Librería matplotlib

Antonio Espín Herranz

Contenidos

- Formas de trabajar con matplotlib
- Estructura del gráfico
- Componentes
- Tipos de gráficos
- Módulo pyplot
- Objetos de la librería

Trabajar con matplotlib

- **Modo pylab**
  - Es un modo interactivo similar a como trabaja mathlab. Se suele utilizar una consola mejorada de Python → Ipython
  - Se importa el modulo : from pylab import *

- Al utilizar matplotlib se puede utilizar a partir del **módulo pyplot** utiliza funciones y comandos similares a MATLAB.
  - De más alto nivel sin tener que trabajar con los objetos de bajos nivel de la librería.
  - Es más cómodo.

- A través de **los objetos de la librería**. En este caso más complejo, hay que crear y hacer referencia a estos objetos.

pylab

- Desde una consola de Python:
  - Entrando desde el entorno virtual de conda
  - Teclear Python

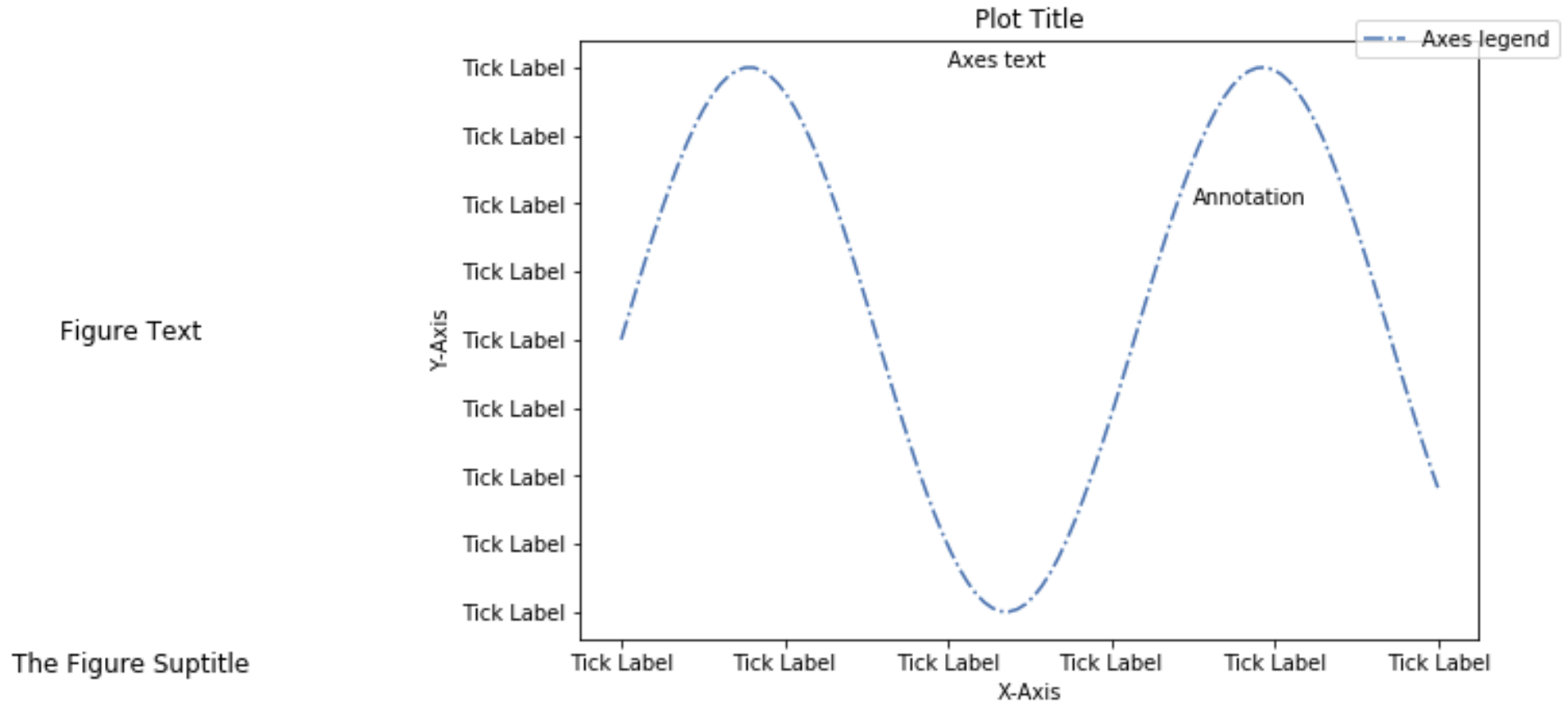
```
>>>from pylab import *
>>>x = arange(10.)
>>>plot(x)
>>>show()
```

pylab con ipython

- **ipython** consola de Python mejorada.
- [1] from pylab import *
- [2] x = arange(10.)
- [3] **ion() → activar modo interactivo, solo en ipython**
- [4] plot(x) → debería dibujar la gráfica sin llamar a **show**()

- La figura se puede limpiar llamando a la función: **clf**()

Estructura del gráfico



Componentes

- **Figure**:
  - Ventana o página general que se dibuja todo.
  - El componente de nivel superior.
  - Puede haber múltiples figuras independientes.
  - Puede tener un título.
  - Se puede agregar una leyenda.
- **Axes**:
  - A la figura se le añaden ejes.
  - Son las áreas donde se representas las gráficas.
  - Según el tipo de gráfico deseado se utilizarán funciones: plot(), scatter(), etc.

- Todos los métodos del objeto Axes existen como una función en el módulo pyplot

Componentes

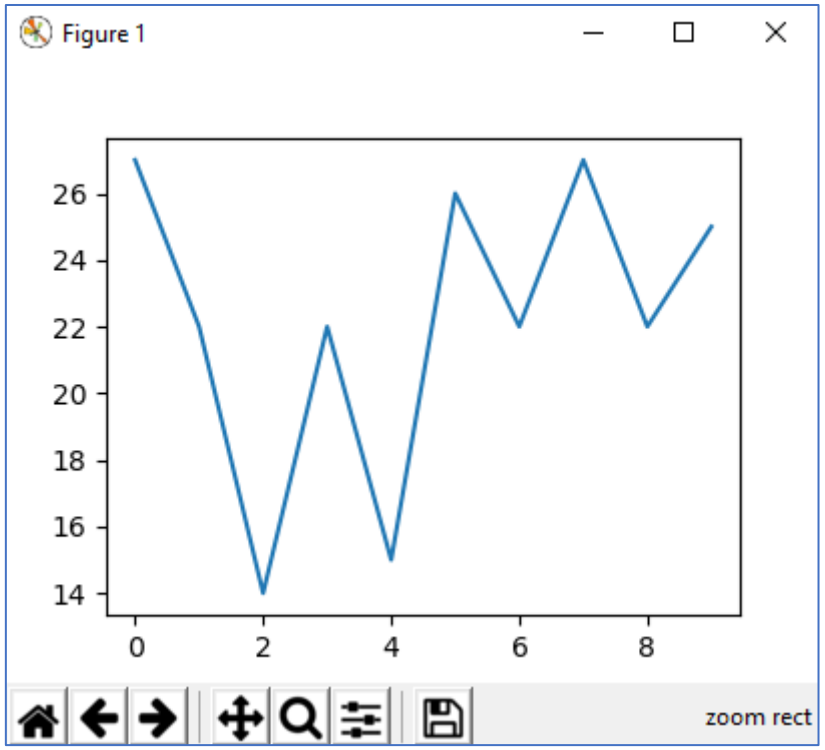
- Cada eje tiene propiedades x-axis / y-axis que contiene **ticklines** y **ticklabels**.

- Las etiquetas de los ejes, título, leyendas, escalas de los ejes y la cuadrícula se pueden personalizar.

Trabajar con matplotlib

- El gráfico se representa en una ventana como esta:

- Con la **barra de la parte inferior** se puede interactuar con el gráfico.
  - Volver a la vista inicial (por ej. Si hemos hecho zoom)
  - Desplazar
  - Zoom
  - Configurar subgráficos
  - Guardar el gráfico



Pasos para crear el gráfico

- Obtener los datos eje X, Y
- Situar el gráfico, podemos tener más de un gráfico en la misma ventana.
- El tipo de gráfico se elije por medio de una función: plot, bar, etc.

Varios gráficos en la misma ventana

- Esta librería permite situar varios gráficos dentro de la misma ventana. Se realiza con un sistema de cuadrícula.

- La sintaxis:
  - pyplot.subplot2grid(numfilas, numCols, (a,b), rowspan=alto, colspan=ancho)
  - Los dos primeros parámetros implican el tamaño de la cuadrícula, es una tupla.
  - (a,b) donde se sitúa el gráfico. Coordenadas, empiezan en 0
  - rowspan=cuantas filas ocupa el gráfico.
  - colspan = cuantas columnas ocupa el gráfico.

- Primero se indica la cuadrícula y luego se sitúa el gráfico.
  - Repetir el proceso para cada uno de los gráficos.

