Curso de Python para Analisis de Datos

Clase 6

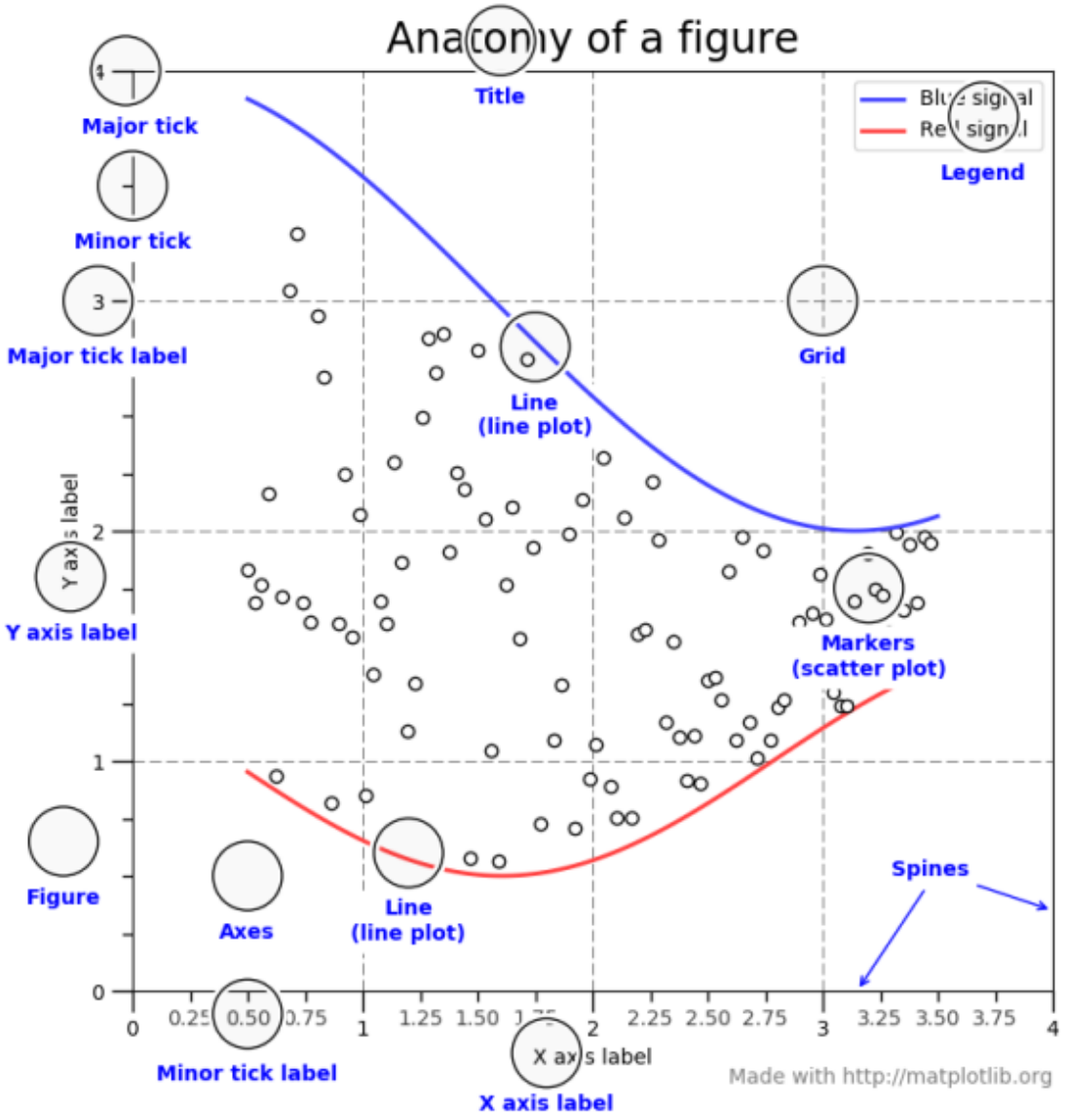
Indice:

Matplotlib

Es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en Python y su extensión matemática NumPy. Se basa en el programa matlab.

