotlib, Pandas o Seaborn. El código de abajo usa una combinación de Pandas y Matplotlib.

Librerías en Python

La librería de Python para gráficos más popular es **matplotlib.pyplot**.

```
import matplotlib.pyplot as plt
```

También tenemos la biblioteca **seaborn** la cual ofrece gráficos con mejor estética.

```
import seaborn as sns
```

Casos de uso

Los histogramas funcionan dividiendo todo el rango de valores en una serie de intervalos y luego contando cuántos valores caen en cada intervalo. Si bien los intervalos suelen tener el mismo tamaño, no es necesario que lo tengan.

En pocas palabras, los histogramas nos grafica en barras la frecuencia en la que se repiten los datos en un conjunto de datos.

Tips:

Tenga en cuenta que los gráficos en matplotlib reciben datos tipo arrays de numpy por lo cual debemos convertir los dataframe en un conjunto de datos de numpy. Este es muy fácil de hacer, solo se toman los datos que deseamos gráficas, lo pasamos como

parámetro en el array de numpy y lo almacenamos en una variable para posterior a esto utilizarlo en nuestros gráficos. Debajo un breve ejemplo.

DataFrame

	hombre
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9

Dataframe a Array Numpy

```
[13] arr = np.array(df['hombre'])
     arr
```

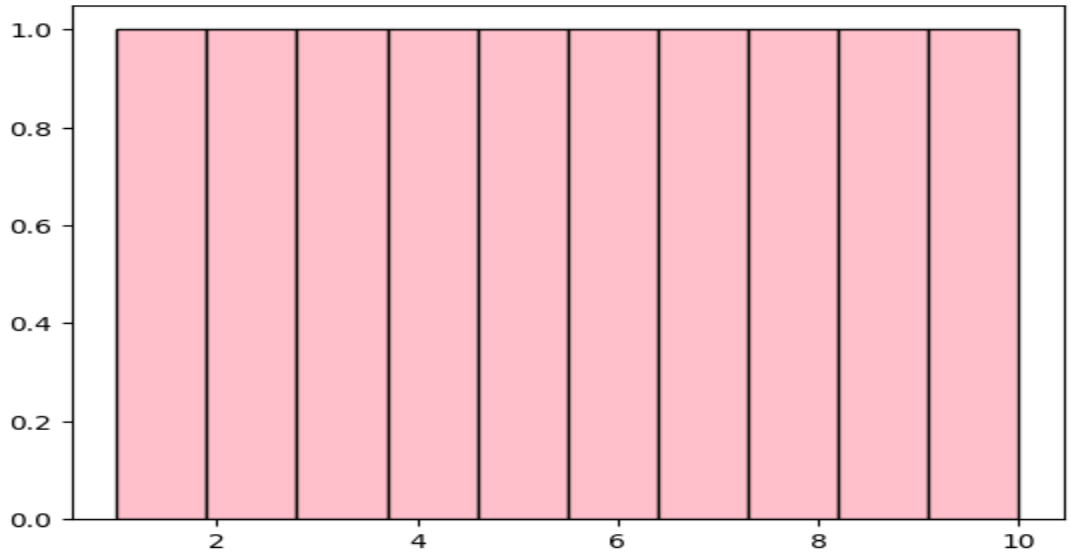
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Ejemplo

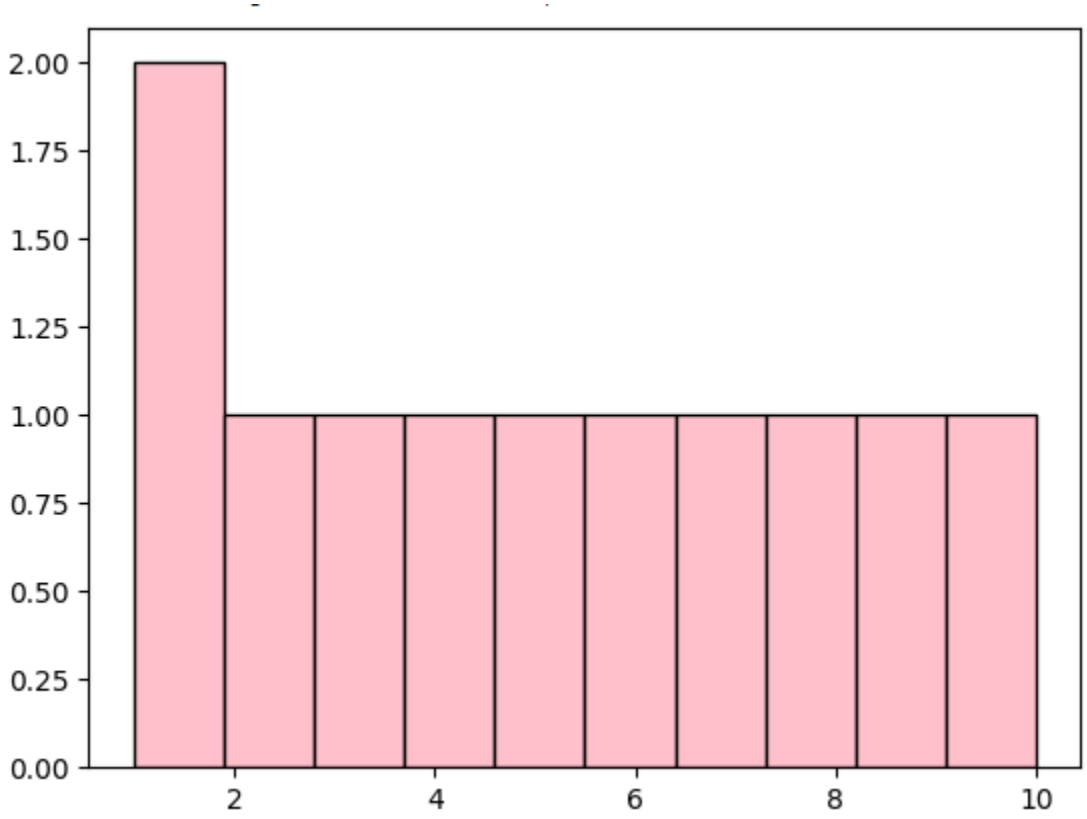
plt.hist(arr(Conjunto de datos), color="pink"(color de las barras de frecuencia), ec="black"(color de los bordes de las barras de frecuencia)) existen más parámetros pero estos son los más utilizados.