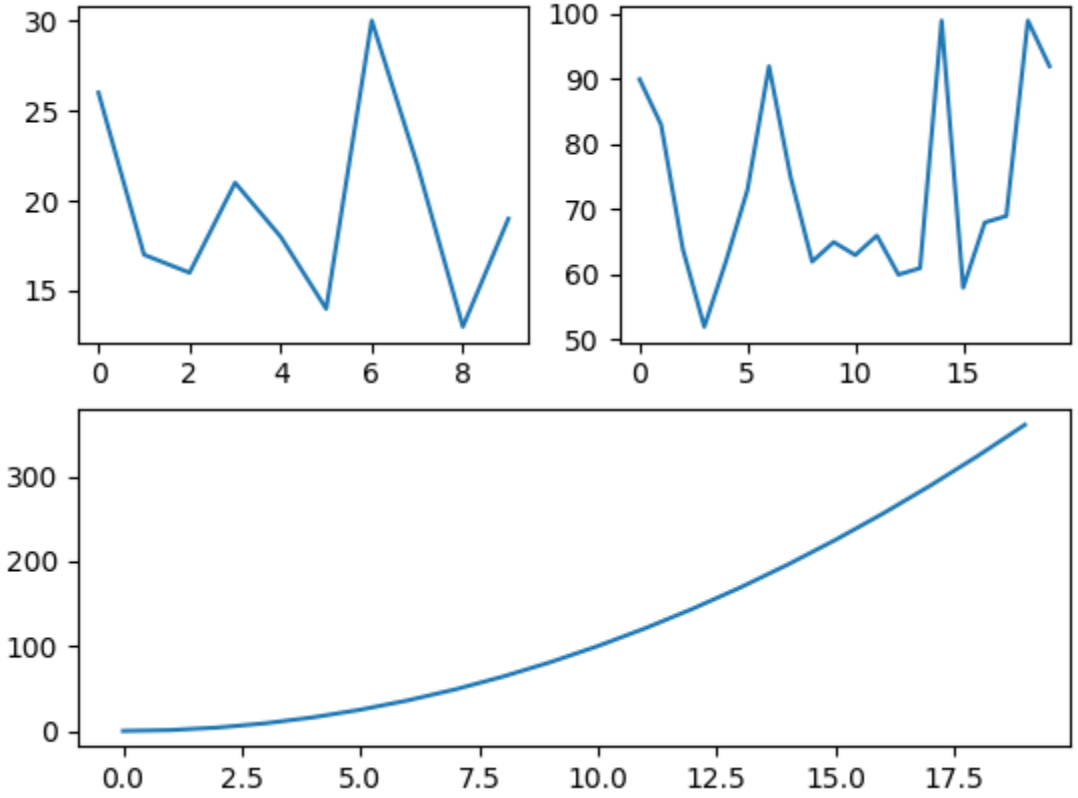
Ejemplo

```
plt.subplot2grid((2,2),(0,0))
plt.plot(x1, y1)

plt.subplot2grid((2,2),(0,1))
plt.plot(x2, y2)

plt.subplot2grid((2,2),(1,0), colspan=2)
plt.plot(x3, y3)

plt.show()
```



Varias gráficas en el mismo gráfico

- También podemos situar varias gráficas dentro del mismo gráfico con el objetivo de comparar o superponer gráficas.

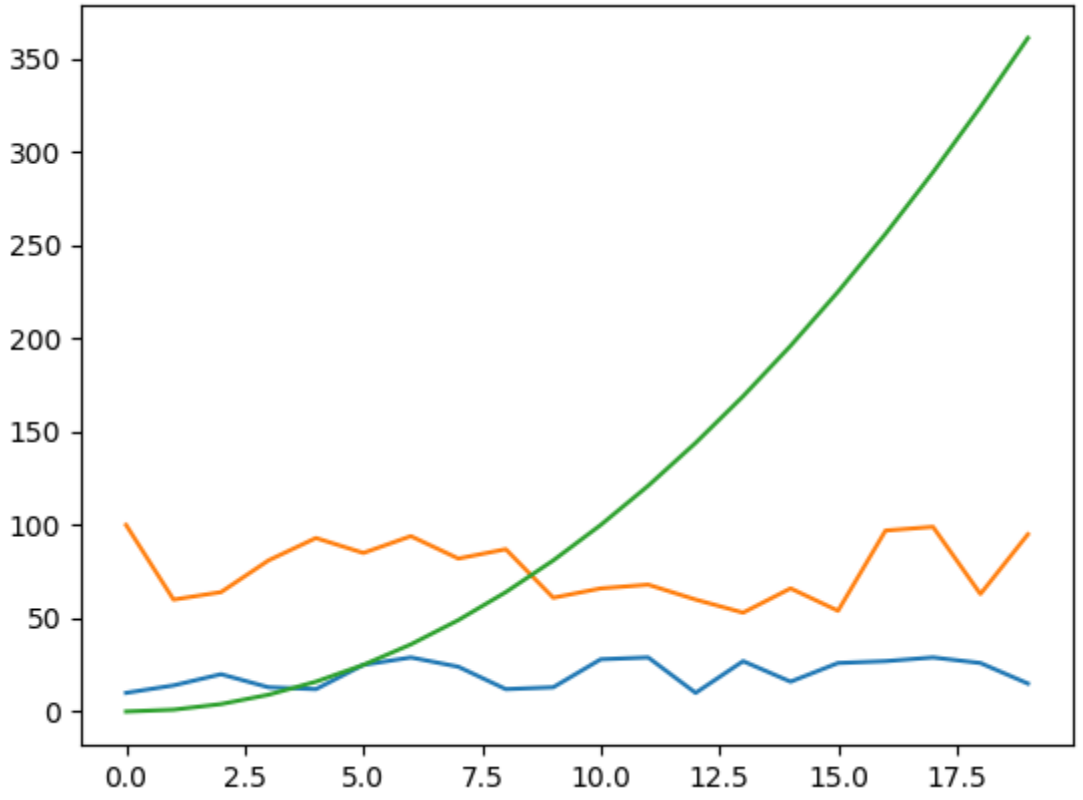
- Consiste en realizar llamadas al método plot (según el gráfico) con el mismo eje X (que será compartido por todos los gráficos).

Ejemplo

- Mismo eje X con distintas gráficas.

```
plt.plot(x, y1)
plt.plot(x, y2)
plt.plot(x, y3)

plt.show()
```



Títulos y etiquetas

- Se puede configurar el título general de la ventana y cada gráfica puede tener sus títulos y etiquetas para los ejes

plt.suptitle("Gráficas comparativas")

plt.subplot2grid((1,2),(0,0))

plt.title("Gráfica1")

plt.xlabel("x gráfica1")

plt.ylabel("y gráfica1")

plt.plot(x, y)

plt.subplot2grid((1,2),(0,1))

plt.title("Gráfica2")

plt.xlabel("x gráfica2")

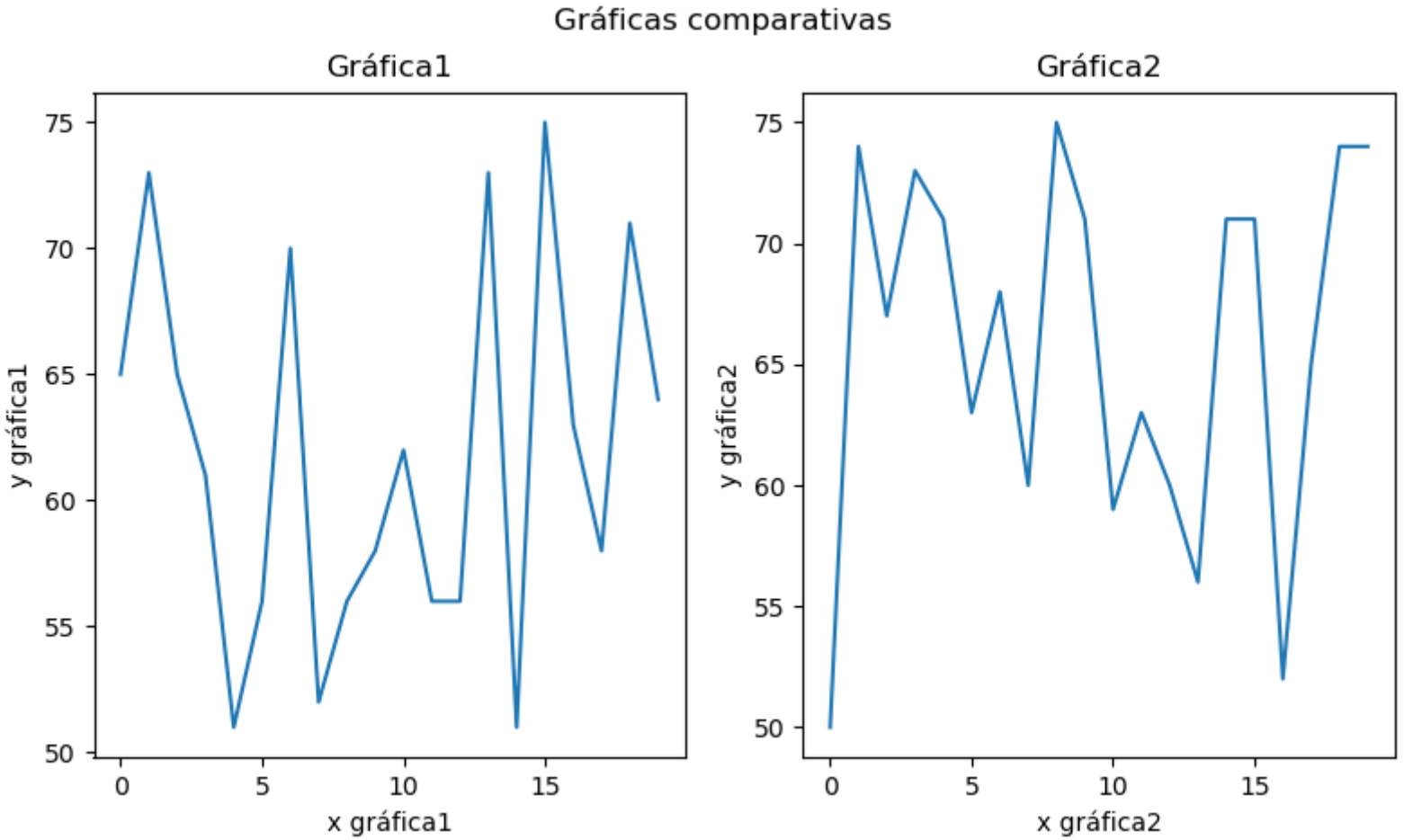
plt.ylabel("y gráfica2")

plt.plot(x, z)

plt.show()

- En **plt.title** se puede añadir otro parámetro: **fontsi**

Con el tamaño de la fuente.