`matplotlib` tiene una amplia base de código que puede ser desalentador para muchos usuarios nuevos. Sin embargo, la mayoría de la biblioteca se puede entender con un marco conceptual bastante simple y conocimiento de algunos puntos importantes. Todos los graficos y las funciones de matplotlib siguen las mismas reglas. Por ende una vez que nos familiaricemos con algunas, el resto seran mas faciles.

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Legend
Grid
Line
(line plot)
Y axis label
Markers
(scatter plot)
Figure
Axes
Line
(line plot)
Spines
Minor tick label
X axis label
Made with http://matplotlib.org

Hay diferentes formas de plotear con Matplotlib:

figure(num, figsize, dpi, facecolor, edgecolor, frameon)

num = numeración de la figura, si num = None, las figuras se numeran automáticamente.

figsize = w, h tuplas en pulgadas. Tamaño de la figura

dpi = Resolución de la imagen en puntos por pulgada.

facecolor = Color del rectángulo de la figura.

edgecolor = Color del perímetro de la figura.

frameon = Si es falso, elimina el marco de la figura.

Para crear mas figuras en una misma ventana podemos utilizar el siguiente comando:

subplot(numRows, numCols, plotNum)

numRows = Número de filas

numCols = Número de columnas

plotNum = Número de gráfica

Tambien podemos incluir varias gráficas en una única figura:

plot(x, y, linestyle, linewidth, marker) –>

x = Abcisas.

y = Ordenadas. Tanto x como y pueden ser abcisas tuplas, listas o arrays, pero ambas deben tener el mismo tamaño.

linestyle = color y tipo de dibujar la gráfica. Por ejemplo ‘k- -‘

linewidth = ancho de línea.

marker = Marcador.

Vamos a generar unos datos para empezar a graficar. Utilizamos de `numpy` la funcion `linspace` que nos retorna valores entre los valores especificados. **np.linspace(desde, hasta, cantidad)**

In [ ]:
```
#Generamos los valores del eje x
x = np.linspace(-10, 10, 50)
x
```

In [ ]:
```
x2 = list(range(-10, 10))
```

In [ ]:
```
x2
```

In [ ]:
```
type(x2)
```

In [ ]:
```
type(x)
```

Vemos la diferencia entre `range()` y `linspace()`?

`range` nos retorna una lista mientras que `linspace` nos devuelve un array

Recordemos la fórmula de una recta o funcion lineal:

$$y = a \cdot x + b$$

In [ ]:
```
['hola', 'chau'] * 2
```

In [ ]:
```
# Definimos los parámetros de la recta
a = -2
b = 3
```

In [ ]:
```
# Creamos y a partir de la función
y = a*x + b
y
```

La manera más rápida y fácil de graficar es usar el método `plot` de matplotlib

In [ ]:
```
len(x), len(y)
```

In [ ]:
```
plt.plot(x, y)
```

Generamos nuestro primer gráfico! Ahora vamos a mejorarlo un poco.

In [ ]:
```
# Generamos el mismo gráfico que antes
plt.plot(x, y)
# Ahora le agregamos una grilla
plt.grid(True)
```

In [ ]:
```
# Generamos el mismo gráfico que antes
plt.plot(x, y)
# Ahora le agregamos una grilla, un título y los nombres de los ejes
plt.grid(True)
plt.title('y = ax + b')
plt.xlabel('x')
plt.ylabel('y')
```

Ejercicio: graficar un polinomio de grado 2 ($y = ax^2 + bx + c$) con los coeficientes que ustedes quieran, para los valores de x desde el -5 al 15.