Quedaría de la siguiente manera:

```
plt.hist(arr, color="pink", ec="black")
```



El gráfico anterior nos muestra la cantidad de hombres que existe en cada número del conjunto de datos, es decir del 1 al 10 solo hay una frecuencia por número en el eje X, si en caso contrario en la posición 1 al 2 existiera más de una frecuencia, entonces este sobre saldría de los demás que solo tiene una sola frecuencia. Ejemplo:

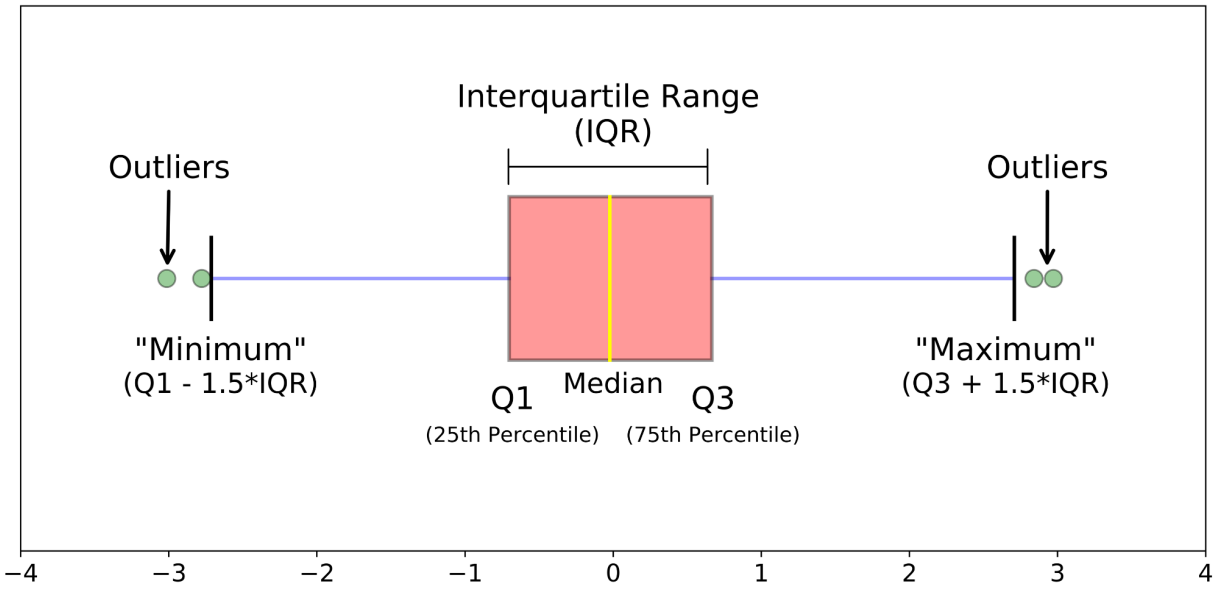


Boxplots

Concepto

Un boxplot es un tipo de gráfico que se utiliza para representar la distribución de un conjunto de datos **numéricos** a través de cinco estadísticas resumidas: el valor mínimo, el primer cuartil, la mediana, el tercer cuartil y el valor máximo. Los parámetros que recibe un boxplot son los siguientes:

Mediana (percentil 2/50): el valor medio del conjunto de datos.
Primer cuartil (percentil 1/25): el número medio entre el número más pequeño (no el "mínimo") y la mediana del conjunto de datos.
Tercer cuartil (percentil 3/75): el valor medio entre la mediana y el valor más alto (no el "máximo") del conjunto de datos.
Rango intercuartil (RIQ): del percentil 25 al 75. bigotes (se muestran en azul)
valores atípicos (se muestran en círculos verdes)
"máximo": Q3 + 1.5 * RIQ
"mínimo": Q1 -1.5 * RIQ

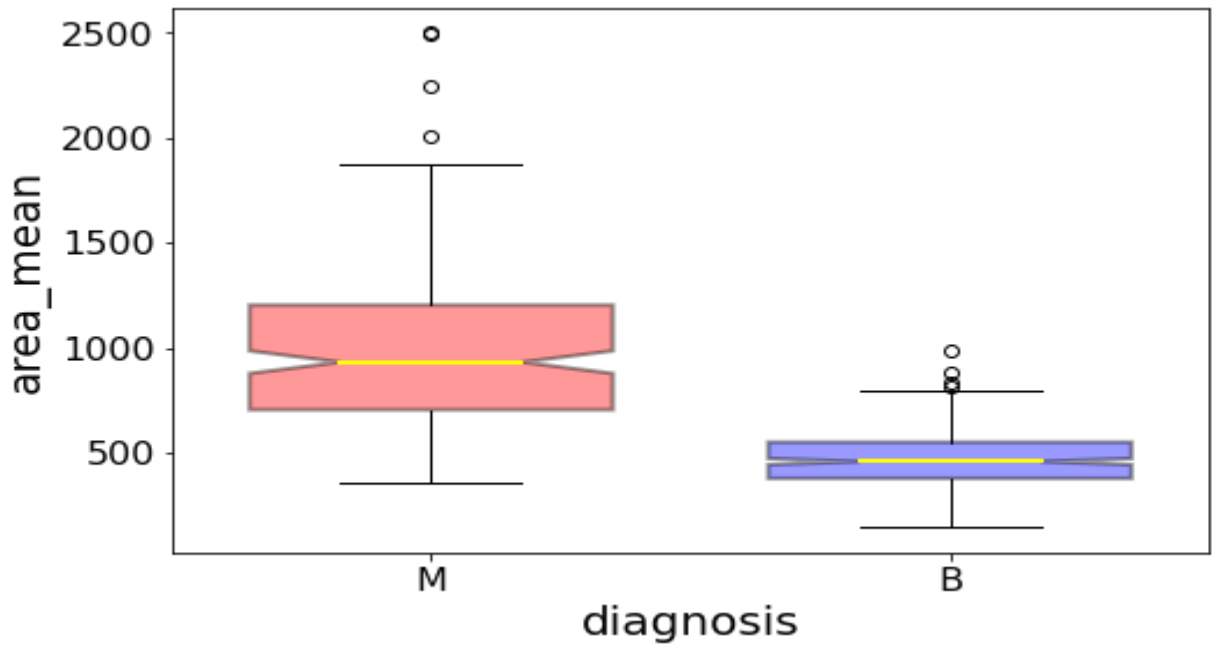



```
malignant = df.loc[df['diagnosis']=='M','area_mean']
benign = df.loc[df['diagnosis']=='B','area_mean']
fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize =
(8,5))
```

```
boxplots = axes.boxplot([malignant,benign](Variables a evaluar),
          notch = True (La muesca se dibuja en la mediana del conjunto de datos y se calcula a partir de la distribución de los datos),
           labels=['M', 'B'](Descripción de las variables numéricas en el gráfico),
           widths = .7 (se refiere al ancho relativo de las cajas en el gráfico, En los boxplots, las cajas se dibujan entre el primer y tercer cuartil),
           patch_artist=True(Especificar si se deben rellenar las cajas de los boxplots con color),
           medianprops = dict(linestyle='-', linewidth=2, color='Yellow') (El parámetro "medianprops" acepta un diccionario que especifica las propiedades de la línea de mediana, como el color, el grosor y el estilo de la línea. ),
           boxprops = dict(linestyle='--', linewidth=2, color='Black', facecolor = 'blue', alpha = .4)
          )(El parámetro "boxprops" acepta un diccionario que especifica las propiedades de la caja, como el color, el grosor y el estilo de línea, el parámetro "alpha" se utiliza para ajustar la transparencia de los elementos en un gráfico);

boxplot1 = boxplots['boxes'][0]
boxplot1.set_facecolor('red')
plt.xlabel('diagnosis', fontsize = 20);
plt.ylabel('area_mean', fontsize = 20);
plt.xticks(fontsize = 16);
plt.yticks(fontsize = 16);
```

Ejemplo



Subplots

Concepto

En programación y visualización de gráficos, los subplots se refieren a la capacidad de dividir una figura en varias sub-figuras, cada una de las cuales representa un gráfico o una trama diferente. En otras palabras, los subplots son múltiples gráficos que se muestran juntos en una sola figura.