Cuadrícula, leyenda, texto y anotaciones

- Disponemos de las siguientes funciones para modificar estas regiones del gráfico: **grid**(), **legend**(), **text**() y **annotate**()

- Se encuentran en el módulo pyplot:

  - **plt.grid(True, ls='-', color='0.5')**
    - ls es el tipo de línea, en este caso es continua,
    - El color 0 es negro y 1 es blanco.

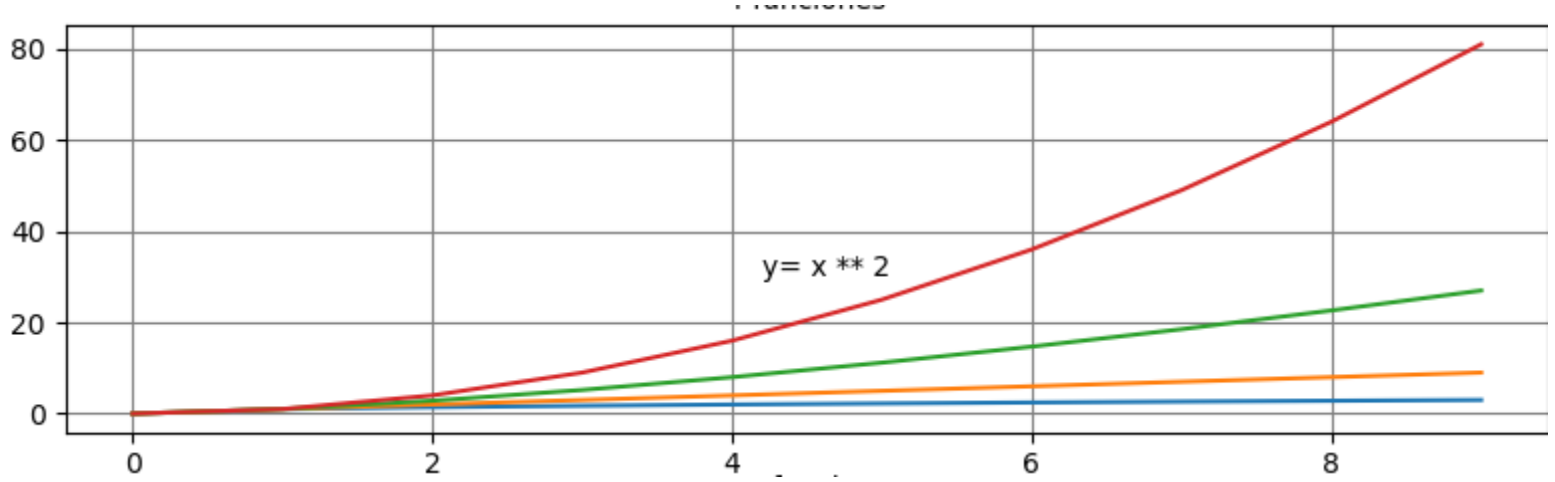
  - **plt.text(x,y,"texto")**
    - Se indican las coordenadas dentro del gráfico donde se incrusta el texto.

  - **plt.leyend()**
    - Coloca de forma automática la leyenda del gráfico.

  - **plt.annotate(texto, xytext, ha, va, arrowprops)**
    - **Texto**: Texto a introducir en el gráfico
    - **xytext**: Coordenadas (es una tupla).
    - **ha**: Alineación horizontal
    - **va**: Alineación vertical
    - **arrowprops**: Un dict con las propiedades de la flecha: Como propiedades del dict podemos utilizar: arrowstyle, facecolor, color,
    - Para crear anotaciones dentro del gráfico.
    - Se dibujan flechas de un texto hacía la gráfica (por ejemplo, la función que estamos representando).

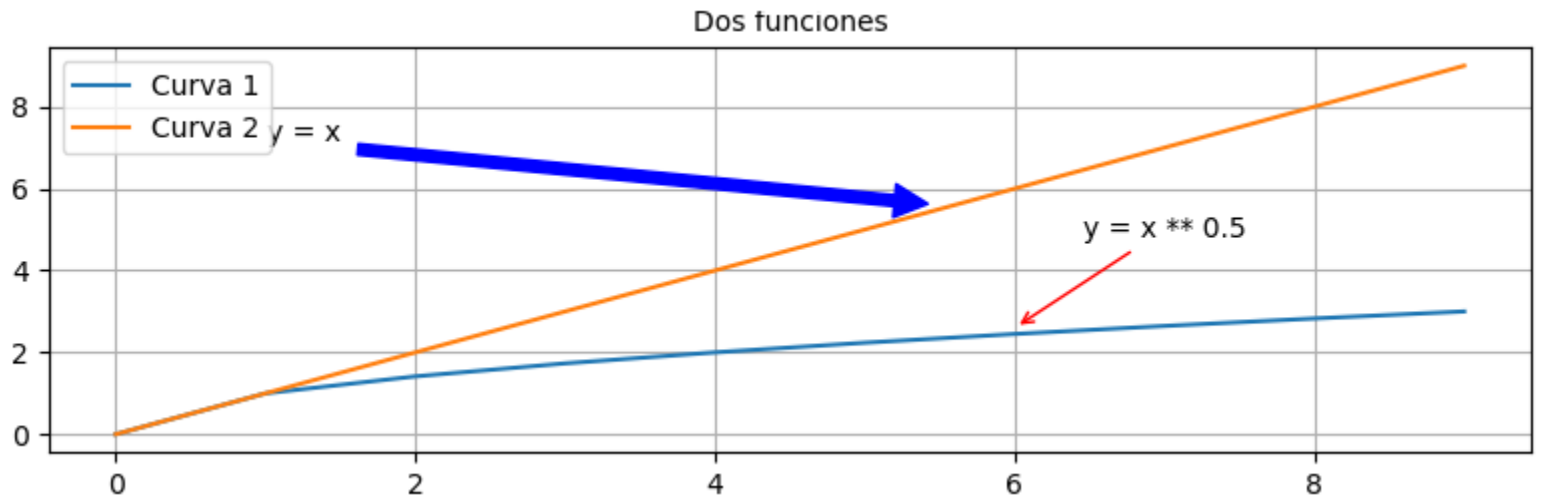
Ejemplo

- plt.suptitle("Ejemplo de funciones simultáneas", fontsize=24)
- plt.subplot2grid((2,2),(0,0), colspan=2)
- plt.title("4 funciones", fontsize=10)
- plt.plot(X, Y1)
- plt.plot(X, Y2)
- plt.plot(X, Y3)
- plt.plot(X, Y4)
- **plt.grid(True, ls='-', color='0.5')**
- **plt.text(4.2,30,"y= x ** 2")**



Ejemplo 2

- plt.subplot2grid((2,2),(1,0), colspan=2)
- plt.title("Dos funciones", fontsize=10)
- plt.plot(X, Y1, label='Curva 1')
- plt.plot(X, Y2, label='Curva 2')
- plt.grid()
- plt.legend()
- plt.annotate("y = x", xytext=(1.5, 7), xy=(5.5, 5.6), ha='right',va='bottom',arrowprops={'color':'blue','shrink':0.02})
- plt.annotate("y = x ** 0.5", xytext=(7, 5), xy=(6, 2.6), ha='center',va='center',arrowprops={'arrowstyle':'->','color':'red'})
- plt.show()



Tipos de gráficos: puntos

- Recopilar los valores del eje X / Y

- Se puede establecer los límites en X / Y
  - Con las funciones: plt.xlim(0, 11) / plt.ylim(0,100)
  - Esta operación se puede realizar en todos los gráficos, afecta al rango de valores de los ejes.

  - plt.**scatter**(eje_x, eje_y)
  - plt.show()

Tipos de gráficos: Barras

- Las barras se pueden crear horizontales y verticales
- plt.**bar**(eje_X, eje_Y) → las barras se pintan verticales
- Se pueden especificar otros parámetros como width, color, align.