In [ ]:
```
x = np.linspace(-5,15,50)
```

In [ ]:
```
a = 1
b = -2
c = 3
```

In [ ]:
```
y = a*x**2 + b*x + c
```

In [ ]:
```
# Generamos el mismo gráfico que antes
plt.plot(x, y)
# Ahora le agregamos una grilla, un título y los nombres de los ejes
plt.grid(True)
plt.title('y = ax2 + bx + c')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim(5, 7.5)
plt.ylim(10, 50)
```

Ejercicio: graficar la función $tan(\theta)$ para los valores de θ en el intervalo $(-\pi, \pi)$

In [ ]:
```
np.pi
```

In [ ]:
```
tita = np.linspace(-np.pi/2, np.pi/2, 500)
```

In [ ]:
```
np.tan(np.pi/2)
```

In [ ]:
```
tita
```

In [ ]:
```
y = np.tan(tita)
```

In [ ]:
```
y
```

In [ ]:
```
plt.plot(tita, y)
plt.ylim(-50, 50)
```

Matplotlib nos permite superponer gráficos

In [ ]:
```
x = np.linspace(0,np.pi*4, 100)
seno = np.sin(x)
coseno = np.cos(x)
```

In [ ]:
```
plt.plot(x, seno)
plt.plot(x, coseno)
plt.grid(True)
```

Vamos a personalizar un poco más los gráficos. A la función plot, podemos pasarle un argumento de formato (fmt) en el que le especificamos el color del trazo, el tipo de marcador y el tipo de línea.

Podemos elegir de los siguientes colores:

========== ==========

character color

========== ==========

'b' blue

'g' green

'r' red

'c' cyan

'm' magenta

'y' yellow

'k' black

'w' white

========== ==========

Los siguientes marcadores:

========== ==========

character description

========== ==========

'.' point marker

',' pixel marker

'o' circle marker

'v' triangle_down marker

'^' triangle_up marker

'<' triangle_left marker

'>' triangle_right marker

'1' tri_down marker

'2' tri_up marker

'3' tri_left marker

'4' tri_right marker

's' square marker

'p' pentagon marker

'*' star marker

'h' hexagon1 marker

'H' hexagon2 marker

'+' plus marker

'x' x marker

'D' diamond marker

'd' thin_diamond marker

'|' vline marker

'_' hline marker

========== ==========

Y los siguientes tipos de linea:

========== ==========

character description

========== ==========

'-' solid line style

'--' dashed line style

'-.' dash-dot line style

':' dotted line style

========== ==========

Combinando los caracteres para [color][marcador][linea] podemos personalizar cada trazo.

In [ ]:
```
plt.plot(x, seno, 'ro:')
plt.plot(x, coseno, 'g-')
```

In [ ]:
```
x = np.linspace(0, 10, 10)
plt.plot(np.concatenate((-x,x)), np.concatenate(((-x)**2,x**2)), 'mD--')
plt.plot(np.concatenate((-x,x)), np.concatenate(((-x)**3,x**3)), 'cd-')
```

In [ ]:
```
#Podemos agrandar el gráfico usando el parámetro figsize
plt.figure(figsize=(8, 8))
plt.plot(np.concatenate((-x,x)), np.concatenate(((-x)**2,x**2)), 'mD')
plt.plot(np.concatenate((-x,x)), np.concatenate(((-x)**3,x**3)), 'cd')
```

Ejemplo donde se aplican diferentes objetos:

Tipos de líneas, colores, marcadores, leyenda, textos en los ejes, malla y se guarda la figura en Matplolib:

In [ ]:
```
#Seteamos los parametros
a = np.linspace(0,20,50)
b= np.sin(a)
c=plt.plot(a, b, 'c-3', linewidth = 2)
c=plt.plot(a+0.2, b-1, 'r-o', linewidth = 2)

# Objetos de la grafica
plt.xlabel("Tiempo (s)", fontsize = 20)
plt.ylabel(r"$y (\mu m)$", fontsize = 24, color = 'blue')
plt.text(0, 2, "Descripcion de la grafica, donde nosotros podemos explicar o
 comentar", fontsize = 20)
plt.title("velocidad (m/s)", fontsize = 20)
plt.legend( ('Etiqueta1', 'Etiqueta2', 'Etiqueta3'), loc = 'upper left')
plt.grid(True)
plt.savefig('figura3.png', dpi = 300) #guarda la gráfica con 300dpi (puntos p
or pulgada)
plt.show()
```