Casos de uso

Los subplots son útiles cuando se desea comparar diferentes conjuntos de datos o visualizar múltiples variables en un solo gráfico. Por ejemplo, si se tiene un conjunto de datos con múltiples variables, se pueden crear subplots para mostrar cada variable en su propio gráfico individual dentro de una sola figura. Esto hace que sea más fácil ver las diferencias y similitudes entre las variables.

Ejemplos:

```
plt.figure(figsize=(8,4))
# Subplot 1
plt.subplot(1, 2, 1);
plt.plot(df_intercept['feature'].values,
df_intercept['predicted'].values, c = 'r');
plt.scatter(df_intercept['feature'].values,
df_intercept['actual'].values, c= 'k');
plt.title('intercept', fontsize = 12);
# Subplot 2
plt.subplot(1, 2, 2);
plt.plot(df_nointercept['feature'].values,
df_nointercept['predicted'].values, c = 'r');
plt.scatter(df_nointercept['feature'].values,
df_nointercept['actual'].values, c= 'k');
plt.title('no intercept', fontsize = 12);
```

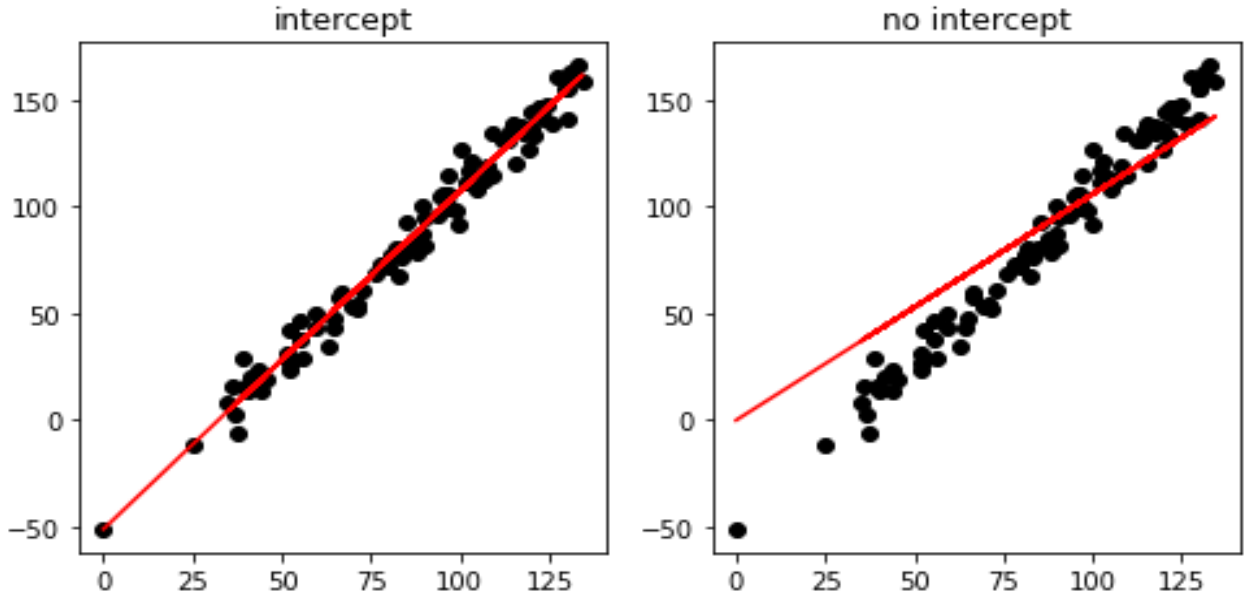


Gráfico de Barra y de Pastel

Concepto

En la librería Matplotlib de Python, se puede crear un gráfico de barras usando la función bar o barh. La función bar se utiliza para crear un gráfico de barras vertical, mientras que la función barh se utiliza para crear un gráfico de barras horizontal.

A continuación, se muestra un ejemplo básico de cómo crear un gráfico de barras vertical utilizando la función bar:

```
import matplotlib.pyplot as plt
# Datos de ejemplo
categorias = ['A', 'B', 'C', 'D']
valores = [10, 20, 30, 15]

# Crear un gráfico de barras vertical
plt.bar(categorias, valores)

# Personalizar el gráfico
plt.title('Ejemplo de gráfico de barras')
plt.xlabel('Categorías')
plt.ylabel('Valores')
plt.show()
```

Ejemplo

Gráfico horizontal **barh()**

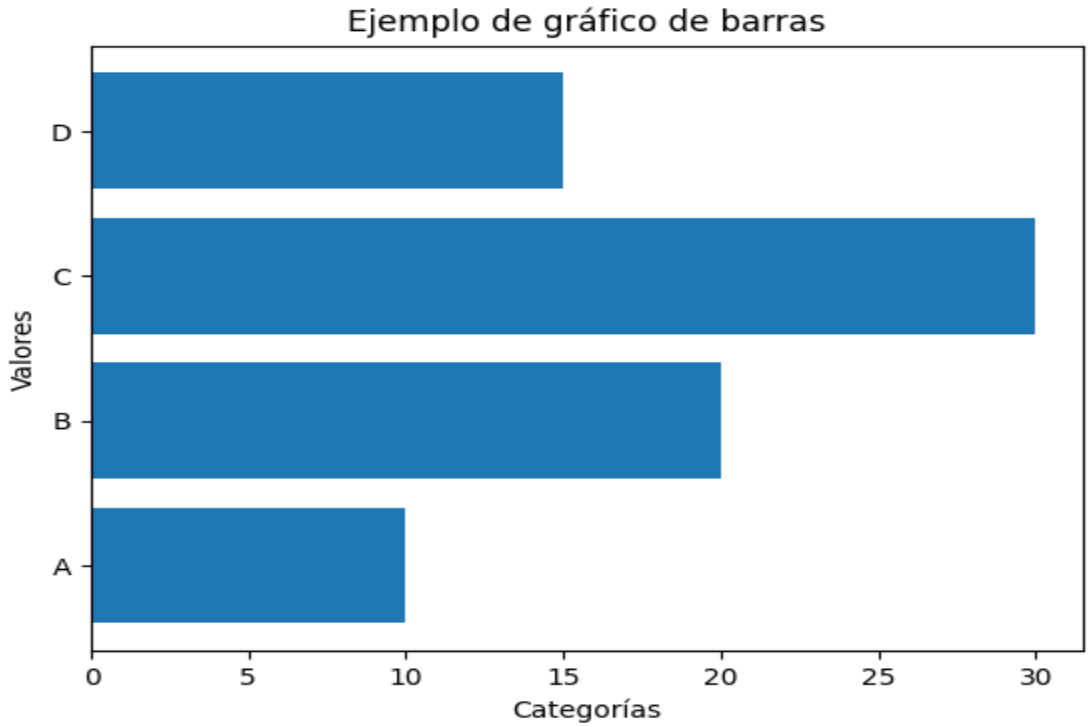
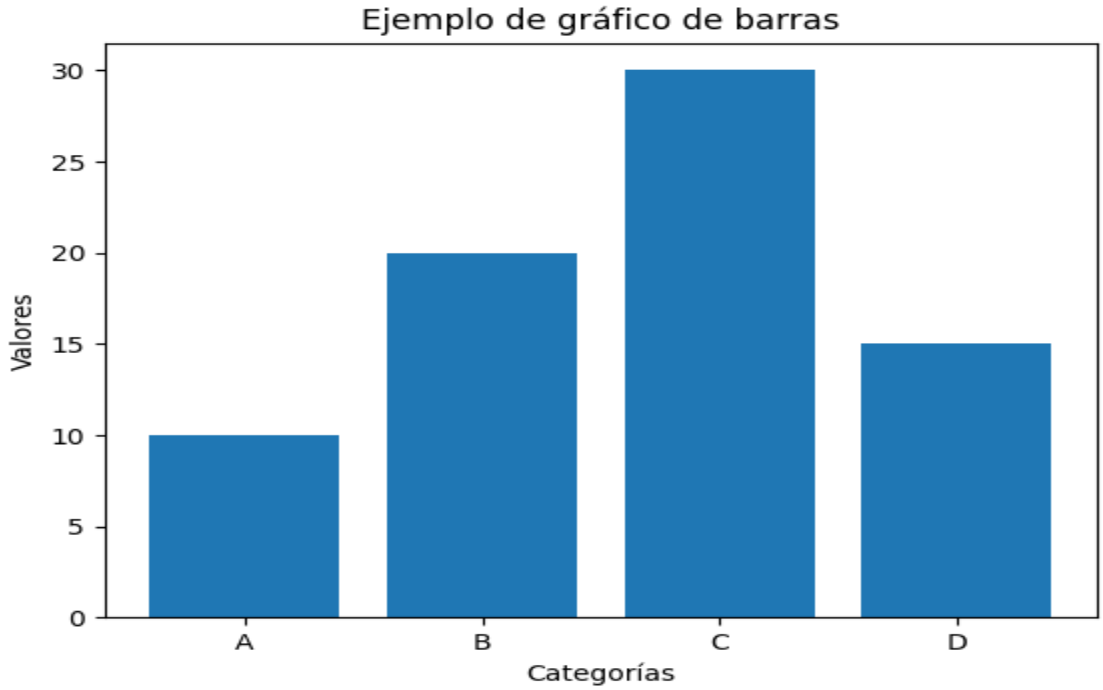


Grafico vertical **bar()**



Envoltorios de Matplotlib para Pandas y Seaborn

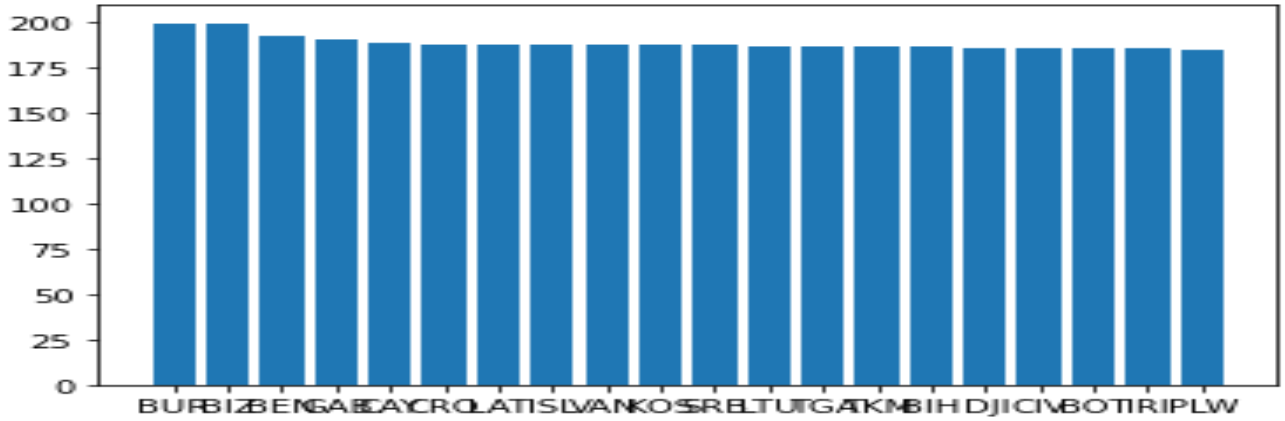
Conceptos

Los envoltorios de Matplotlib se refieren a las bibliotecas y herramientas que se construyen encima de la biblioteca de visualización de datos Matplotlib en Python. Estos envoltorios se utilizan para simplificar el proceso de creación de visualizaciones y gráficos, agregando características adicionales y mejorando la facilidad de uso de Matplotlib.

Seaborn: Seaborn es un envoltorio de Matplotlib que se utiliza para crear gráficos estadísticos complejos y elegantes con solo unas pocas líneas de código. Seaborn se enfoca en visualizaciones de datos de alta calidad y ofrece varios estilos de trama predefinidos que mejoran la apariencia visual de los gráficos. Algunas de las características avanzadas de Seaborn incluyen tramas de distribución, mapas de calor, tramas de regresión, diagramas de violín y tramas de pares.

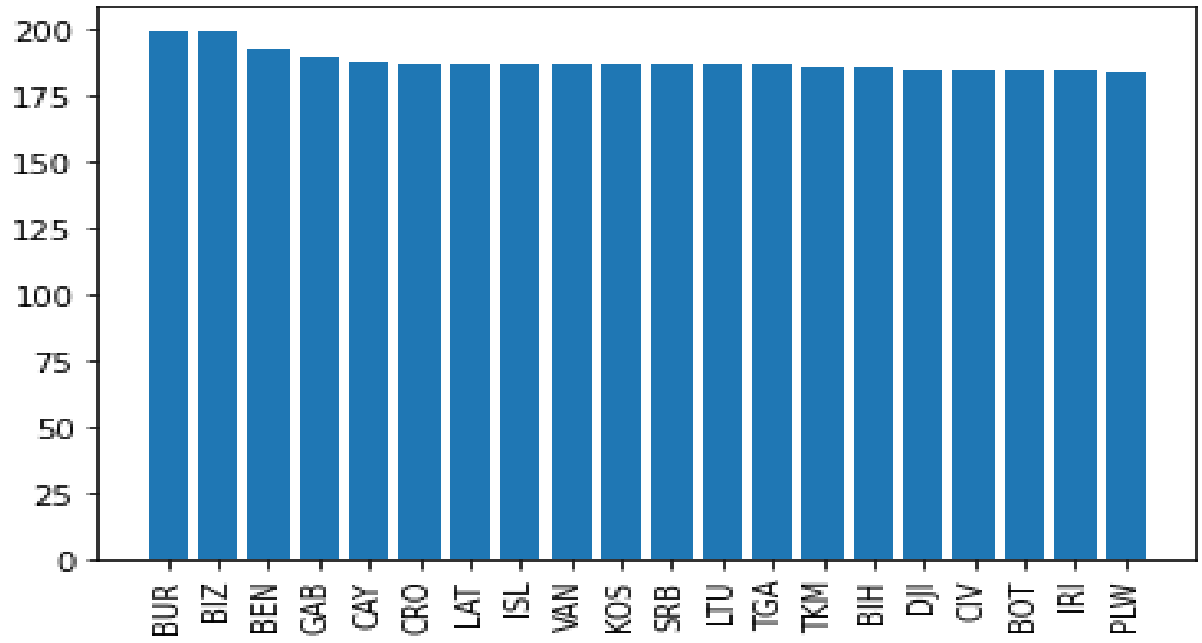
Caso de uso

En este caso veremos como no se distinguen las etiquetas en el eje X lo cual impide que se puedan presentar los datos de una manera efectiva y entendible. Para evitar esto existen los envoltorio de matplotlib en donde veremos un ejemplo más adelante.



Ejemplos

```
plt.bar(top20Height.index, top20Height.values);
plt.xticks(rotation = 90);
```



Estilos

El estilo por defecto no es el más estéticamente agradable.

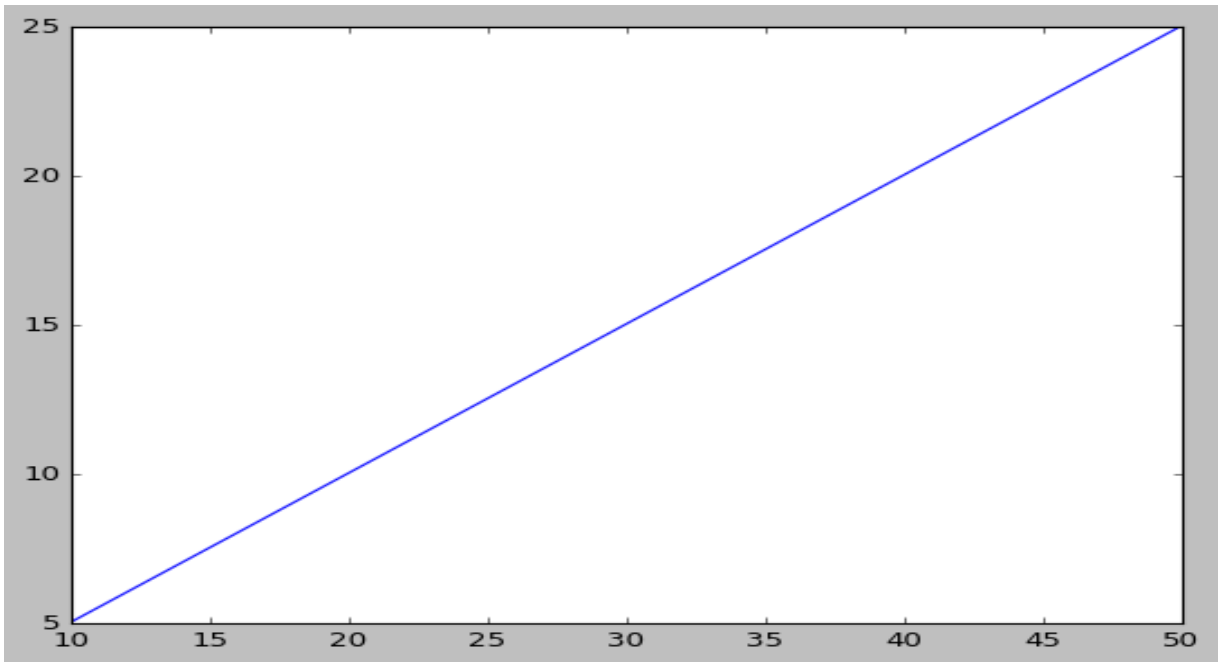
Se puede usar **plt.style**.available para ver diferentes estilos de estética disponibles para las figuras.

```
plt.style.available
```

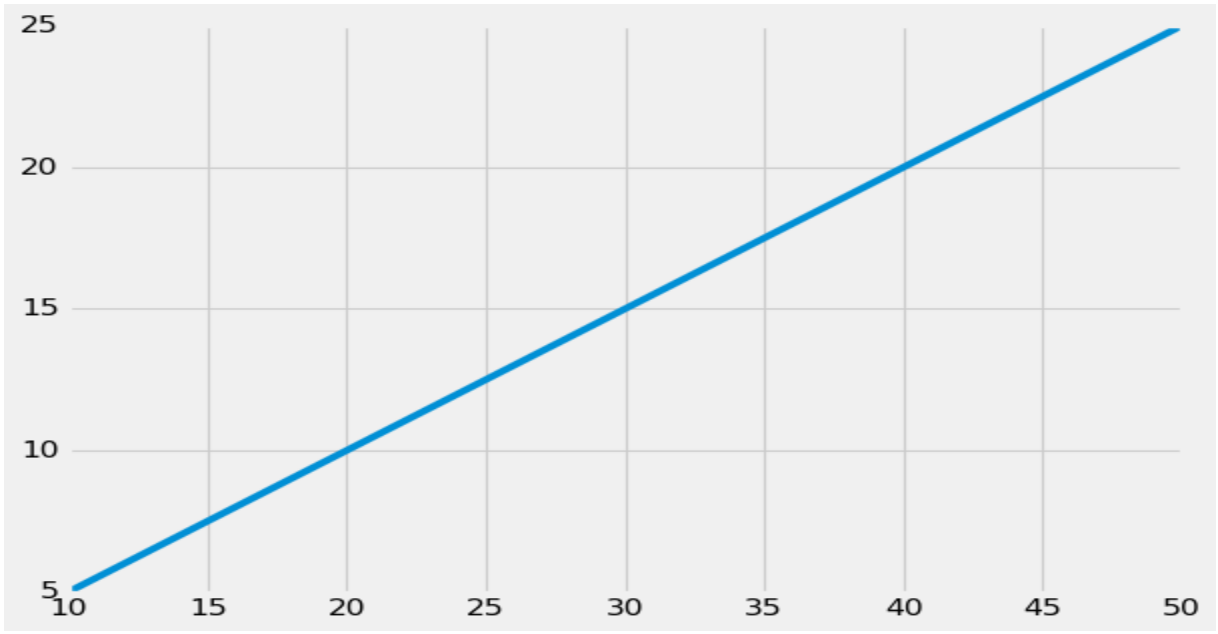
```
['Solarize_Light2', '_classic_test_patch',' bmh', 'classic',
'dark_background', 'fast', 'fivethirtyeight', 'ggplot',
'grayscale', 'seaborn', 'seaborn-bright',
'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette',
'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted',
'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel',
'seaborn-poster', 'seaborn-talk', 'seaborn-ticks',
'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

Ejemplos:

```
plt.style.use('classic')
plt.plot([10, 20, 30, 40, 50], [5, 10, 15, 20, 25])
```

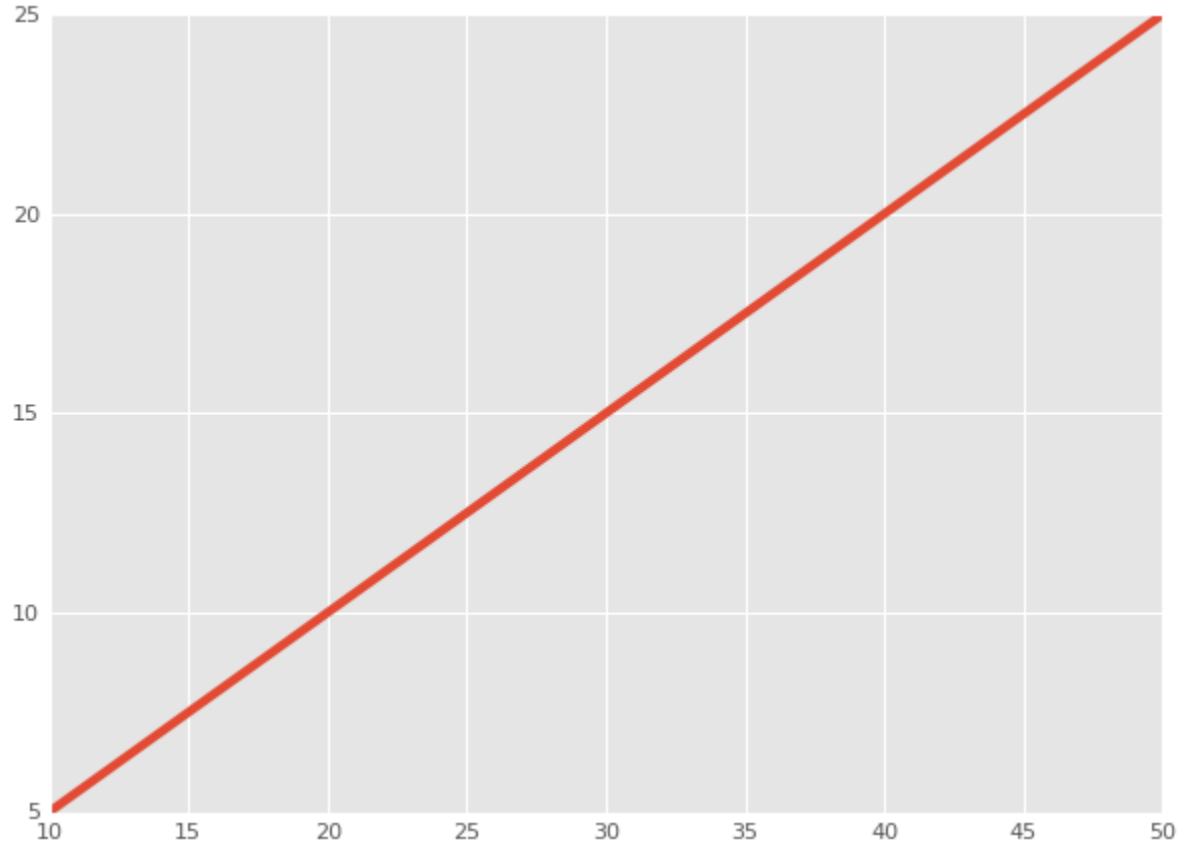



```
plt.style.use('fivethirtyeight')
plt.plot([10, 20, 30, 40, 50], [5, 10, 15, 20, 25])
```




```
plt.style.use('ggplot')
```

```
plt.plot([10, 20, 30, 40, 50], [5, 10, 15, 20, 25])
```



Establecer el tipo de marcador y los colores

Concepto

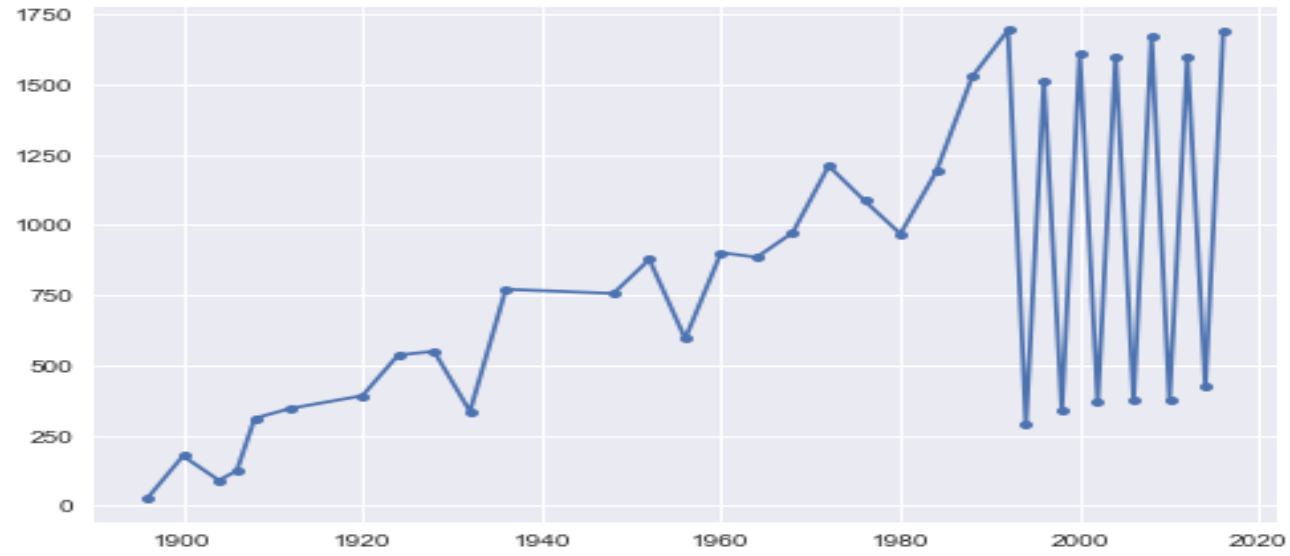
En la visualización de datos, los marcadores son símbolos que se colocan en un gráfico para representar cada punto de datos en una trama. Los marcadores se utilizan comúnmente en gráficos de dispersión, pero también se pueden utilizar en otros tipos de gráficos. A continuación se presentan algunos tipos comunes de marcadores utilizados en la visualización de datos:

Ejemplos

cadena	descripción
'.'	marcador de punto
','	marcador de píxel
'o'	marcador de círculo
'v'	marcador triangle_down (triángulo hacia abajo)
'^'	marcador triangle_up (triángulo hacia arriba)
'<'	marcador triangle_left (triángulo hacia la izquierda)
'>'	marcador triangle_right (triángulo hacia la derecha)
's'	marcador cuadrado

'*'	marcador estrella
'+'	marcador más
'x'	marcador x

```
plt.plot(uniqueYears, numOlympian, marker = '.', markersize = 10)
```



Estilos MATLAB vs sintaxis de objeto

Concepto

Matplotlib, una de las bibliotecas de visualización de datos más populares en Python, ofrece dos formas de crear gráficos: la sintaxis de estilo MATLAB y la sintaxis de objeto.

La sintaxis de estilo MATLAB es una forma más simple y directa de crear gráficos utilizando Matplotlib. Es similar a la forma en que se crearían gráficos en MATLAB, y es muy útil para aquellos que están familiarizados con MATLAB. Esta sintaxis utiliza funciones como "plot" y "scatter" para crear gráficos y toma argumentos como la lista de valores X e Y, así como parámetros para personalizar el gráfico.

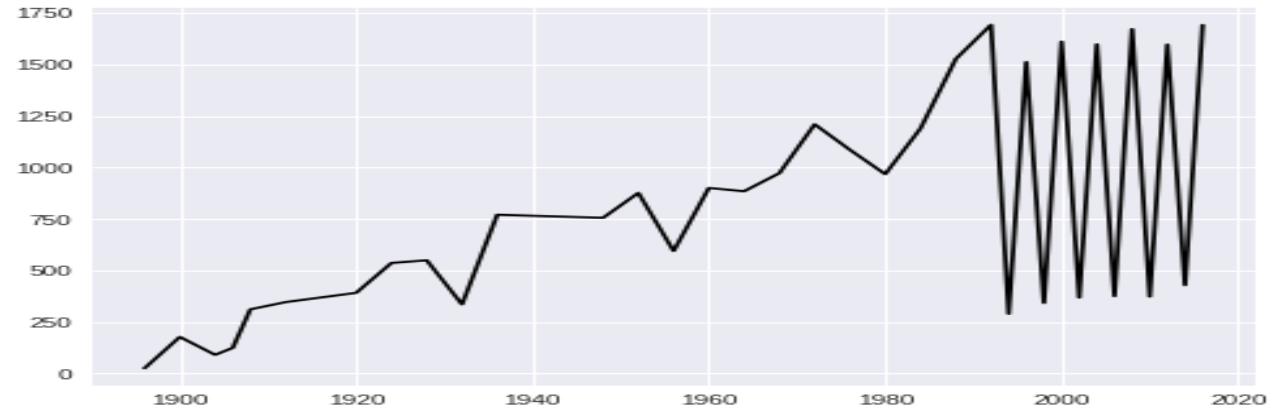
Por otro lado, la sintaxis de objeto es una forma más orientada a objetos de crear gráficos con Matplotlib. En lugar de llamar a funciones para crear gráficos, los gráficos se crean a través de objetos en un sistema jerárquico de objetos. Los gráficos se construyen agregando elementos como ejes, líneas y leyendas a un objeto de la figura principal. Esto da a los usuarios un mayor control sobre la personalización de gráficos y permite la creación de gráficos más complejos.

Si bien ambas sintaxis pueden ser utilizadas para crear gráficos en Matplotlib, la sintaxis de objeto es más poderosa y versátil. La sintaxis de estilo MATLAB es más adecuada para usuarios que desean crear gráficos simples y rápidos sin la necesidad de personalización avanzada.

Ejemplos

Estilo MATLAB

```
plt.style.use('seaborn')
plt.plot(uniqueYears, numOlympian, c= 'k')
```



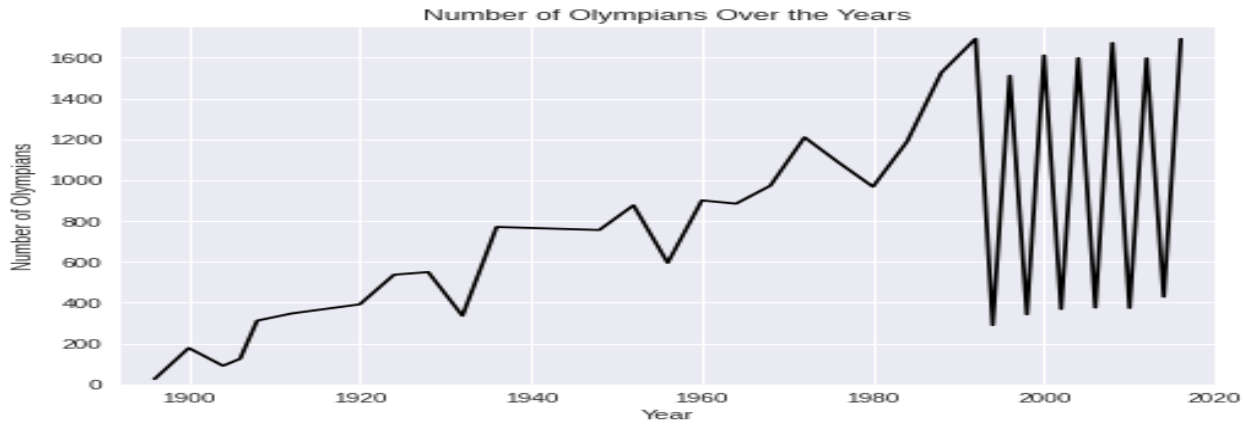
Orientado a objetos

```
fig, axes = plt.subplots(nrows = 1, ncols = 1)
axes.plot(uniqueYears, numOlympian, c= 'k');
```

Combinación

A menudo verán ambos tipos de sintaxis usados juntos. En el caso del código de abajo, los ejes son el código orientado a objetos y los comandos plt.xticks y plt.yticks son estilo MATLAB.

```
fig, axes = plt.subplots(nrows = 1, ncols = 1);
axes.plot(uniqueYears, numOlympian, c= 'k');
plt.xticks(fontsize = 25);
plt.yticks(fontsize = 25);
```



Establecer títulos, etiquetas y límites

Concepto

En la visualización de datos, los títulos, etiquetas y límites son herramientas importantes para mejorar la comprensión de los gráficos. Aquí hay una breve descripción de cómo establecer títulos, etiquetas y límites en Matplotlib:

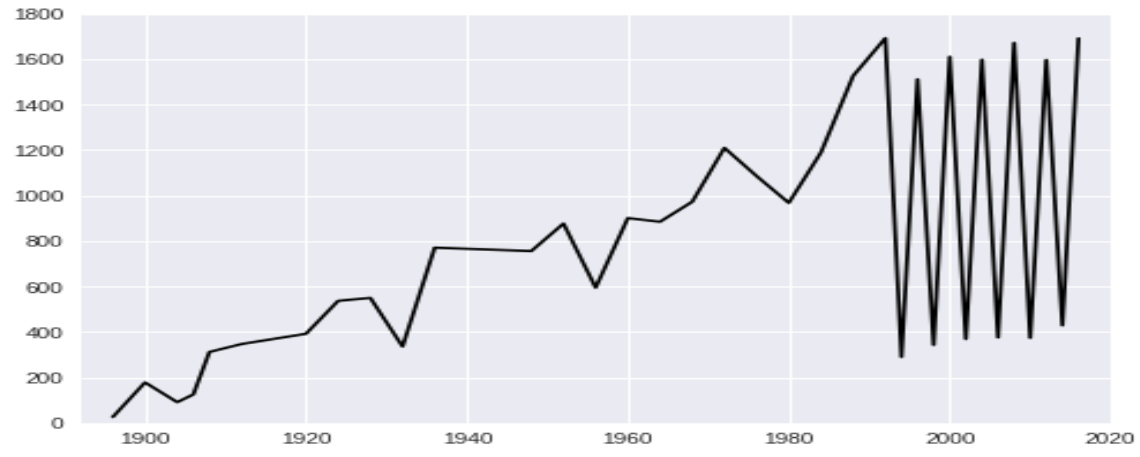
Etiquetas de eje: Las etiquetas de eje se utilizan para proporcionar una descripción de los datos representados en un eje de un gráfico. Puede establecer las etiquetas de los ejes x e y utilizando las funciones xlabel() e ylabel() de Matplotlib

Límites de eje: Los límites de eje se utilizan para establecer los límites mínimos y máximos de los ejes x e y de un gráfico. Puede establecer los límites de los ejes utilizando las funciones xlim() y ylim() de Matplotlib.

Ejemplos

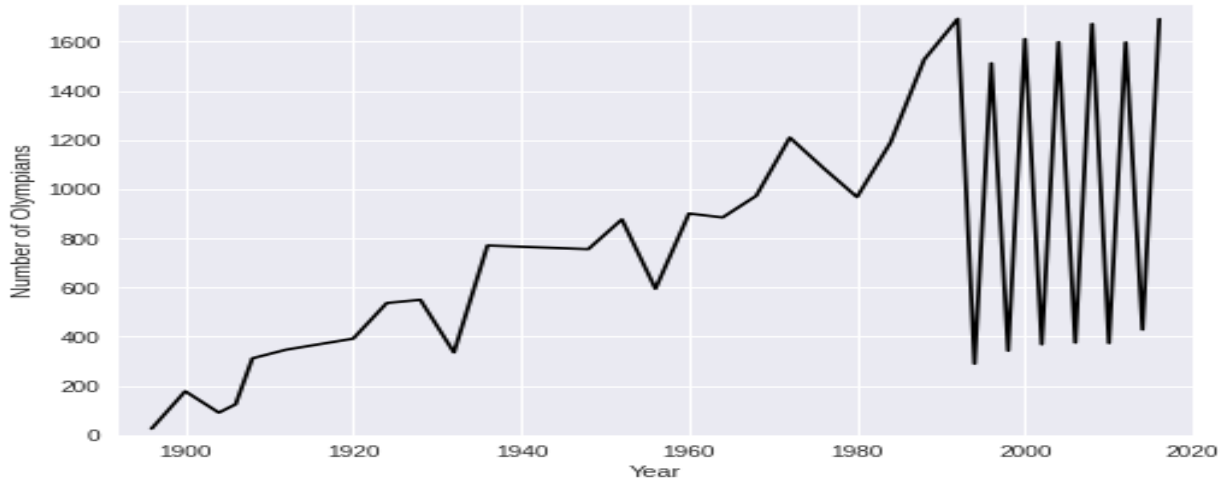
Un ejemplo de etiquetas de eje, para establecer la etiqueta del eje x como "Edad" y la etiqueta del eje y como "Frecuencia", puede escribir plt.xlabel("Edad") y plt.ylabel("Frecuencia").

```
# Set xlim and ylim
plt.plot(uniqueYears, numOlympian, c= 'k')
plt.xlim(left=1892,right=2020)
plt.ylim(bottom=0,top=1800)
```



Ejemplo de límites de eje, para establecer los límites del eje x de 0 a 100 y los límites del eje y de 0 a 50, puede escribir plt.xlim(0, 100) y plt.ylim(0, 50).

```
# Set xlabel and ylabel
plt.plot(uniqueYears, numOlympian, c= 'k')
plt.xlim(left=1892,right=2020)
plt.ylim(bottom=0,top=1750)
plt.xlabel('Year')
plt.ylabel('Number of Olympians')
```



Folium

Concepto

"Folium es una biblioteca de Python que se usa para visualizar datos geoespaciales. Es muy fácil de usar, pero es una biblioteca eficaz. Folium es un envoltorio de Python para Leaflet.js , que es una de las principales bibliotecas JavaScript de código abierto para trazar mapas interactivos".

Es una de las numerosas herramientas disponibles para trazar datos de mapas. Algunas otras herramientas incluyen GeoPandas, Plotly, y Tableau.

Trazar mapas

Folium hace el trazado de mapas simple. Todo lo que necesitan es la latitud y longitud del área que desean generar un mapa.

Ejemplo

Por ejemplo, tracemos un mapa para Denver, Colorado, que tienen unas coordenadas de latitud y longitud de: 39.742043, -104.991531.

```
import folium
map = folium.Map(location = [39.742043, -104.991531])
map
```

¡El resultado es un mapa interactivo en el que se puede hacer zoom y desplazarse! ¡Intenten hacer esto en sus ciudades natales!

También podemos agregar marcadores:

```
folium.Marker(location=[39.849312, -104.673828],popup='Denver
International Airport').add_to(map)
map
```

Hay mucho que pueden hacer con esta biblioteca, en particular los mapas coropléticos.