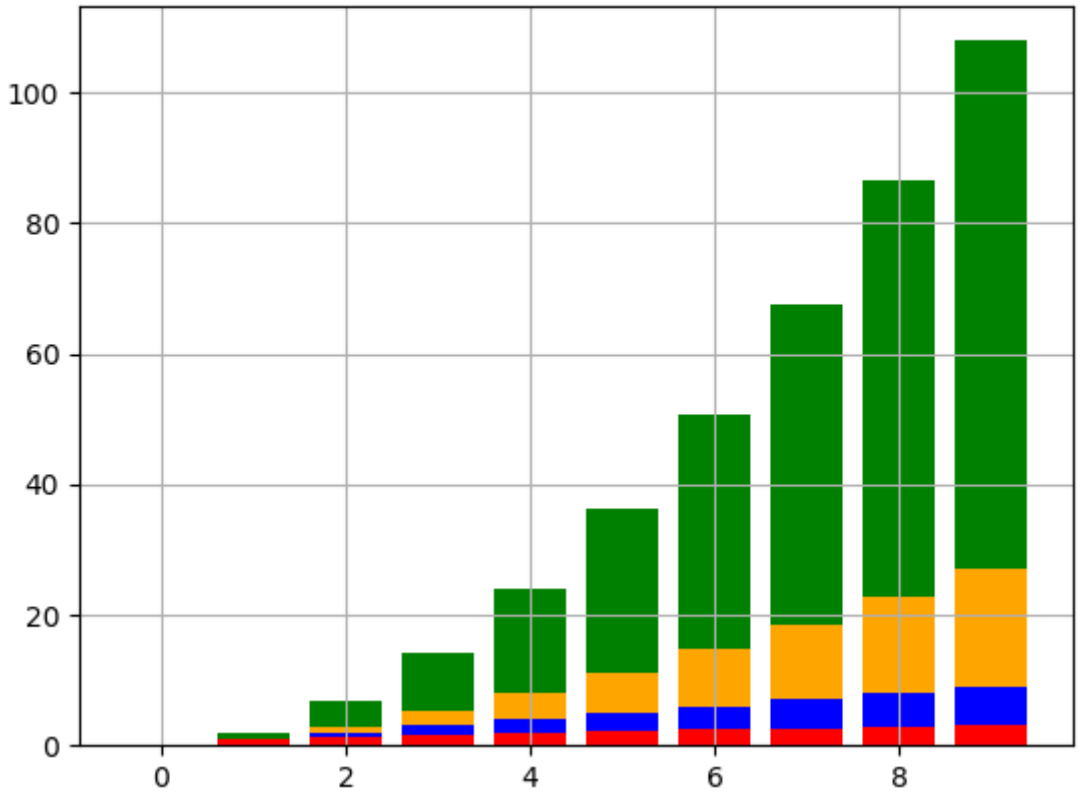
- plt.**barh**(eje_x, eje_Y) → las barras se pintan horizontales.
- Se pueden especificar otros parámetros como height

- Al igual que ocurre con los gráficos de las líneas (plot) se pueden especificar varios ejes Y y un solo eje X.
  - Pintará unas barras encima de otras.

Barras apiladas

- plt.bar(X, Y1, color='red')
- plt.bar(X, Y2, color='blue', **bottom**=Y1)
- plt.bar(X, Y3, color='orange', **bottom**=Y2)
- plt.bar(X, Y4, color='green', **bottom**=Y3)
- plt.show()

- Con **bottom** referenciamos con respecto a que gráfica se apila.



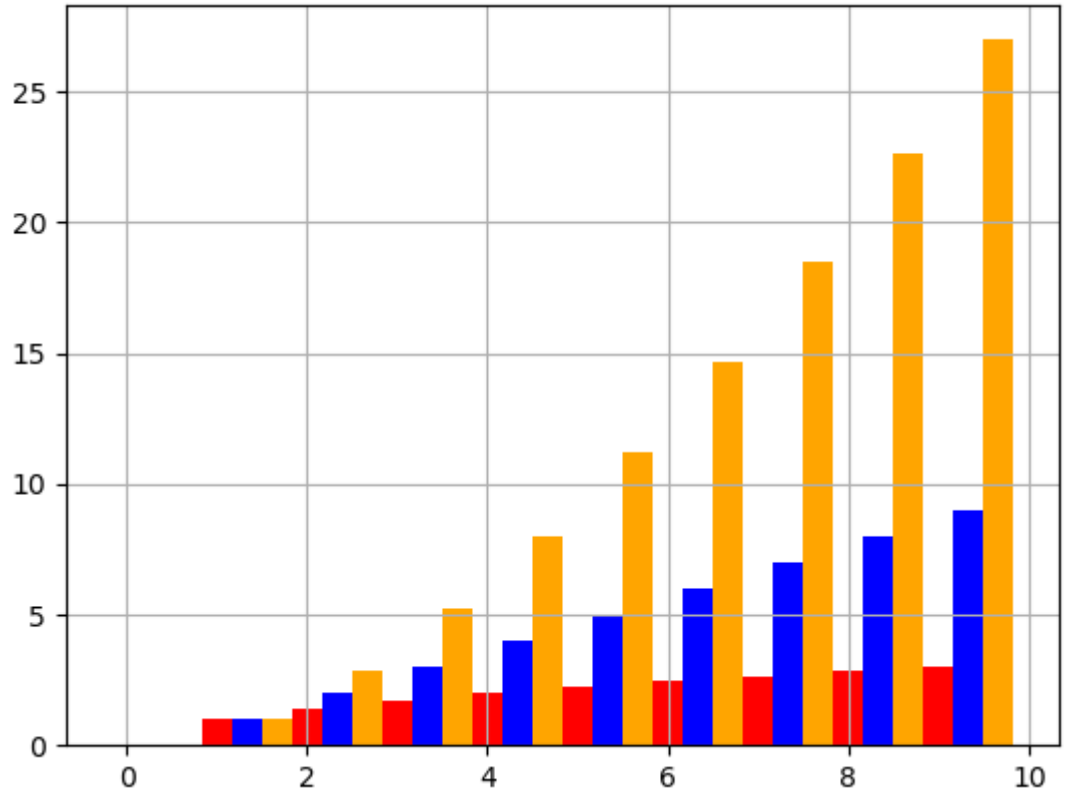
Varias barras en el mismo gráfico

- X = list(range(10))
- **X1 = [x+1/3 for x in X]**
- **X2 = [x+2/3 for x in X]**

- Y1 = [x**0.5 for x in X]
- Y2 = [x for x in X]
- Y3 = [x**1.5 for x in X]
- Y4 = [x**2 for x in X]

- plt.grid(True)
- plt.bar(X, Y1, color='red', **width=1/3**, align='center')
- plt.bar(X1, Y2, color='blue', **width=1/3**, align='center')
- plt.bar(X2, Y3, color='orange', **width=1/3**, align='center')
-

- El ancho de cada X se reparte entre el número de gráficas a mostrar.
- El ancho 1 ocupa todo el valor.



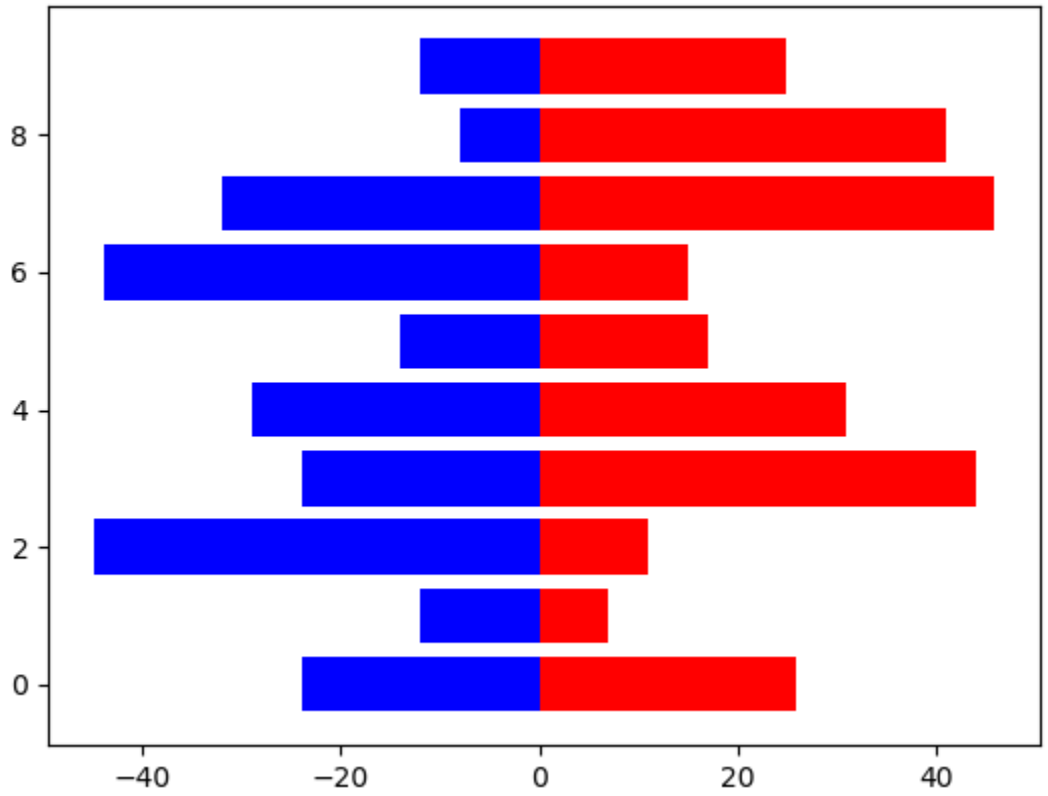
Barras Verticales

```
Y = list(range(10))

  X1 = [randint(5,50) for x in Y]      # positivos
  X2 = [randint(-50,-5) for x in X1] # negativos

  plt.barh(Y, X1, color='r',align='center')
  plt.barh(Y, X2, color='blue',align='center')

  plt.show()
```



Tipos de gráficos: Líneas

- plot()
  - Se puede indicar una etiqueta para mostrarlo luego en la leyenda.