Ejemplo de subplot

In [ ]:
```
#Seteamos los parametros
a = np.linspace(0,20,50)
b= np.sin(a)
plt.figure()

# plot 1
plt.subplot(2,2,1)
plt.plot(a, b,'r')

# Segunda grafica
plt.subplot(2,2,2)
plt.plot(a+2, b*25,'g')

# Tercera grafica
plt.subplot(2,2,3)
plt.plot(b, a,'b')

# Cuarta grafica
plt.subplot(2,2,4)
plt.plot(a, b,'k')

# Mostramos en pantalla
plt.show()
```

Bar Plot

También podemos generar gráficos de barras como el siguiente:

In [ ]:
```
plt.bar([0,1,2,3], [10, 15, 12, 8])
```

En este caso, tenemos que especificar el eje X y la altura de cada una de las barras. El eje X puede ser una lista de valores o las etiquetas de cada categoría.

In [ ]:
```
plt.bar(['Equipo 1', 'Equipo 2'], [10, 15])
```

Qué pasa si graficamos 2 barras para cada valor?

In [ ]:
```
plt.bar(['Equipo 1', 'Equipo 2'], [8, 5])
plt.bar(['Equipo 1', 'Equipo 2'], [10, 15])
```

Vemos que se superponen los valores y el gráfico no queda bien. Tenemos 2 alternativas para graficar varias barras juntas: -Apliarlas -Agruparlas

In [ ]:
```
# Apilados
plt.bar(['Equipo 1', 'Equipo 2'], [10, 15])
plt.bar(['Equipo 1', 'Equipo 2'], [8, 5], bottom = [10, 15]) # "Levantamos el
segundo gráfico"
```

In [ ]:
```
#Agrupados

fig, ax = plt.subplots(figsize=(8,6))

ancho = 0.3 #Seteamos el ancho de las barras
x = np.array([0,1]) #Seteamos el eje X
etiquetas_x = ['Equipo 1', 'Equipo 2'] #Seteamos las etiquetas
valores_1 = [10, 15] # Seteamos los valores para la primera barra
valores_2 = [8, 5] # Seteamos los valores para la segunda barra

ax.bar(x, height=valores_1, width=ancho)
ax.bar(x+ancho, height=valores_2, width=ancho)
ax.set_xlabel('Equipos')
ax.set_xticks(x+ancho/2)
ax.set_xticklabels(('Equipo 1', 'Equipo 2'))
```

Como vemos, matplotlib es muy poderoso, pero cuando queremos hacer gráficos customizados se vuelve más complejo. Por eso es importante leer la documentación y buscar ejemplos en internet que nos ayuden a hacer lo que queremos graficar.

Historgrama

Es un grafico de la representación de distribuciones de frecuencias, en el que se emplean rectángulos dentro de unas coordenadas.

Variables, Graficos de Barras e Histogramas

A primera vista pueden parecer muy similares, pero si nos fijamos bien existen claras diferencias entre ambos tipos de gráficos, que encierran conceptos totalmente diferentes.

Ya sabemos que hay distintos `tipos de variables`. Por un lado están las `cuantitativas`, que pueden ser continuas o discretas. Las `continuas` son aquellas que pueden tomar un valor cualquiera dentro de un intervalo, como ocurre con el peso o la presión arterial (en la práctica pueden limitarse los valores posibles debido a la precisión de los aparatos de medida, pero en la teoría podemos encontrar cualquier valor de peso entre el mínimo y máximo de una distribución). Las `discretas` son las que solo pueden adoptar ciertos valores dentro de un conjunto como, por ejemplo, el número de hijos o el número de episodios de isquemia coronaria.