  - plt.plot(X, Y1, **label**='Curva 1')
  - plt.plot(X, Y2, **label**='Curva 2')
  - plt.grid()
  - plt.legend()

Tipos de gráficos: Tipo tarta

- Para los gráficos tipo tarta el módulo pyplot dispone de la función **pie**().

- En una lista se envían los valores de cada porción y se pueden indicar las etiquetas en otra lista.

- Ejemplo:

```
plt.pie(L, labels=["Ana","Jorge","Maria","Miguel","Raul"])
plt.show()
```

- Otros parámetros:
  - shadow = True # poner una sombra
  - labeldistance = 0.7 # La distancia de la etiqueta al centro de la tarta

Tipos de gráficos: Histogramas

- Disponemos de la función: **hist**()
- Agrupar valores y contar el número de repeticiones
- Similares a los gráficos de barras.
- Por ejemplo, en una imagen con colores de 0 a 255 se puede generar un histograma con la frecuencia de cada pixel (cuantos pixels hay de cada color).

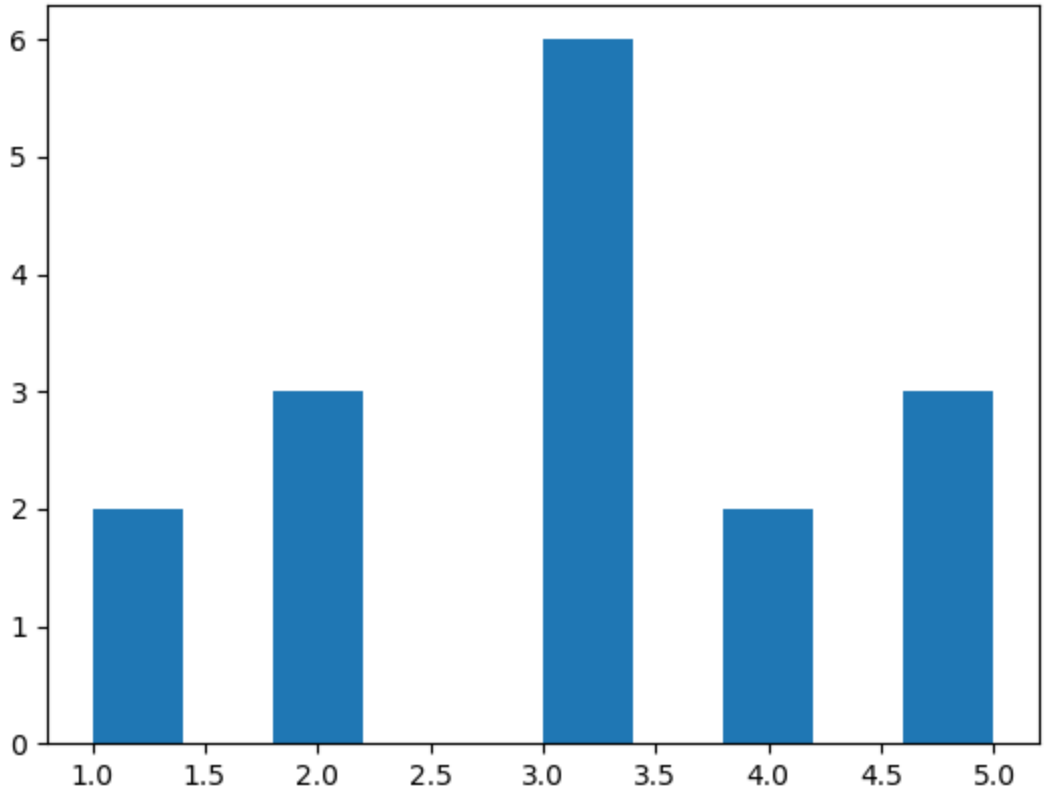
Ejemplo

- Una lista como esta:
- x = [1,3,3,2,2,2,5,5,5,3,3,3,3,1,4,4]

- plt.hist(x)
- plt.show()

- Genera el siguiente histograma →

- Se puede pasar una lista de listas de valores para representar varios histogramas en el mismo gráfico.



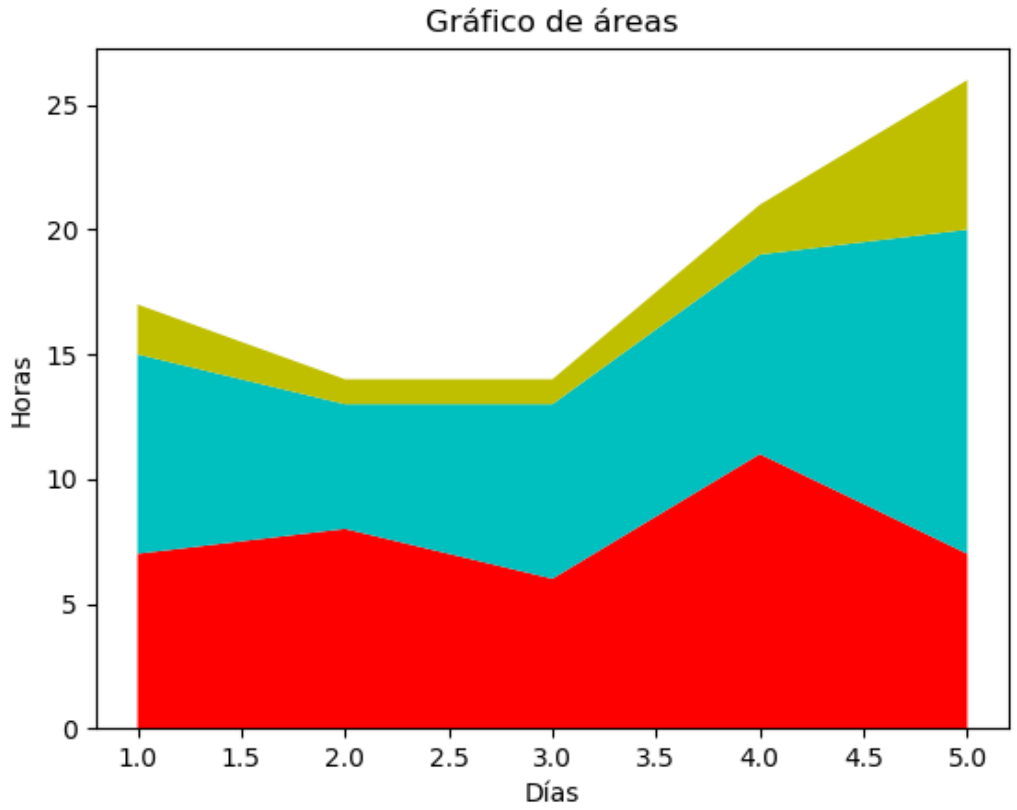
Tipos de gráficos: Áreas

- Disponemos de la función **stackplot**,
- Recibe los valores del eje X y listas de valores que representan cada una de las áreas.

```
dias = [1, 2, 3, 4, 5]

area1 = [7, 8, 6, 11, 7]
area2 = [8, 5, 7, 8, 13]
area3 = [2,1,1,2,6]

plt.stackplot(dias, area1, area2, area3,
colors =['r', 'c','y'])
```



Tipos de gráficos: Cajas

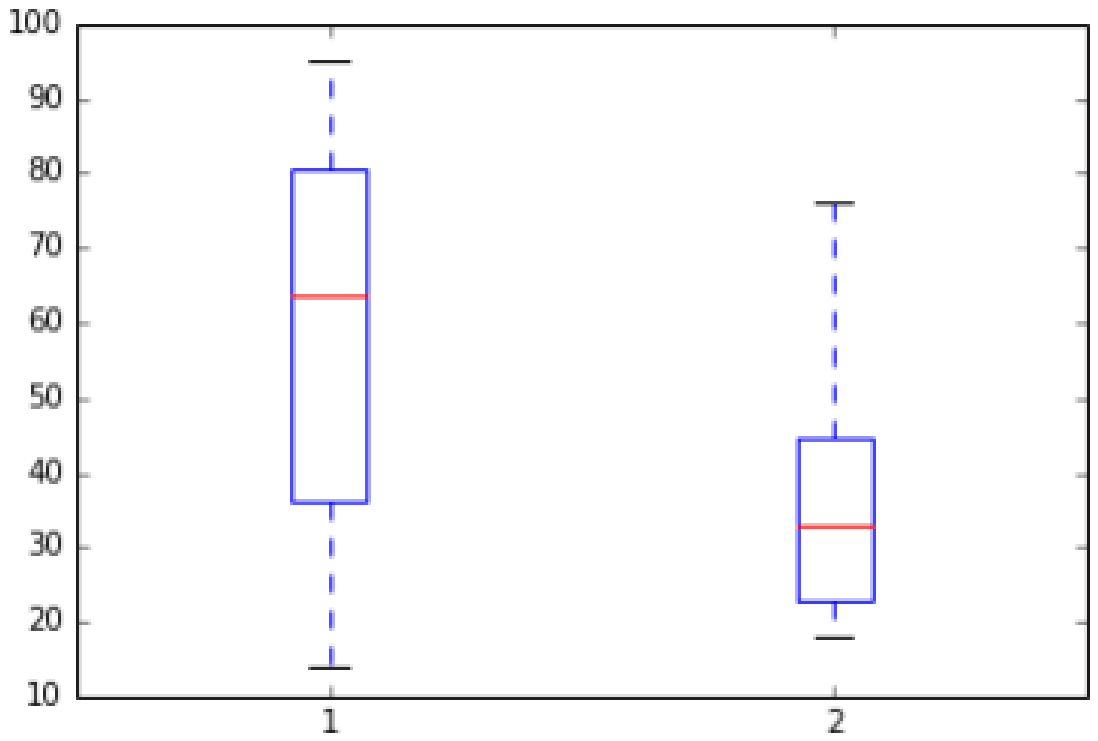
- Los diagramas de caja son diagramas descriptivos que ayudan a comparar la distribución de diferentes series de datos.

- El ejemplo más básico de un diagrama de caja en matplotlib se puede lograr simplemente pasando los datos como una lista de listas:

- Disponemos de la función **boxplot()**

Ejemplo

```
import matplotlib.pyplot as plt
dataline1 = [43,76,…,25,23,85]
dataline2 = [34,45,…,56,32,23]
data = [ dataline1, dataline2 ]
plt.boxplot( data )
```



Ejemplo 2

- Utilizando la librería numpy

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(123)
dataline1 = np.random.normal( loc=50, scale=20, size=18 )
dataline2 = np.random.normal( loc=30, scale=10, size=18 )
data = np.stack( [ dataline1, dataline2 ], axis=1 )
```

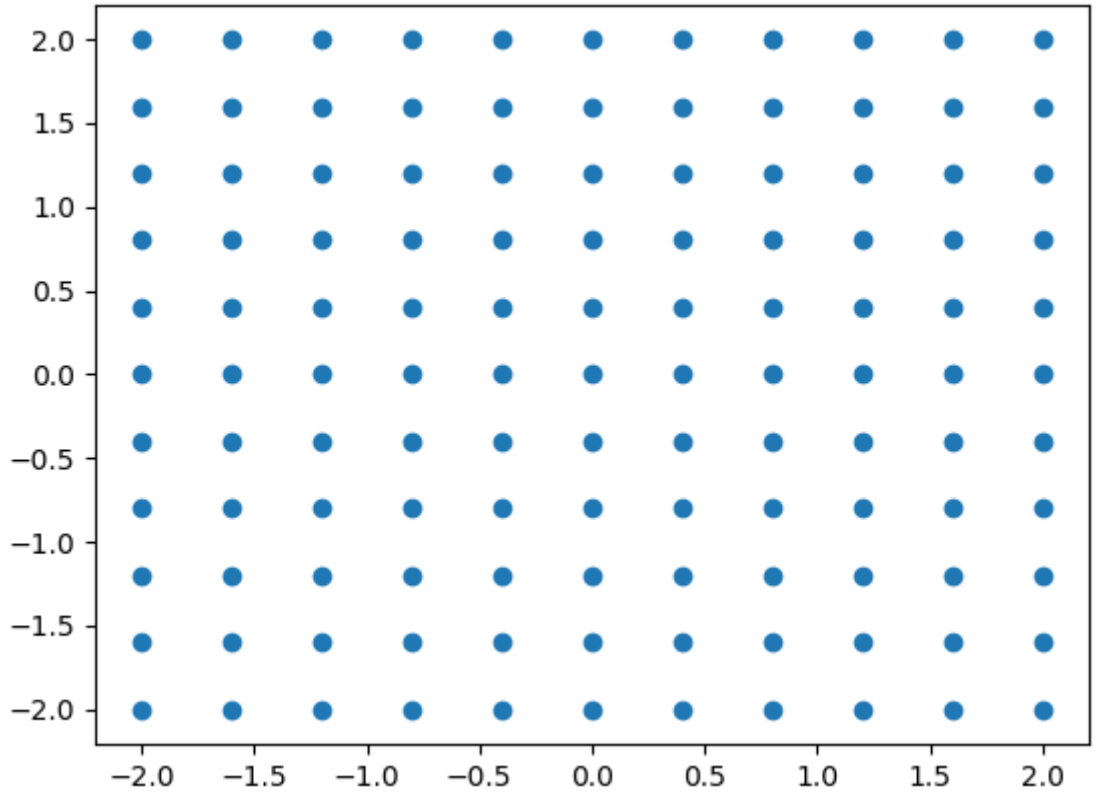
Tipos de gráficos: De contorno

- Disponemos de la función **contour** y **contourf**
- La segunda opción es para hacerlo relleno
- A ambas funciones se le manda x,y,z
- Se le puede modificar la transparencia con el parámetro alpha=0..1
- Se puede añadir una **barra de colores** para el contorno:
- **plt.colorbar()**
- Para ello recoger el contorno cuando llamamos a la función **contour / contourf**

Ejemplo (1ª parte)

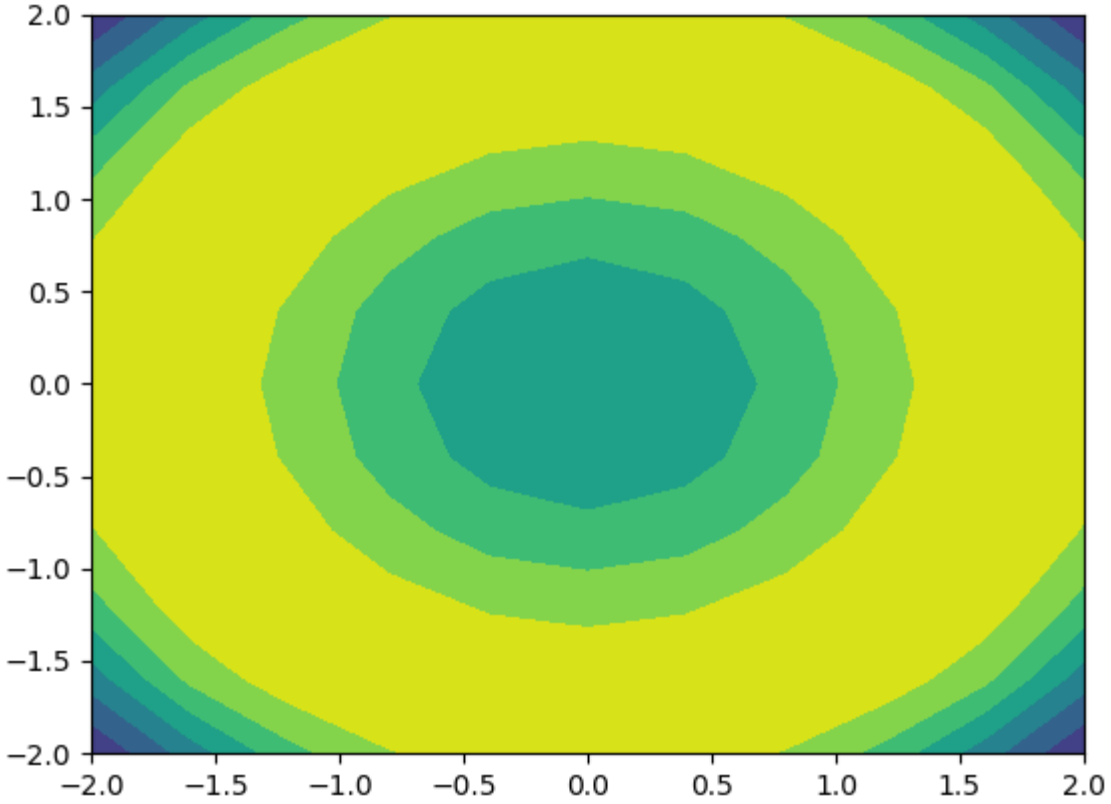
- Partimos de los puntos:
- x = np.linspace(-2, 2, 11)
- y = np.linspace(-2, 2, 11)

- xmesh, ymesh = np.meshgrid(x,y)
- plt.scatter(xmesh, ymesh)
- plt.show()



Ejemplo (2ª parte)

- Añadir un eje Z para el contorno:
  - zmesh = np.sin((xmesh**2+ymesh**2)*0.5)
  - # sin relleno:
  - #plt.contour(xmesh, ymesh, zmesh)
  - # con relleno
  - plt.contourf(xmesh, ymesh, zmesh)

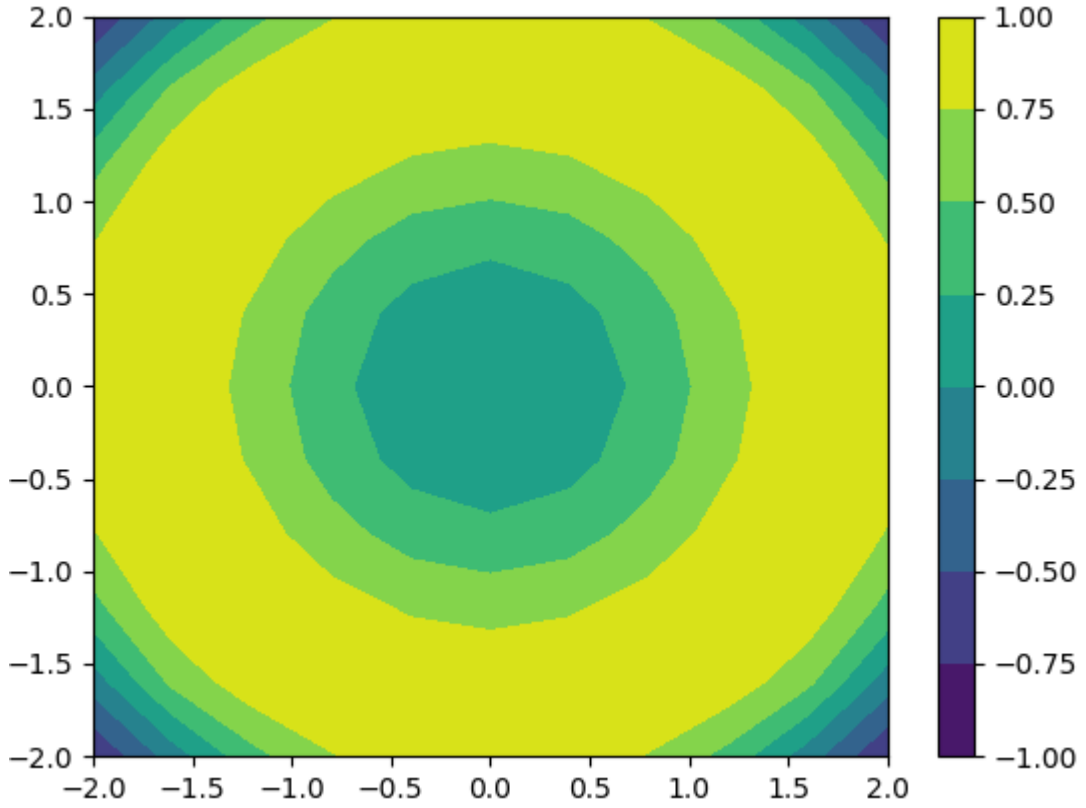


Con la barra de colores

con = plt.contourf(xmesh, ymesh, zmesh)

plt.colorbar(con)

plt.show()



Formatos

- **Archivo de imagen**
  - plt.savefig(“nombre_archivo”)
    - plt.savefig(“archivo.png”)
    - plt.savefig(“archivo.png”, transparent=True)

- **Archivo PDF:**
  - from matplotlib.backends.backend_pdf import PdfPages
  - pp = PdfPages(“fichero.pdf”)
  - pp.savefig()
  - pp.close()

Colores

Símbolo	Color
“b”	Azul
“g”	Verde
“r”	Rojo
“c”	Cian
“m”	Magenta
“y”	Amarillo
“k”	Negro
“w”	Blanco

Marcas y líneas

Símbolo	Descripción
"-"	Línea continua
"--"	Línea a trazos
"-."	Línea a puntos y rayas
":"	Línea punteada
"."	Símbolo punto
","	Símbolo pixel
"o"	Símbolo círculo relleno
"v"	Símbolo triángulo hacia abajo
"^"	Símbolo triángulo hacia arriba
"<"	Símbolo triángulo hacia la izquierda
">"	Símbolo triángulo hacia la derecha
"s"	Símbolo cuadrado
"p"	Símbolo pentágono
"*"	Símbolo estrella
"+"	Símbolo cruz
"x"	Símbolo X
"D"	Símbolo diamante
"d"	Símbolo diamante delgado

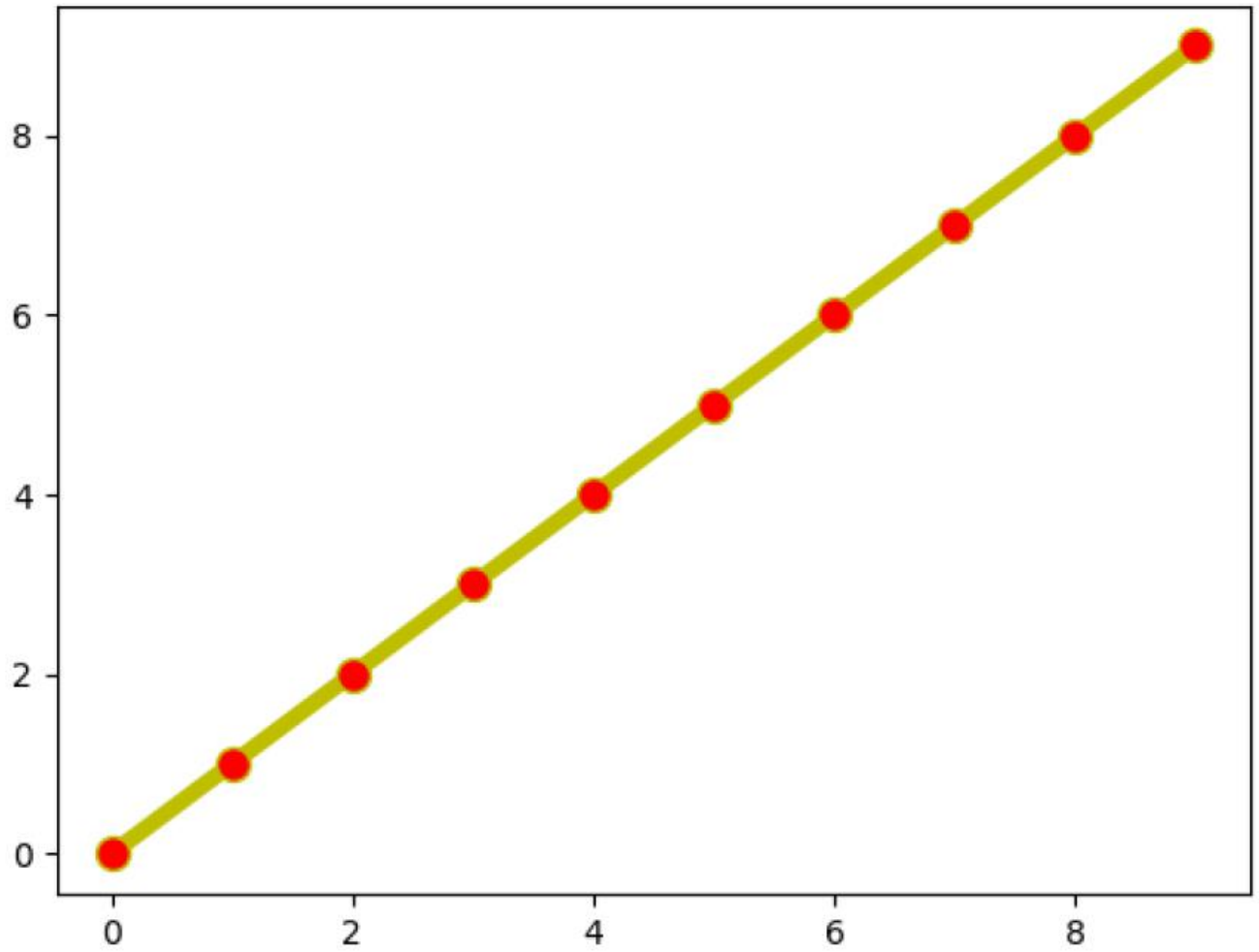
Más parámetros de plot()

- Cuando llamamos a la función **plot**() para representar la gráfica se pueden pasar más parámetros de configuración del aspecto del gráfico:

Parámetro	Significado y valores
alpha	grado de transparencia, float (0.0=transparente a 1.0=opaco)
color o c	Color de matplotlib
label	Etiqueta con cadena de texto, string
markeredge-color o mec	Color del borde del símbolo
markeredge-width o mew	Ancho del borde del símbolo, float (en número de puntos)
markerfaceco-lor o mfc	Color del símbolo
markersize o ms	Tamaño del símbolo, float (en número de puntos)
linestyle o ls	Tipo de línea, '-' '–' '-.' ':' 'None'
linewidth o lw	Ancho de la línea, float (en número de puntos)
marker	Tipo de símbolo,'+' '*' ',' '.' '1' '2' '3' '4' '<' '>' 'D' 'H' '^' '_' 'd' 'h' 'o' 'p' 's' 'v' 'x' 'l' TICKUP TICKDOWN TICKLEFT TICKRIGHT

```
In [15]: plot(x, lw=5, c='y', marker='o', ms=10, mfc='red')
Out[15]: [<matplotlib.lines.Line2D object at 0x8f0d14c>]
```

Resultado



Recoger instancias y modificar

- Cuando llamamos a la función plot() también es posible recoger las instancias de las gráficas creadas para modificar a posteriori.

```
>>> # Hago tres dibujos, capturando sus instancias
>>> # en las variables p1, p2 y p3
>>> p1, p2, p3 = plot(x, x,'b.',x, x2, 'rd', x, x3, 'g^')

>>> show()                    # Muestro en dibujo por pantalla
>>> p1.set_marker('o')        # Cambio el símbolo de la gráfica 1
>>> p3.set_color('y')         # Cambio el color de la gráfica 3
>>> draw()                    # Hacer los cambios
```

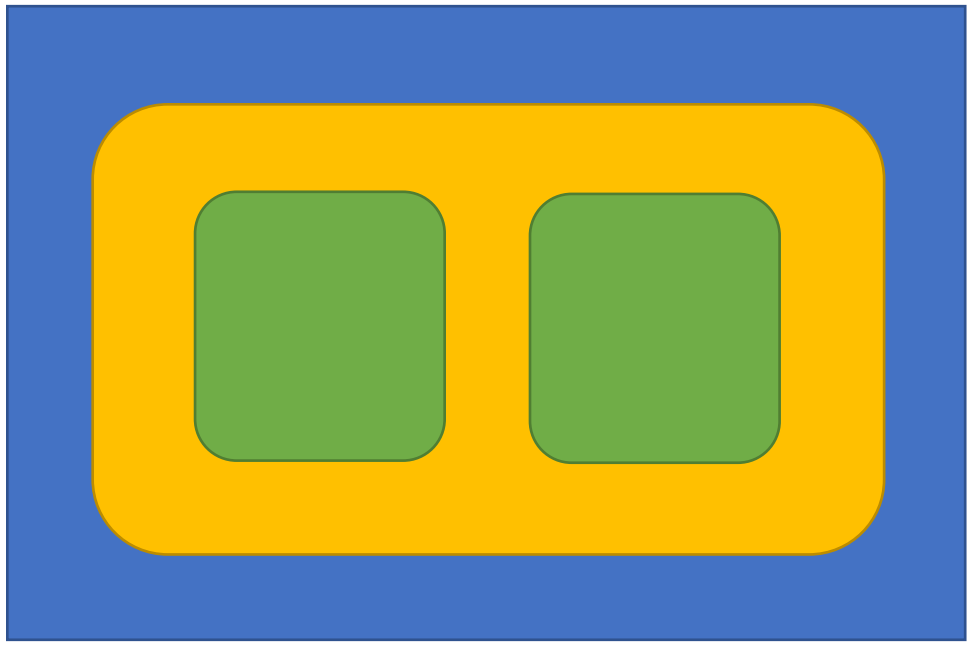
Objetos de la librería

- La ventana principal se desglosa en 3 objetos principales:

- Color Azul: **FigureCanvas**
  - Contenedor de clase Figure.

- Color Naranja: **Figure**
  - Contenedor de la clase Axes

- Color Verde: **Axes**
  - Zona que contiene los elementos gráficos(líneas, círculos, texto)



Objetos de la librería

- import matplotlib.pyplot as plt

- **# Todos los objetos hay que recogerlos en variables:**
- **# Crear la figura:**
- fig = plt.figure()

- **# Crear el gráfico**
- ax1 = fig.add_subplot(filas, cols, subgrafico_activo)
- ax1.plot(X, Y)

- **# Por ejemplo**: add_subplot(2,2,3) → **4 subgraficos y activo el 3**
- **# También se puede indicar 223 (mismo significado)**
- ax1.plot() # **Irá cambiando el método según elijamos un tipo de gráfico u otro.**

Clase Axes

• anotate()	Hacer anotaciones con flechas
• bar()	Gráfico de barras
• grid()	Colocar una rejilla en el objeto Axes
• pie()	Genera gráficos tarta
• scatter()	Puntos individuales
• set_color()	Configurar el color
• set_tick_params()	Parámetros de las marcas
• set_title()	El título del gráfico correspondiente
• set_xlabel()	Etiqueta eje X
• set_xlim()	Límites del eje X del gráfico
• set_xtickslabels()	Etiquetas de las marcas en el eje X

Clase Axes

• set_xticks()	Indicamos las marcas en el eje X
• set_ylabel()	Etiqueta para el eje Y
• set_ylim()	Límites en el eje Y
• set_ytickslabels()	Etiquetas para las marcas del eje Y
• set_yticks()	Las marcas del eje Y
• tick_params()	Configurar parámetros en las marcas
• twinx()	Permite compartir el eje x con ejes y independientes
• twiny()	Permite compartir el eje y con ejes x independientes

Clase Figure

- set_window_title()
- subtitle()

Coloca un titulo a la ventana

Pasando primero por un objeto **Canvas**

Título global al objeto figure

Módulo pyplot

- **Funciones**:
  - **figure**() Crea un objeto Figure con determinadas características.
  - **rcParams**(): configurar los parámetros que tienen todos los gráficos por defecto.

- El módulo pyplot era una forma de acceder a funciones de los gráficos sin tener que utilizar los objetos de la librería.
  - Simplifican el uso de la librería matplotlib

Ejemplo

```
plt.rcParams['toolbar'] = 'None'
plt.rcParams['font.size'] = '10'

fig = plt.figure(figsize=(12,5), frameon=False, facecolor='yellow')
fig.canvas.set_window_title(" …")
fig.suptitle("….", fontsize=15, color="red")

ax1=fig.add_subplot(121)
ax1.set_xlim(0.5, 10.5)
ax1.set_ylim(0, …)
axi1.grid(True, axis="y", ls="-",color="0.3")
ax1.set_xticks(…) # Las divisiones del eje X
ax1.set_xlabel("Etiqueta eje X")
ax1.set_ylabel("Etiqueta eje Y")
ax1.set_xtickslabels(['eti1', 'eti2','…'], fontsize=9, color='b')
ax1.set_title("Título gráfico", color="blue")
```

Compartir ejes

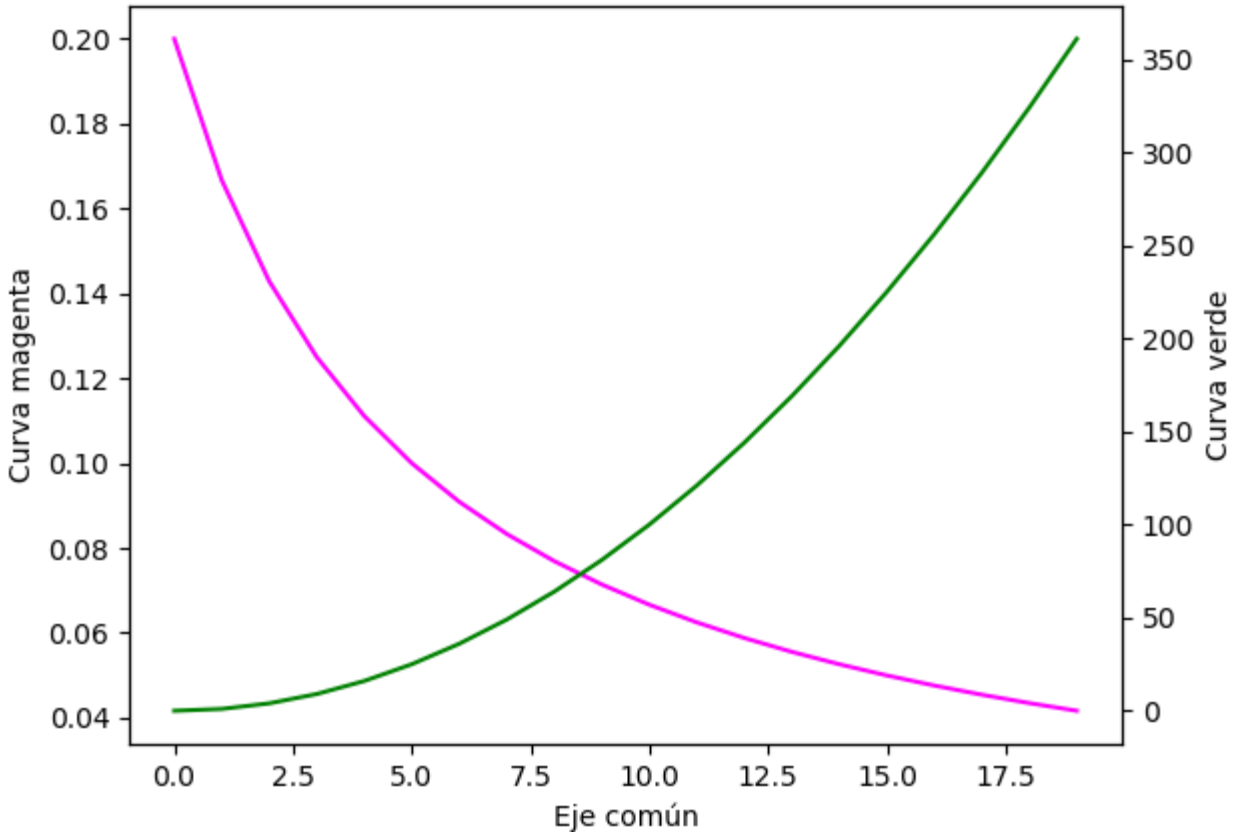
- Las funciones **twinx** y **twiny** nos permiten compartir uno de los ejes y tener el otro distinto para dos gráficas en el mismo objeto gráfico.

Ejemplo

```
X = list(range(20))
Y1 = [1/(x+5) for x in X]
Y2 = [x**2 for x in X]

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(X, Y1, color="magenta")
ax1.set_xlabel("Eje común")
ax1.set_ylabel("Curva magenta")

ax2 = ax1.twinx()
c2, = ax2.plot(X, Y2)
c2.set_color("green")
ax2.set_ylabel("Curva verde")
plt.show()
```



Enlaces de interés

- Enlace a la librería:
  - https://matplotlib.org/
  - https://matplotlib.org/tutorials/index.html

- https://claudiovz.github.io/scipy-lecture-notes-ES/intro/matplotlib/matplotlib.html

- https://www.datacamp.com/community/blog/python-matplotlib-cheat-sheet