Por otra parte están las variables `cualitativas`, que representan atributos o categorías de la variable. Cuando las variable no incluye ningún sentido de orden, se dice que es cualitativa `nominal`, mientras que si se puede establecer cierto orden entre las categorías diríamos que es cualitativa `ordinal`. Por ejemplo, la variable fumador sería cualitativa nominal si tiene dos posibilidades: sí o no. Sin embargo, si la definimos como ocasional, poco fumador, moderado o muy fumador, ya existe cierta jerarquía y hablamos de variable cualitativa ordinal.

Pues bien, el `diagrama de barras` sirve para representar las variables cualitativas ordinales. En el eje horizontal se representan las diferentes categorías y sobre él se levantan unas columnas o barras cuya altura es proporcional a la frecuencia de cada categoría. También podríamos utilizar este tipo de gráfico para representar variables cuantitativas discretas, pero lo que no es correcto hacer es usarlo para las variables cualitativas nominales.

El gran mérito del diagrama de barras es expresar la magnitud de las diferencias entre las categorías de la variable. Pero ahí está precisamente, su punto débil, ya que son fácilmente manipulables si modificamos los ejes.

Cambiando de tema, el `histograma` es un gráfico con un significado mucho más profundo. Un histograma representa una distribución de frecuencias que se utiliza (o debe) para representar la frecuencia de las variables cuantitativas continuas. Aquí no es la altura, sino el área de la barra lo que es proporcional a la frecuencia de ese intervalo, y está en relación con la probabilidad con la que cada intervalo puede presentarse. Las columnas, a diferencia del diagrama de barras, están todas juntas y el punto medio es el que da el nombre al intervalo. Los intervalos no tienen por qué ser todos iguales (aunque es lo más habitual), pero siempre tendrán un área mayor aquellos intervalos con mayor frecuencia.

Existe, además, otra diferencia muy importante entre el diagrama de barras y el histograma. En el primero solo se representan los valores de las variables que hemos observado al hacer el estudio. Sin embargo, el histograma va mucho más allá, ya que representa todos los valores posibles que existen dentro de los intervalos, aunque no hayamos observado ninguno de forma directa. Permite así calcular la probabilidad de que se represente cualquier valor de la distribución, lo que es de gran importancia si queremos hacer inferencia y estimar valores de la población a partir de los resultados de nuestra muestra.

In [ ]:
```
data = np.random.randn(1000)
```

In [ ]:
```
nbins=10
n, bins, _ = plt.hist(data, bins='auto')
```

In [ ]:
```
n
```

In [ ]:
```
plt.bar(x=list(range(len(n))), height=n)
```

Otros ejemplos de graficos

Scatter Plot

In [ ]:
```
n = 1000
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)
T = np.arctan2(Y,X)
plt.figure(figsize=(8,6))

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.scatter(X,Y, s=75, c=T, alpha=0.5, edgecolor='black'  )

limite = 2.5
plt.xlim(-limite, limite), plt.xticks([])
plt.ylim(-limite, limite), plt.yticks([])

plt.show()
```

Pie Plot

In [ ]:
```
n = 7
Z = np.ones(n)
Z[-1] *= 2

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.pie(Z, explode=Z* .07, colors='rw',
        wedgeprops={"linewidth": 2, "edgecolor": "black"})
plt.gca().set_aspect('equal')
plt.show()
```

Cambios en los ejes

Graficos mas avanzados

In [ ]:
```
x = np.linspace(-np.pi, np.pi, 256, endpoint=True)
S, C = np.sin(x), np.cos(x)
```

In [ ]:
```
# Graficos de seno y coseno de pi
plt.plot(x, S, color= "black", linewidth = 1.2, linestyle = "-.", label="Sen
o")
plt.plot(x, C, color = "red", linewidth = 0.9, linestyle = "--", label= "Cose
no")

# Limite de los graficos
plt.xlim(S.min()*1.2, S.max()*1.2)
plt.ylim(S.min()*1.1, S.max()*1.1)

plt.grid(True)

#Modificar inscripciones y marcas de los ejes
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$-0$', r'$+\pi/2$', r'$+\pi$', ])
plt.yticks(np.linspace(-1.2, 1.2, 3, endpoint=True),
          ['menos uno', '', '1'])


#Modificar ejes (al centro del grafico, color y borrar linea)
ax = plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data', 0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data', 0))

#LEYENDA DEL GRAFICO
plt.legend(loc="upper left")


# MARCAR UNA LINEA EN EL GRAFICO DEL SENO (LINEA)
x0 = 2*np.pi/3
plt.plot([x0,x0], [0, np.sin(x0)], color="blue", linewidth=2.5)

# Marca un punto en el grafico

plt.scatter([x0, ], [np.sin(x0), ], 50, color="green")

# Texto para la marca del grafico y flecha
plt.annotate("punto del seno",
             xy = (x0, np.sin(x0)), xycoords = "data",
             xytext = (+10,+10), textcoords = 'offset points',
             fontsize= 16,
             arrowprops = dict(arrowstyle = "->"))


#COSENO
x0 = 2*np.pi/3
plt.plot([x0,x0], [0, np.cos(x0)], color="green", linewidth=2.5)
plt.scatter([x0, ], [np.cos(x0), ], 50, color="purple")
plt.annotate("punto del coseno",
             xy = (x0, np.cos(x0)), xycoords = "data",
             xytext = (+20,+10), textcoords = 'offset points',
             fontsize= 16,
             arrowprops = dict(arrowstyle = "->"))


#Modificar las etiquetas de los ejes
for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(12)
```

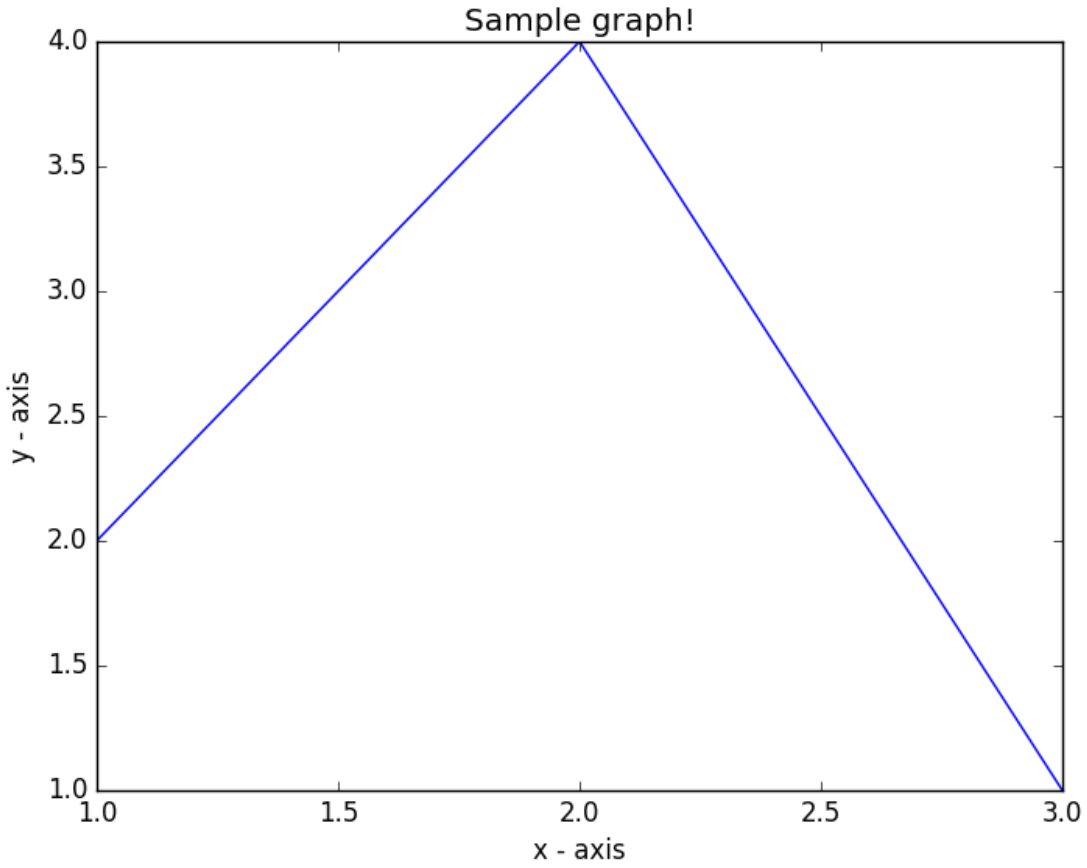
```
    label.set_bbox(dict(facecolor = "yellow", edgecolor= "None",  alpha=0.8))


plt.show()
```

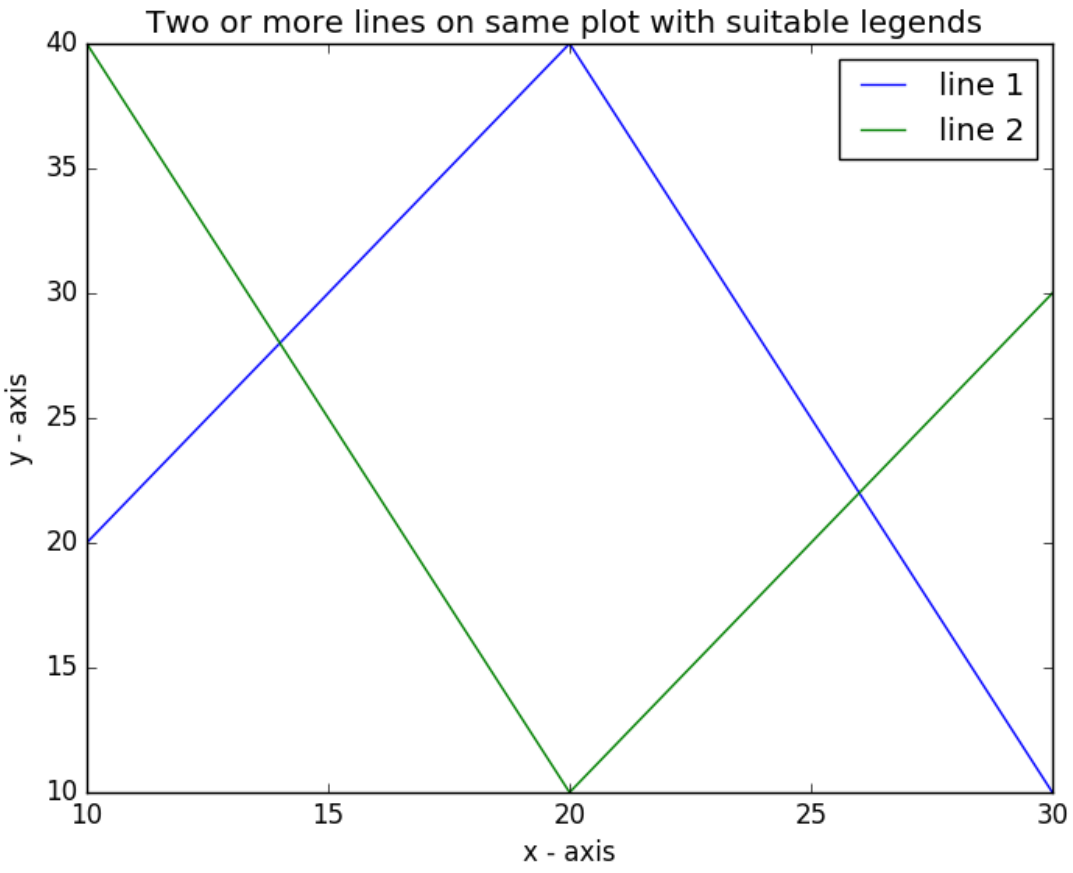
Ejercicios

- Crea un grafico con una recta con etiquetas en los ejes (x, y) y con titulo.

- Crea un grafico igual al ejemplo de la imagen:



- Crea un grafico igual al ejemplo de la imagen:



- Crea un grafico igual al ejemplo de la imagen: