Guide 8 introduction to matplotlib

May 21, 2019

Guía 8: Introducción a Matplotlib
Ingeniería en Estadística, Universidad de Valparaíso
Profesor: Eduardo Jorquera - eduardo.jorquera@postgrado.uv.cl

1 Mat-plot-lib

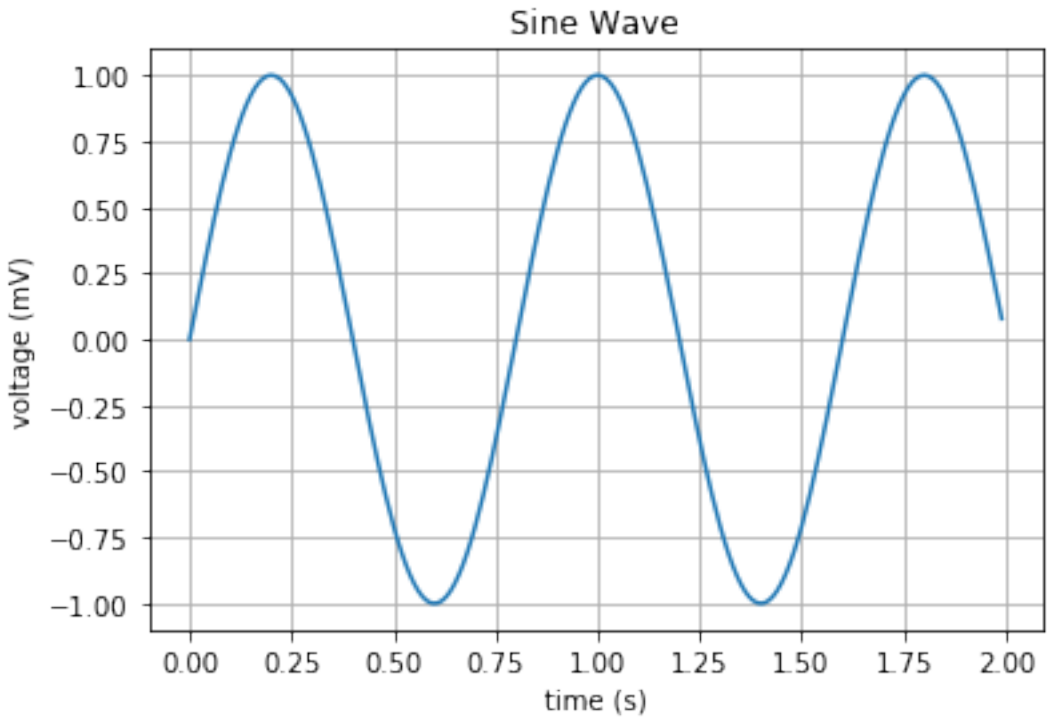
Como lo sugiere su nombre (en inglés), matplotlib es una librería de gráficos de python. Es útil para mostrar resultados, para estudiar el comportamiento de funciones, etc.

2 Gráfico de línea

Usando la función `plot()`, se puede crear un gráfico de línea. Se puede definir la cuadrícula, el eje `x` e `y`, sus etiquetas, escala, título, y opciones de visualización.

Ejemplo:

```
In [2]: from pylab import *
        t = arange(0.0, 2.0, 0.01)
        s = sin(2.5*pi*t)
        plot(t, s)

        xlabel('time (s)')
        ylabel('voltage (mV)')
        title('Sine Wave')
        grid(True)
        show()
```



Las líneas

```
In [ ]: from pylab import *
        t = arange(0.0, 2.0, 0.01)
        s = sin(2.5*pi*t)
```

Simplemente definen lo que se graficará. Al agregarse

```
In [ ]: plot(t, s)
        show()
```

se muestra el gráfico. Las otras sentencias tienen el mismo sentido: el comando `xlabel()` permite establecer la etiqueta del eje x, así como `ylabel()` para el eje y. El comando `title()` establece el título del gráfico y `grid(True)` muestra la cuadrícula.

Si quieres guardar el gráfico recién hecho, puedes agregar a la celda de jupyter una nueva línea que contenga

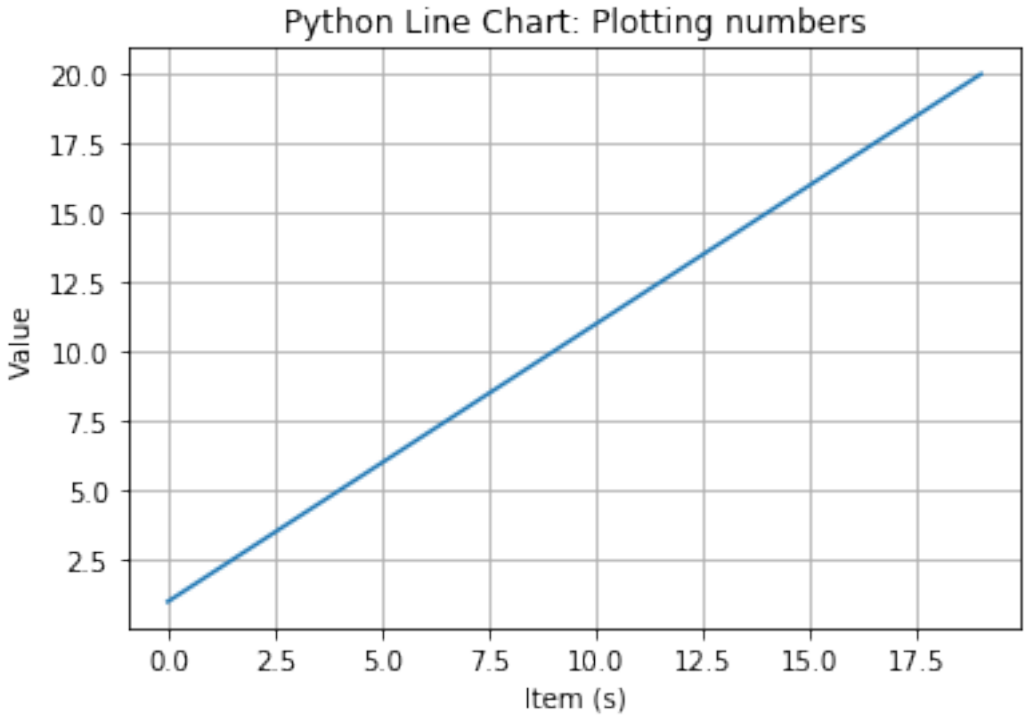
```
savefig("line_chart.png")
```

2.1 Grafico personalizado de línea

Si quieres graficar usando un arreglo (lista), puedes hacerlo de la siguiente manera:

```
In [3]: from pylab import *

        t = arange(0.0, 20.0, 1)
```

```
s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
plot(t, s)

xlabel('Item (s)')
ylabel('Value')
title('Python Line Chart: Plotting numbers')
grid(True)
show()
```



El comando `t = arange(0.0, 20.0, 1)` define un arreglo que comienza desde el 0 hasta el 19 de uno en uno:

```
In [4]: arange(0.0, 20.0, 1)

Out[4]: array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
               13., 14., 15., 16., 17., 18., 19.])
```

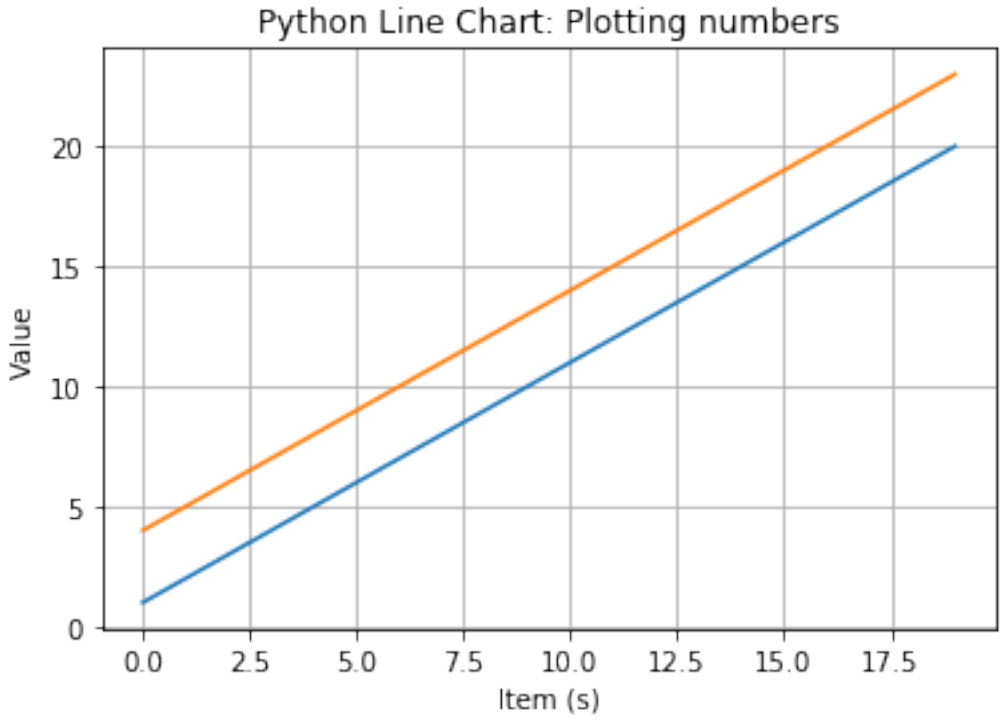
2.2 Gráficos múltiples

Si quieres graficar múltiples líneas en un mismo gráfico, somplemente llama a la función `plot()` múltiples veces.

Ejemplo:

```
In [5]: from pylab import *
```

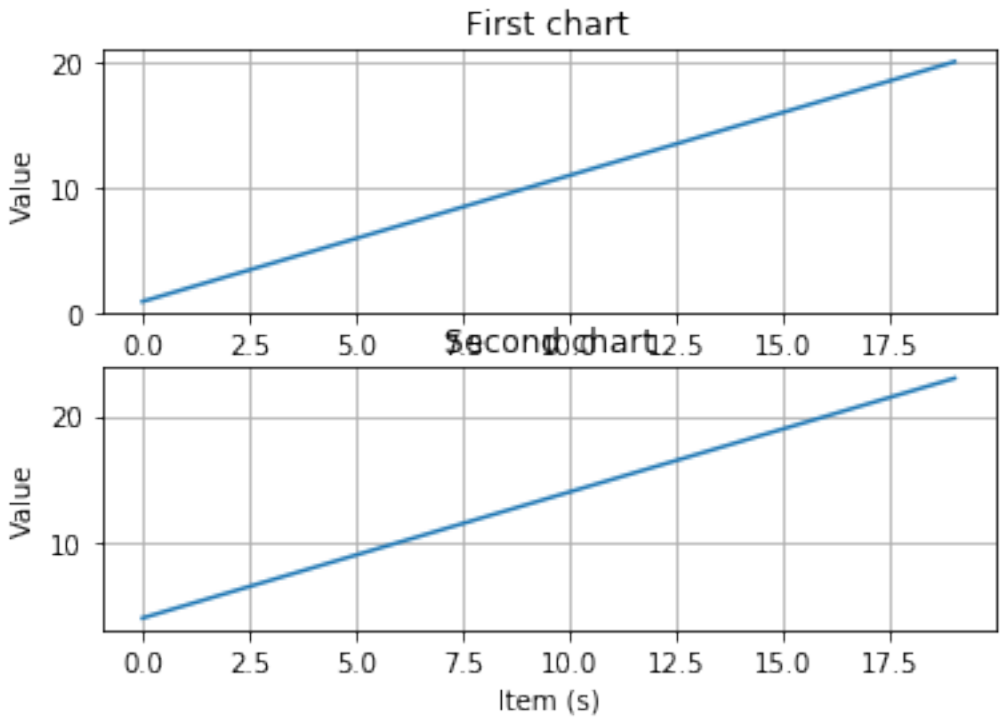
```
t = arange(0.0, 20.0, 1)
s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
s2 = [4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]
plot(t, s)
plot(t, s2)

xlabel('Item (s)')
ylabel('Value')
title('Python Line Chart: Plotting numbers')
grid(True)
show()
```



Si quieres mostrarlos en diferentes cuadrículas, pero en el mismo output (o en la misma ventana), puedes usar lo siguiente:

```
In [6]: import matplotlib.pyplot as plt
        from pylab import *

        t = arange(0.0, 20.0, 1)
        s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
        s2 = [4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]

        plt.subplot(2, 1, 1)
        plt.plot(t, s)
        plt.ylabel('Value')
```

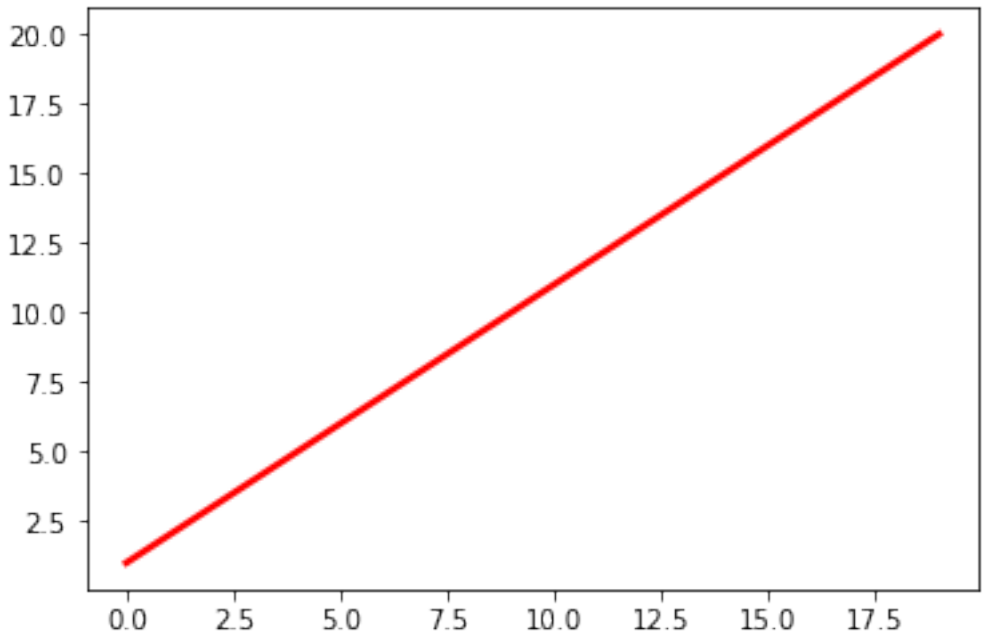
```
plt.title('First chart')
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(t, s2)
plt.xlabel('Item (s)')
plt.ylabel('Value')
plt.title('Second chart')
plt.grid(True)
plt.show()
```



La clave aquí está en la función `plt.subplot()`. Este comando especifica el número de filas, número de columnas y número de figuras (en ese order), por eso en el ejemplo de arriba el comando es `plt.subplot(2, 1, 1)`.

Puedes personalizar tu gráfico engruesando la línea y cambiando los colores:

```
In [12]: plot(t, s, color="red", linewidth=2.5, linestyle="-")

Out[12]: [<matplotlib.lines.Line2D at 0x7ff1cffde1d0>]
```



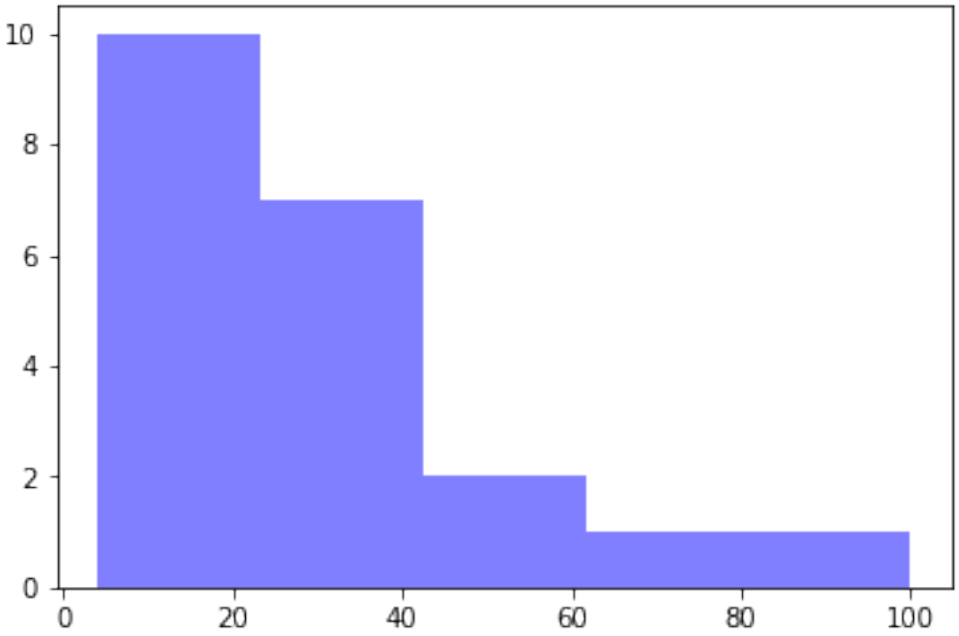
3 Histogramas

Un histograma muestra la frecuencia en el eje vertical y el eje horizontal es otra dimensión. Usualmente son "contenedores", donde cada contenedor tiene un valor mínimo y máximo. Cada contenedor también tiene una frecuencia entre x e infinito.

3.1 Ejemplo de histograma en matplotlib

Aquí hay un ejemplo básico de histograma:

```
In [17]: import numpy as np
         import matplotlib.mlab as mlab
         import matplotlib.pyplot as plt

         x = [21,22,23,4,5,6,77,8,9,10,31,32,33,34,35,36,37,18,49,50,100]
         num_bins = 5
         n, bins, patches = plt.hist(x, num_bins, facecolor='blue', alpha=0.5)
         plt.show()
```

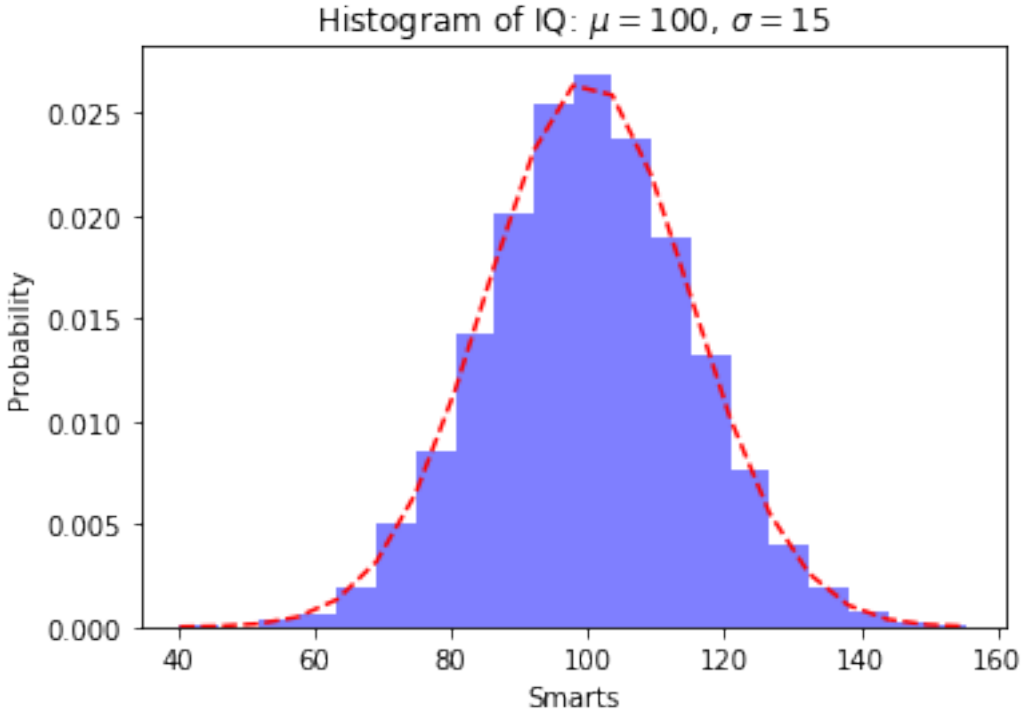


3.2 Un histograma completo usando matplotlib

Puedes agregar muchas cosas a tu histograma, tales como una línea de ajuste a alguna distribución, etiquetas, etc. El código de abajo crea un histograma más avanzado.

```
In [32]: import numpy as np
         import matplotlib.mlab as mlab
         import matplotlib.pyplot as plt
         from scipy.stats import norm

         # Datos de ejemplo
         mu = 100 # media de la distribución
         sigma = 15 # desviación estándar de la distribución
         x = mu + sigma * np.random.randn(10000)

         num_bins = 20
         # el histograma de los datos
         n, bins, patches = plt.hist(x, num_bins, normed=1, facecolor='blue', alpha=0.5)

         # agrega la línea con "mejor ajuste"
         y = norm.pdf(bins, mu, sigma)
         plt.plot(bins, y, 'r--')
         plt.xlabel('Smarts')
         plt.ylabel('Probability')
         plt.title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')
```

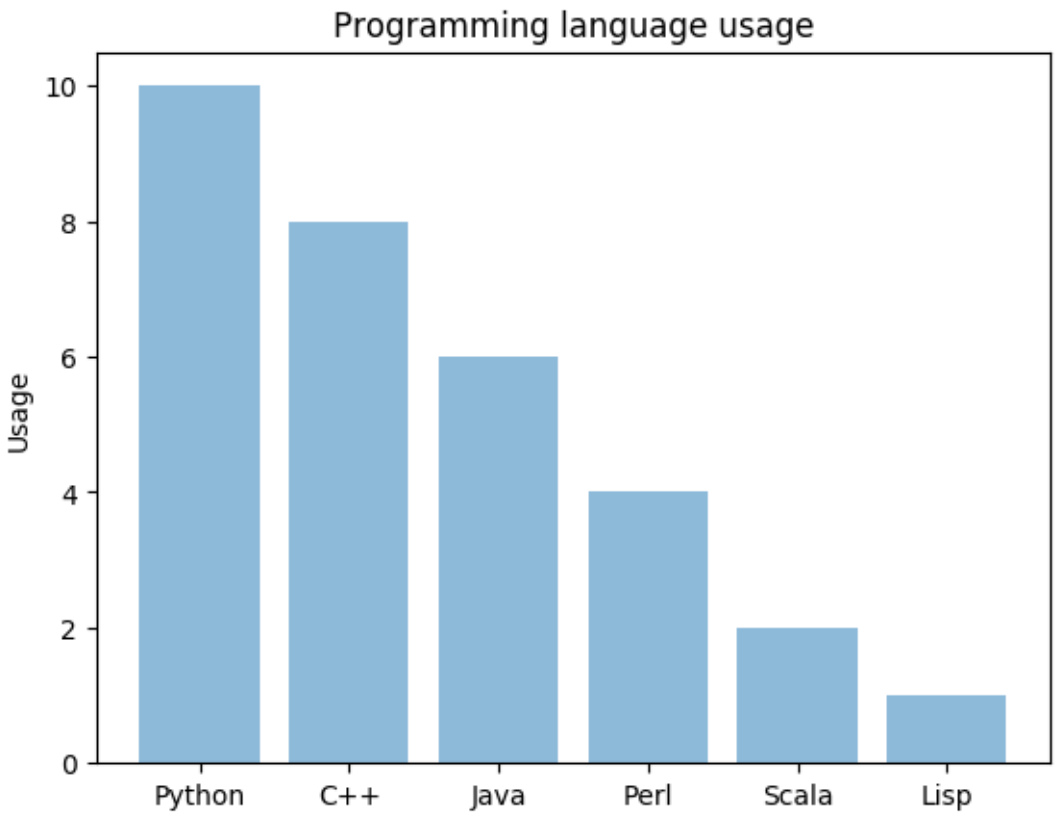
```
# Arregla el espaciado para evitar el revorte del eje y
plt.subplots_adjust(left=0.15)
plt.show()
```



4 Gráfico de barra

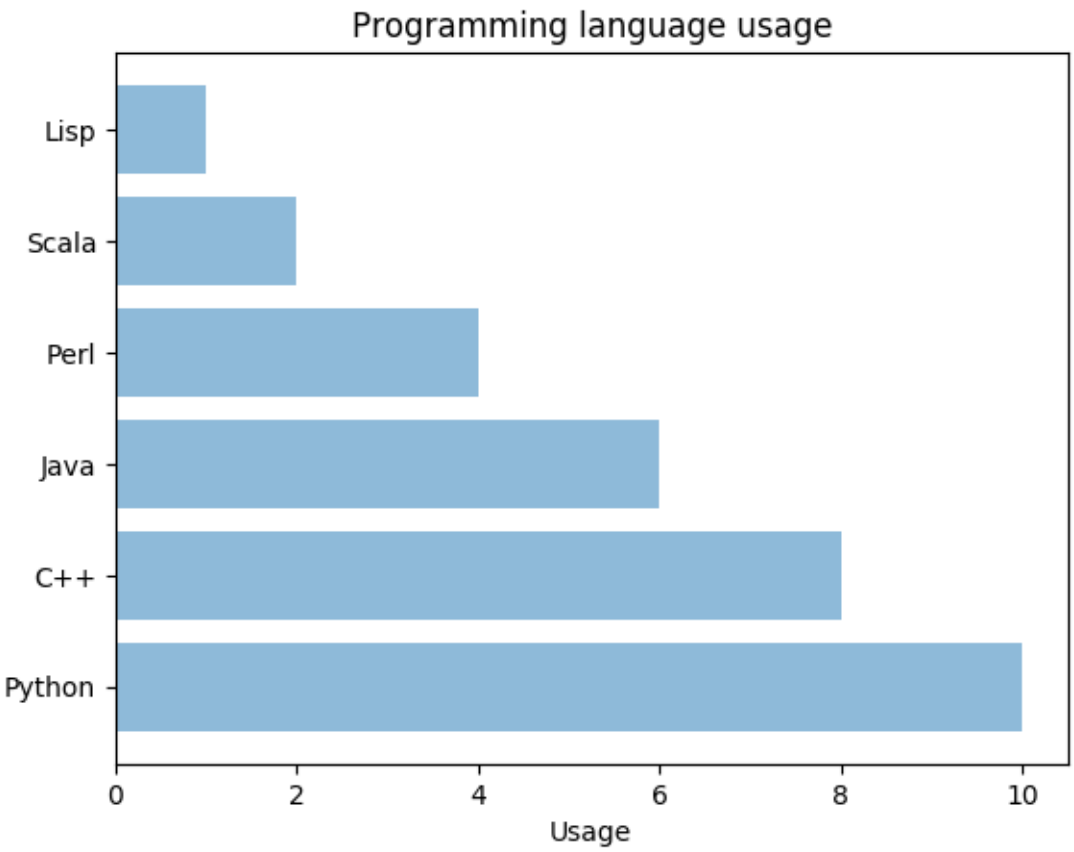
Cuando se está en presencia de variables categóricas es bastante útil, mira el ejemplo para aprender cómo se hace:

```
In [33]: import matplotlib.pyplot as plt; plt.rcdefaults()
         import numpy as np
         import matplotlib.pyplot as plt

         objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
         y_pos = np.arange(len(objects))
         performance = [10,8,6,4,2,1]

         plt.bar(y_pos, performance, align='center', alpha=0.5)
         plt.xticks(y_pos, objects)
         plt.ylabel('Usage')
         plt.title('Programming language usage')

         plt.show()
```



Este gráfico también puede ser horizontal:

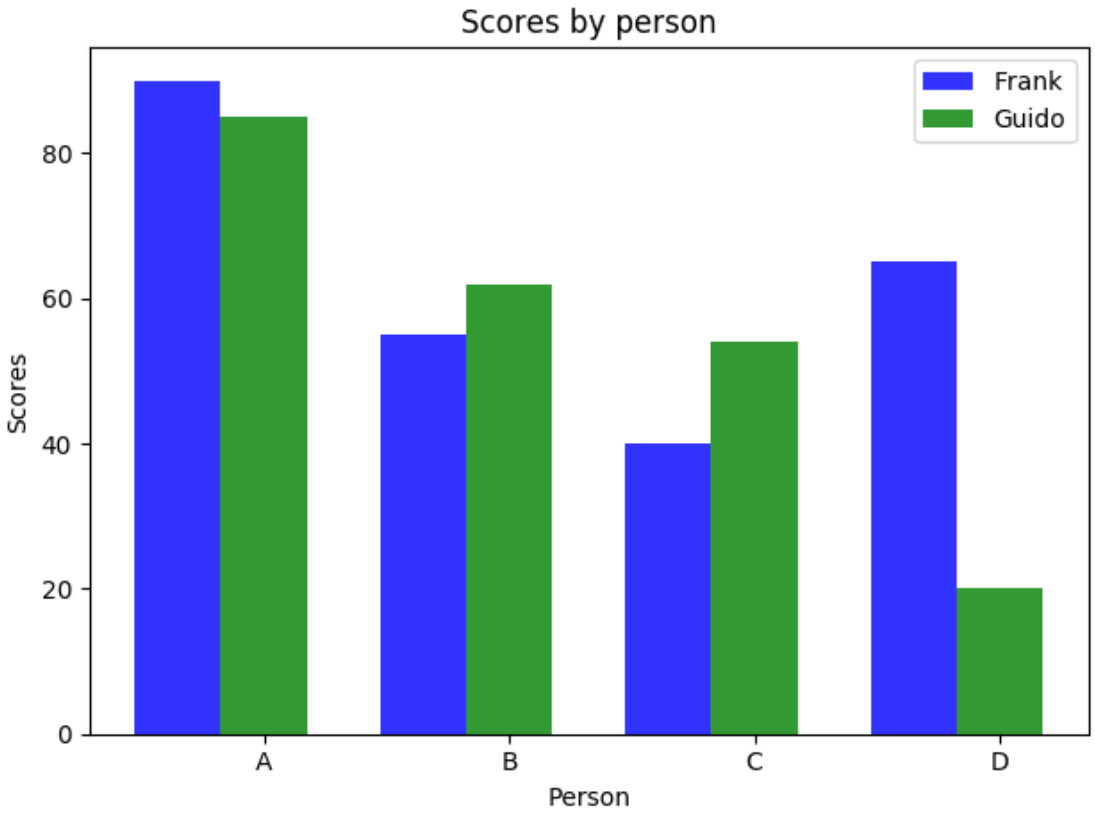
```
In [34]: import matplotlib.pyplot as plt; plt.rcdefaults()
         import numpy as np
         import matplotlib.pyplot as plt

         objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
         y_pos = np.arange(len(objects))
         performance = [10,8,6,4,2,1]

         plt.barh(y_pos, performance, align='center', alpha=0.5)
         plt.yticks(y_pos, objects)
         plt.xlabel('Usage')
         plt.title('Programming language usage')

         plt.show()
```



También puedes comparar dos conjuntos de datos a través del siguiente código:

```
In [36]: import numpy as np
         import matplotlib.pyplot as plt

         # Datos a graficar
         n_groups = 4
         means_frank = (90, 55, 40, 65)
         means_guido = (85, 62, 54, 20)

         # Crear un gráfico
         fig, ax = plt.subplots()
         index = np.arange(n_groups)
         bar_width = 0.35
         opacity = 0.8

         rects1 = plt.bar(index, means_frank, bar_width,
         alpha=opacity,
         color='b',
         label='Frank')
```

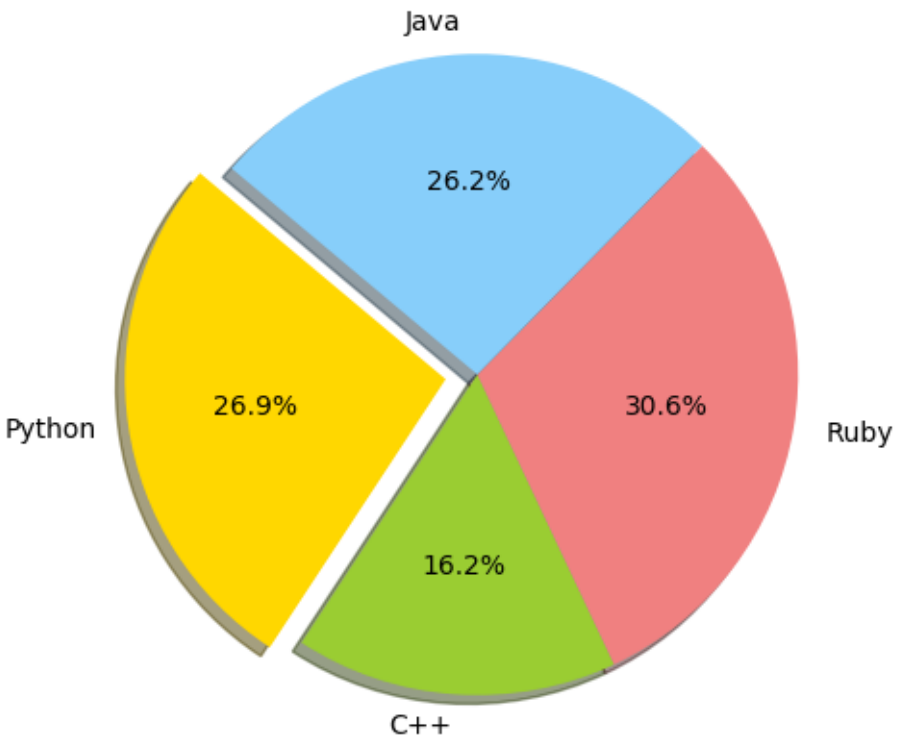
```
rects2 = plt.bar(index + bar_width, means_guido, bar_width,
alpha=opacity,
color='g',
label='Guido')

plt.xlabel('Person')
plt.ylabel('Scores')
plt.title('Scores by person')
plt.xticks(index + bar_width, ('A', 'B', 'C', 'D'))
plt.legend()

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



5 Gráficos de torta

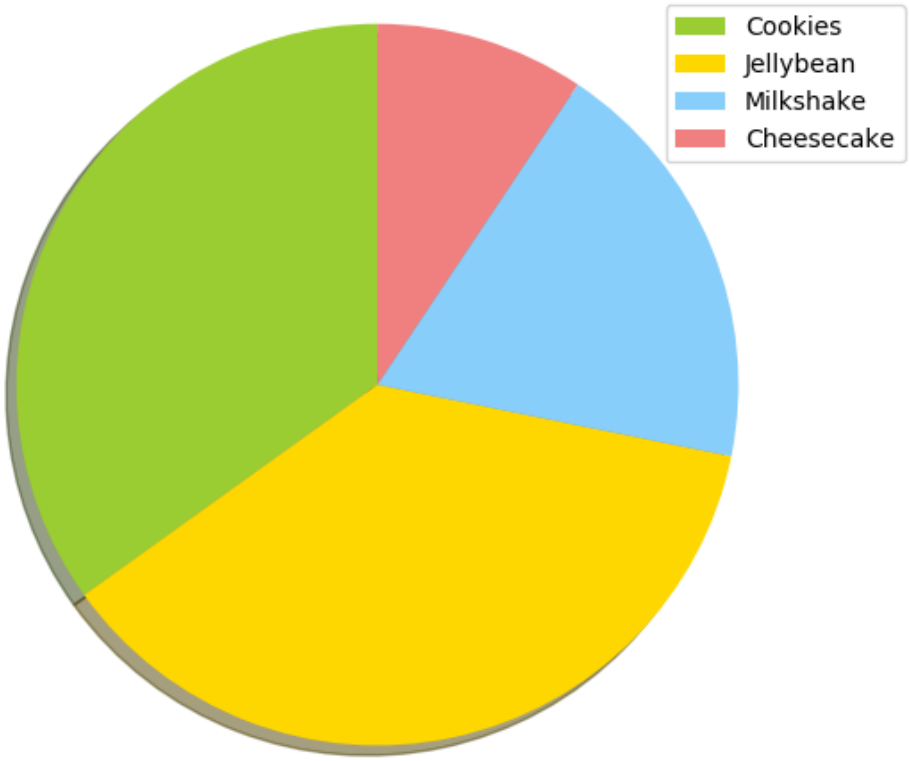
```
In [37]: import matplotlib.pyplot as plt

         # Datos a graficar
         labels = 'Python', 'C++', 'Ruby', 'Java'
```

```
sizes = [215, 130, 245, 210]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0)  # explode 1st slice

# gráfico:
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```



Para agregar una leyenda, usa la función `plt.legend()`.

```
In [38]: import matplotlib.pyplot as plt

         labels = ['Cookies', 'Jellybean', 'Milkshake', 'Cheesecake']
         sizes = [38.4, 40.6, 20.7, 10.3]
         colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
         patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)
         plt.legend(patches, labels, loc="best")
         plt.axis('equal')
         plt.tight_layout()
         plt.show()
```

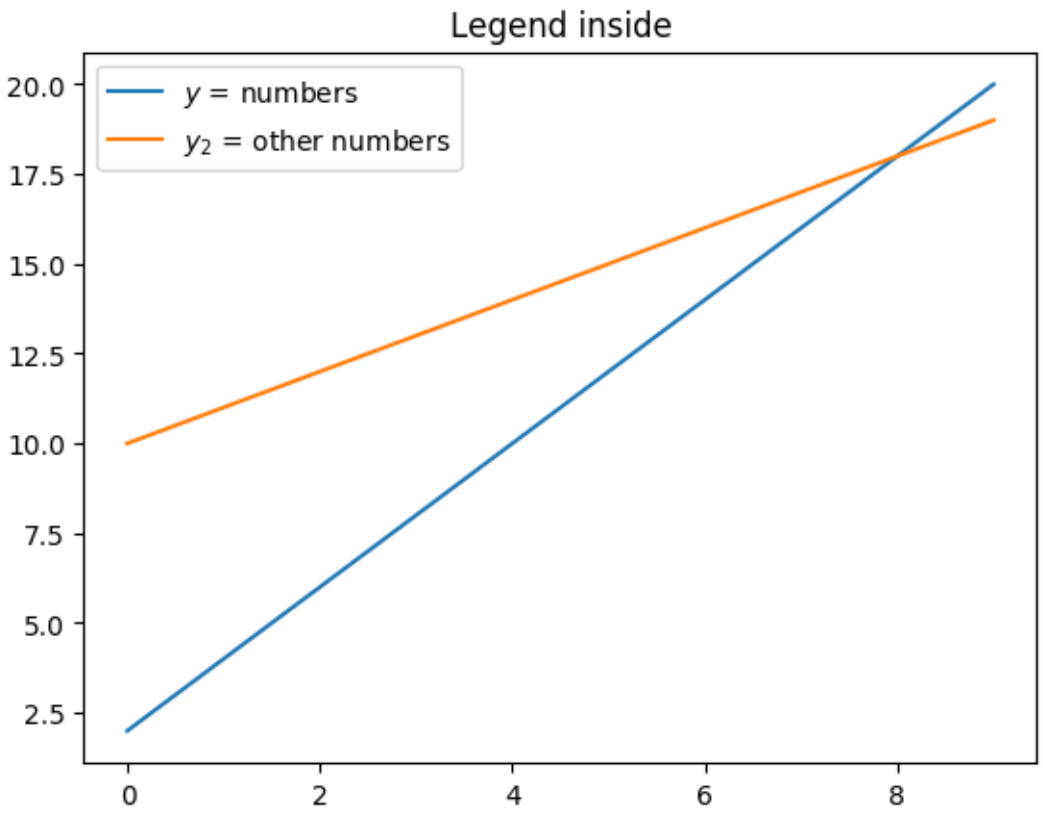


6 Leyendas en matplotlib

Las leyendas pueden ser posicionadas en varios lugares del gráfico: Una leyenda puede estar puesta por dentro o por fuera del gráfico y la posición puede moverse.

El método `legend()` agrega la leyenda al gráfico. Aquí te mostraré algunos ejemplos de leyendas en matplotlib.

```
In [49]: import matplotlib.pyplot as plt
         import numpy as np

         y = [2,4,6,8,10,12,14,16,18,20]
         y2 = [10,11,12,13,14,15,16,17,18,19]
         x = np.arange(10)
         fig = plt.figure()
         ax = plt.subplot(111)
         ax.plot(x, y, label='$y$ = numbers')
         ax.plot(x, y2, label='$y_2$ = other numbers')
         plt.title('Legend inside')
         ax.legend()
         plt.show()
```



6.1 Leyenda abajo del gráfico

Para posicionar la leyenda en la parte de abajo del gráfico, se debe cambiar los argumentos de `legend()` a:

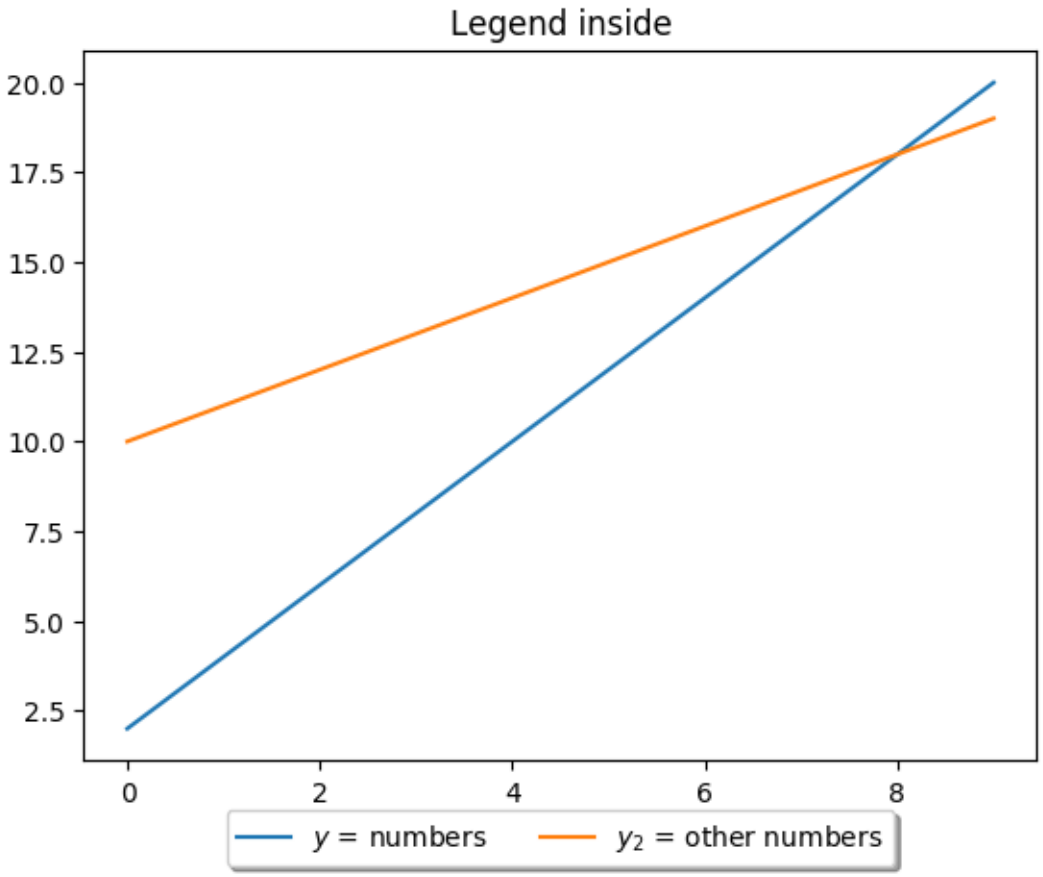
```
In [50]: ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), shadow=True, ncol=2)

Out[50]: <matplotlib.legend.Legend at 0x7ff1b3dd9e80>
```

Toma en consideración que el número de columnas ahora es dos (`ncol=2`) y establecimos una sombra (`shadow=True`).

El código completo sería:

```
In [51]: import matplotlib.pyplot as plt
         import numpy as np

         y = [2,4,6,8,10,12,14,16,18,20]
         y2 = [10,11,12,13,14,15,16,17,18,19]
         x = np.arange(10)
         fig = plt.figure()
         ax = plt.subplot(111)
         ax.plot(x, y, label='$y$ = numbers')
```

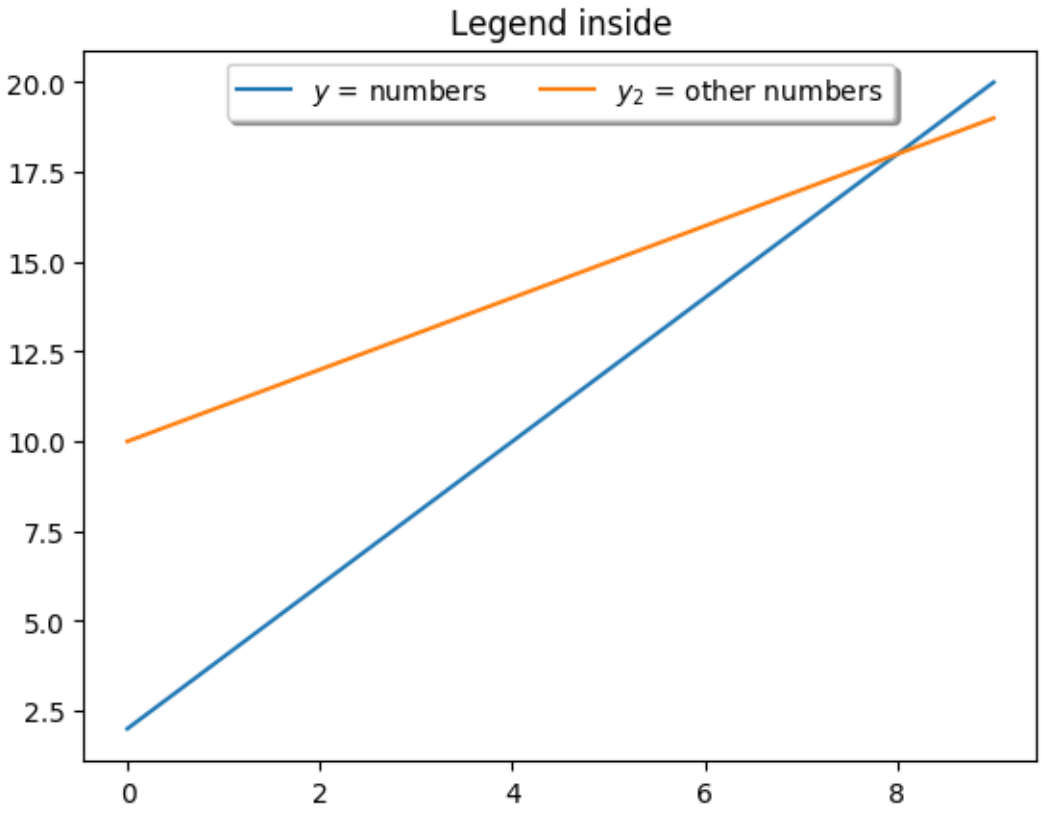
```
ax.plot(x, y2, label='$y_2$ = other numbers')
plt.title('Legend inside')
ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), shadow=True, ncol=2)
plt.show()
```



Para poner la leyenda en la parte de arriba, hay que cambiar los valores de entrada de `bbox_to_anchor`:

```
In [52]: ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.00), shadow=True, ncol=2)

Out[52]: <matplotlib.legend.Legend at 0x7ff1b3b00e80>

In [53]: import matplotlib.pyplot as plt
         import numpy as np

         y = [2,4,6,8,10,12,14,16,18,20]
         y2 = [10,11,12,13,14,15,16,17,18,19]
         x = np.arange(10)
         fig = plt.figure()
         ax = plt.subplot(111)
         ax.plot(x, y, label='$y$ = numbers')
```

```
ax.plot(x, y2, label='$y_2$ = other numbers')
plt.title('Legend inside')
ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.00), shadow=True, ncol=2)
plt.show()
```



6.2 Leyenda por fuera del gráfico a la derecha

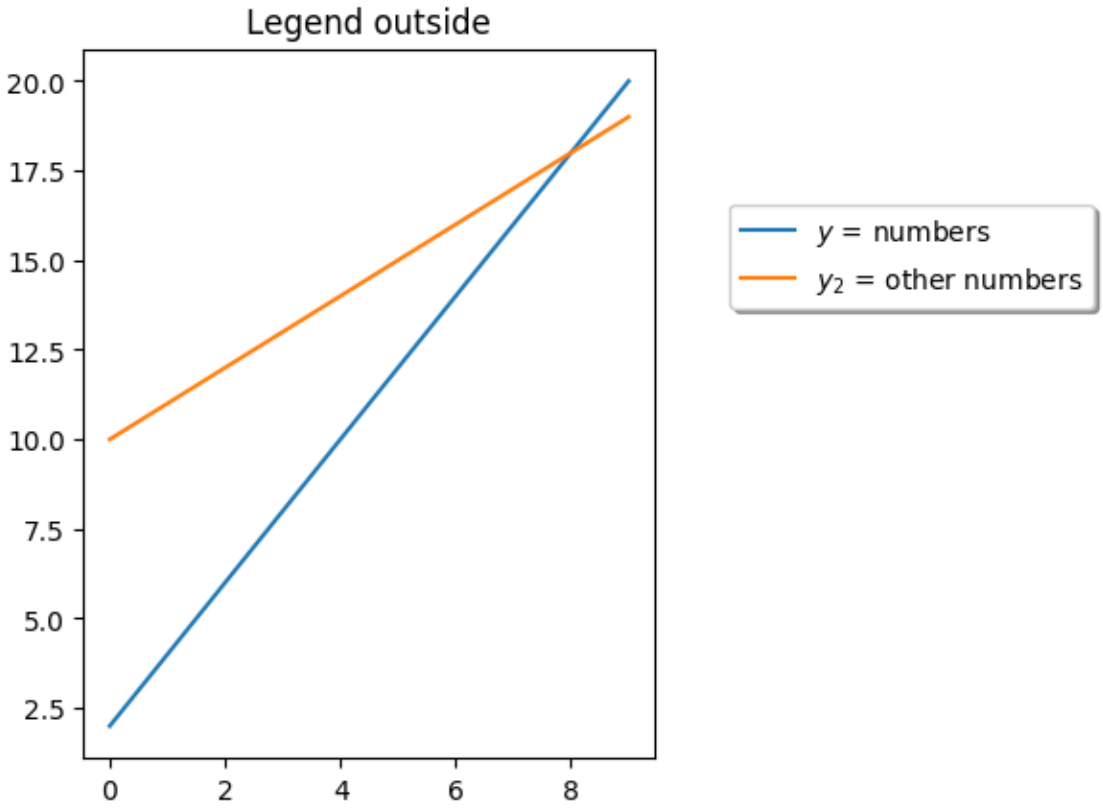
La leyenda se puede posicionar afuera de la caja de gráfico posicionándola respecto a ésta:

```
In [57]: chartBox = ax.get_position()
         ax.set_position([chartBox.x0, chartBox.y0, chartBox.width*0.6, chartBox.height])
         ax.legend(loc='upper center', bbox_to_anchor=(1.45, 0.8), shadow=True, ncol=1)

Out[57]: <matplotlib.legend.Legend at 0x7ff1b3d59780>
```

Aquí el código completo:

```
In [61]: import matplotlib.pyplot as plt
         import numpy as np

         y = [2,4,6,8,10,12,14,16,18,20]
         y2 = [10,11,12,13,14,15,16,17,18,19]
```

```
x = np.arange(10)
fig = plt.figure()
ax = plt.subplot(111)
ax.plot(x, y, label='$y$ = numbers')
ax.plot(x, y2, label='$y_2$ = other numbers')
plt.title('Legend outside')
chartBox = ax.get_position()
ax.set_position([chartBox.x0, chartBox.y0, chartBox.width*0.6, chartBox.height])
ax.legend(loc='upper center', bbox_to_anchor=(1.45, 0.8), shadow=True, ncol=1)
plt.show()
```



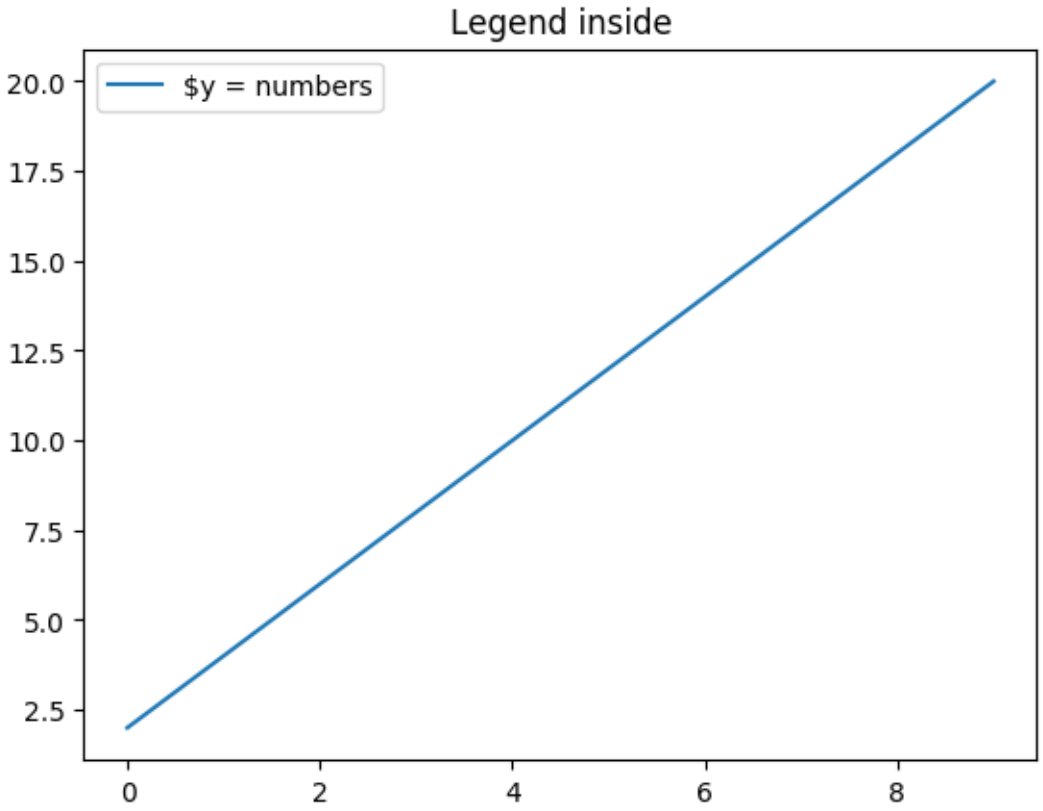
7 Guardar gráficos

Con Matplotlib se pueden guardar gráficos directamente a un archivo usando `savefig()`. El método puede ser usado como sigue

```
In [62]: fig.savefig('plot.png')
```

Ejemplo completo:

```
In [63]: import matplotlib
         import matplotlib.pyplot as plt
         import numpy as np

         y = [2,4,6,8,10,12,14,16,18,20]
         x = np.arange(10)
         fig = plt.figure()
         ax = plt.subplot(111)
         ax.plot(x, y, label='$y = numbers')
         plt.title('Legend inside')
         ax.legend()
         #plt.show()

         fig.savefig('plot.png')
```



Para cambiar el formato, simplemente cambia la extensión del archivo (lo que está después del punto), como sigue:

```
In [64]: fig.savefig('plot.pdf')
```

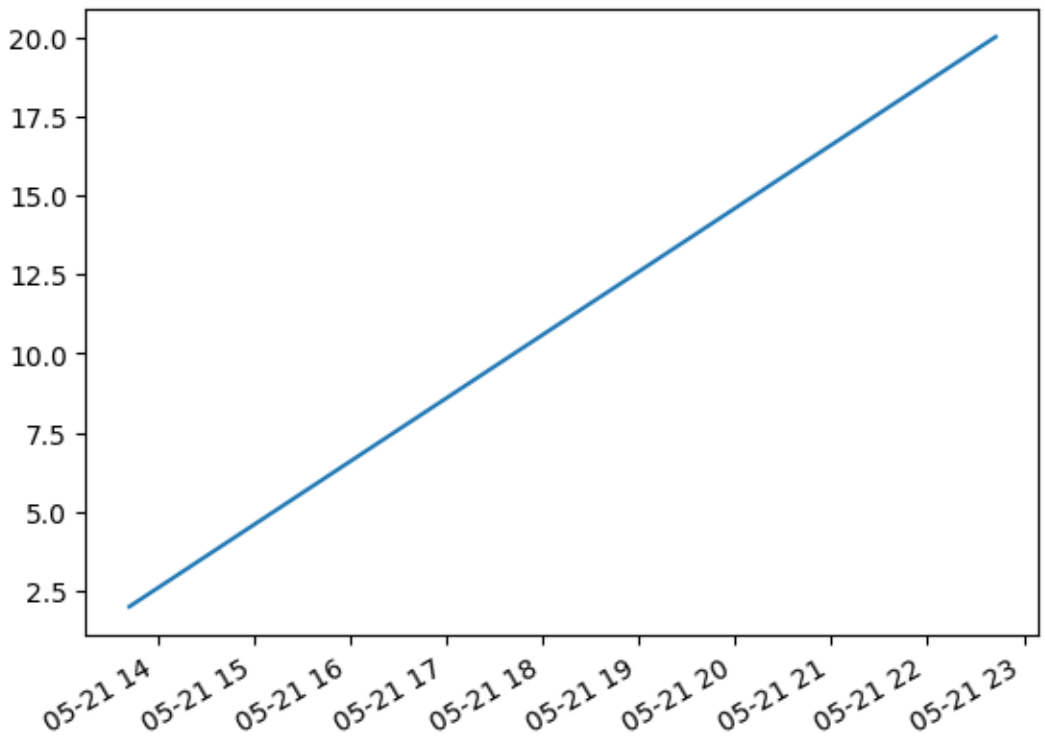
Ahora puedes abrir éste archivo desde tu computador

8 Graficar el tiempo con matplotlib

Matplotlib soporta gráficos con tiempo en el eje x. Los valores de los datos serán puestos en el eje vertical (y). Ahora veremos algunos ejemplos.

8.1 Graficar usando `datetime`

```
In [74]: import matplotlib
         import matplotlib.pyplot as plt
         import numpy as np
         import datetime

         # crear datos
         y = [ 2,4,6,8,10,12,14,16,18,20 ]
         x = [datetime.datetime.now() + datetime.timedelta(hours=i) for i in range(len(y))]

         # graficar
         plt.plot(x,y)
         plt.gcf().autofmt_xdate()
         plt.show()
```



Si quieres cambiar el intervalo usando una de estas líneas de abajo:

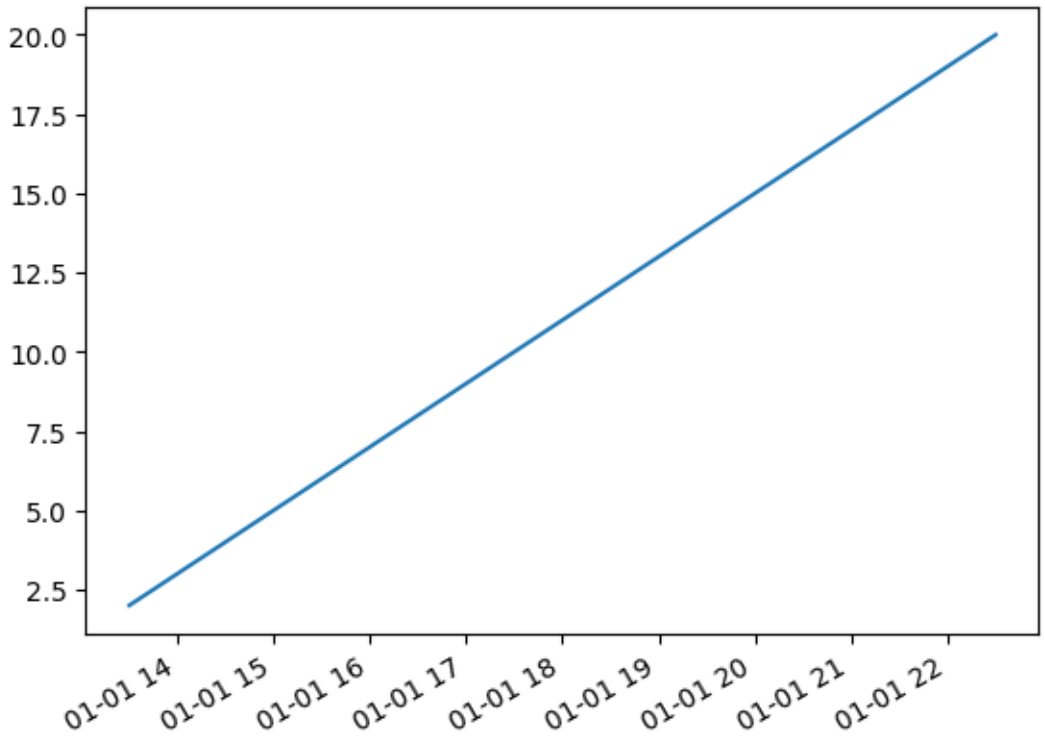
```
In [75]: # minutos
         x = [datetime.datetime.now() + datetime.timedelta(minutes=i) for i in range(len(y))]
```

8.2 Graficar tiempo usando una hora/minuto de forma específica

Para empezar a especificar la fecha, usando nuevamente `datetime` con `datetime.datetime(año, mes, día, hora, minuto)`.

Ejemplo completo:

```
In [76]: import matplotlib
         import matplotlib.pyplot as plt
         import numpy as np
         import datetime

         # create data
         customdate = datetime.datetime(2016, 1, 1, 13, 30)
         y = [ 2,4,6,8,10,12,14,16,18,20 ]
         x = [customdate + datetime.timedelta(hours=i) for i in range(len(y))]

         # plot
         plt.plot(x,y)
         plt.gcf().autofmt_xdate()
         plt.show()
```

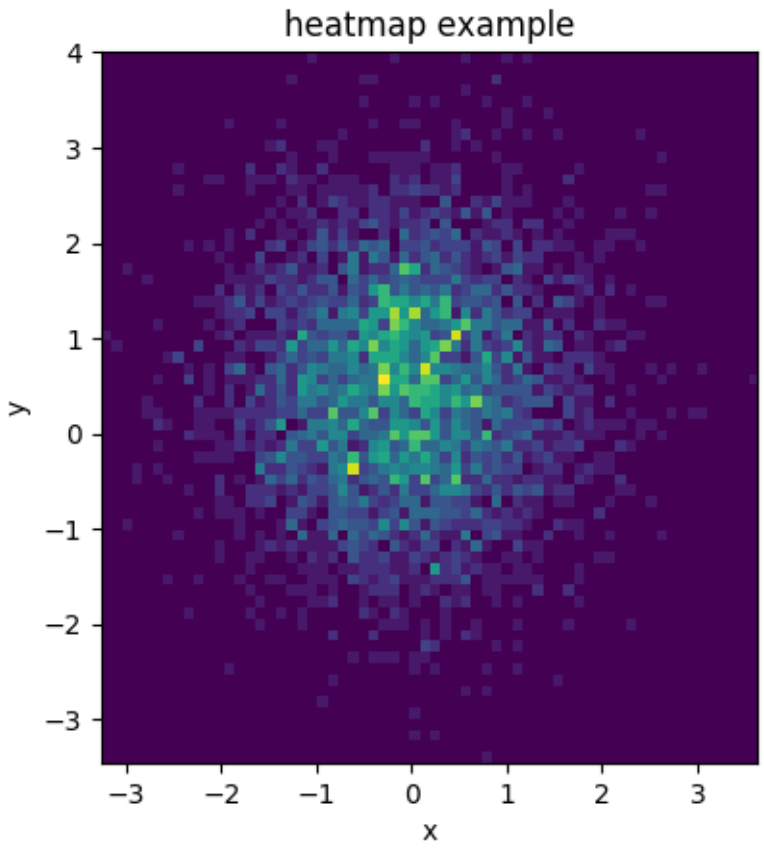


9 Heatmap: mapa de calor en python

La función `histogram2d`puede ser usada para generar un mapa de calor.

Creamos algunos datos aleatorios (x, y) para usarlos en el programa. Establecemos las cajas a 64, tal que el heatmap sea de 64x64. Si quieres otro tamaño, cambia el número de cajas (bins).

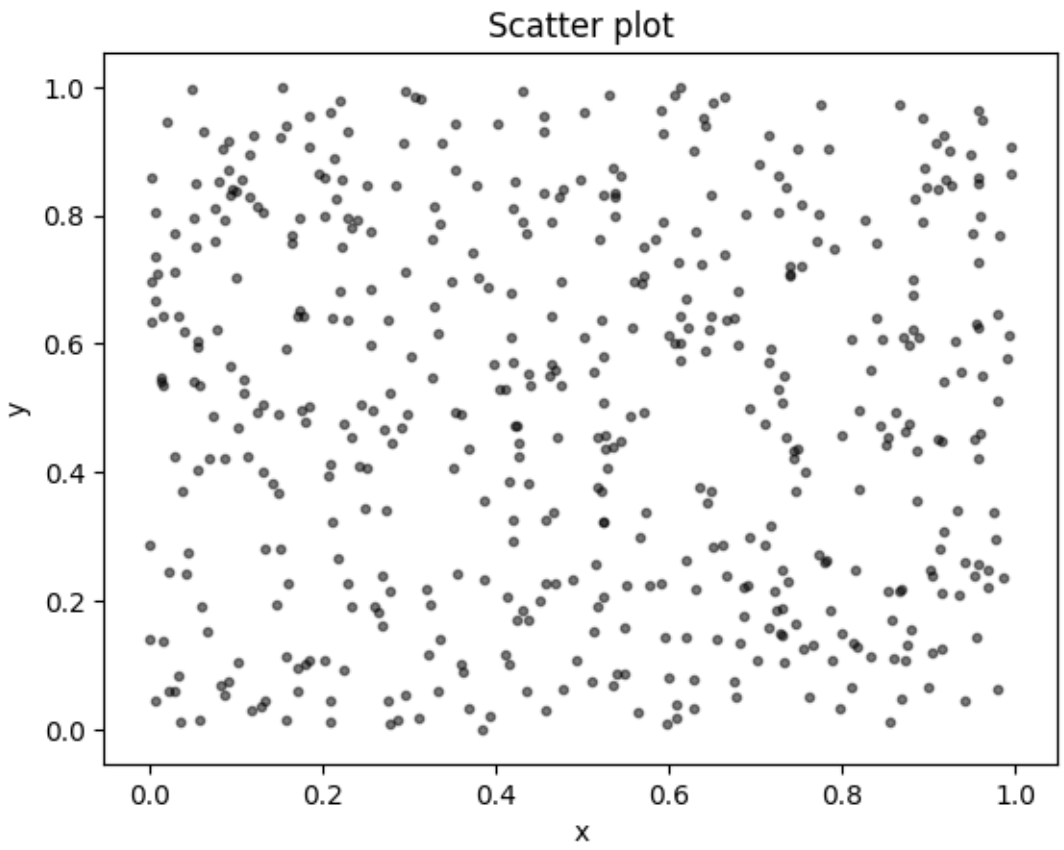
```
In [80]: import numpy as np
         import numpy.random
         import matplotlib.pyplot as plt

         # Create data
         x = np.random.randn(4096)
         y = np.random.randn(4096)

         # Create heatmap
         heatmap, xedges, yedges = np.histogram2d(x, y, bins=(64,64))
         extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]

         # Plot heatmap
         plt.clf()
         plt.title('heatmap example')
         plt.ylabel('y')
         plt.xlabel('x')
         plt.imshow(heatmap, extent=extent)
         plt.show()
```



10 Gráficos de dispersión

Matplotlib tiene una función para crear gráficos de dispersión, o en inglés, *scatterplots*, llamada `scatter()`. Un gráfico de dispersión es un tipo de gráfico que muestra los datos como colección de puntos en el espacio. La posición de un punto depende de su valor 2-dimensional. donde cada valor es una posición en el eje horizontal y vertical.

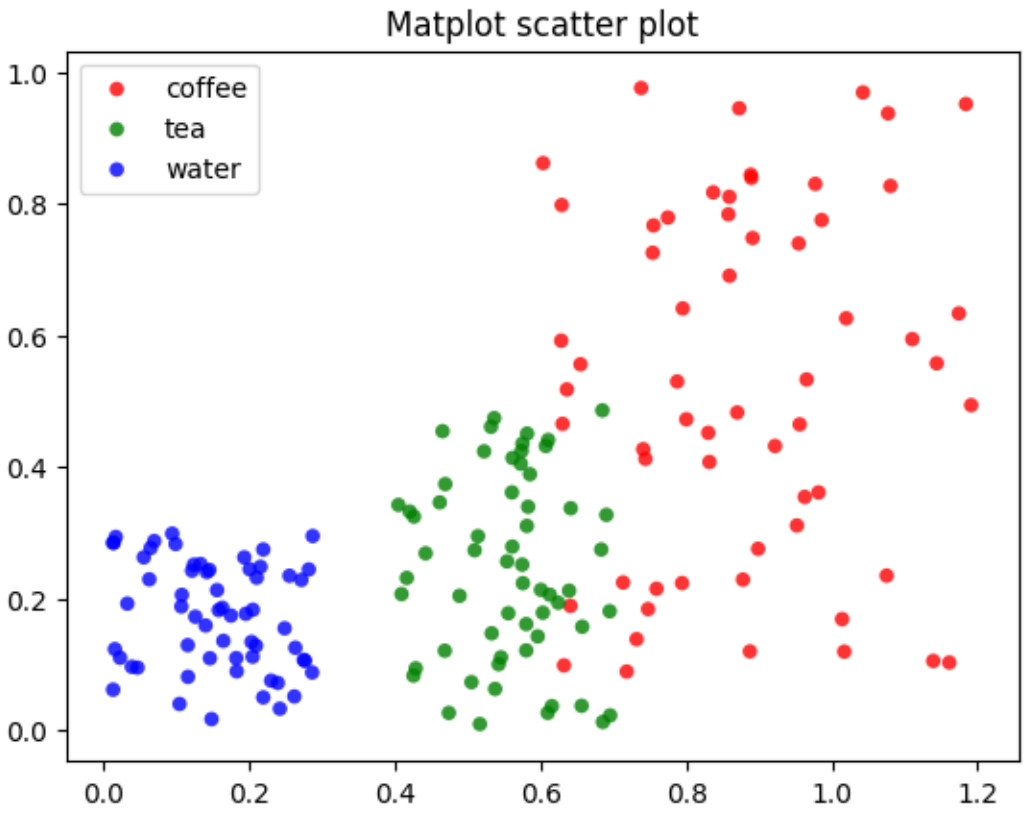
```
In [94]: import numpy as np
         import matplotlib.pyplot as plt

         # Crear los datos
         N = 500
         x = np.random.rand(N)
         y = np.random.rand(N)
         colors = (0,0,0)
         area = np.pi*3

         # Graficar
         plt.scatter(x, y, s=area, c=[colors], alpha=0.5)
         plt.title('Scatter plot')
         plt.xlabel('x')
         plt.ylabel('y')
         plt.show()
```



Los datos pueden ser clasificados en varios grupos. El siguiente código muestra cómo:

```
In [105]: import numpy as np
          import matplotlib.pyplot as plt

          # Create data
          N = 60
          g1 = (0.6 + 0.6 * np.random.rand(N), np.random.rand(N))
          g2 = (0.4+0.3 * np.random.rand(N), 0.5*np.random.rand(N))
          g3 = (0.3*np.random.rand(N),0.3*np.random.rand(N))

          data = (g1, g2, g3)
          colors = ("red", "green", "blue")
          groups = ("coffee", "tea", "water")

          # Create plot
          fig = plt.figure()
          ax = fig.add_subplot(1, 1, 1, facecolor="1.0")

          for data, color, group in zip(data, colors, groups):
              x, y = data
```

```
    ax.scatter(x, y, alpha=0.8, c=color, edgecolors='none', s=30, label=group)

plt.title('Matplot scatter plot')
plt.legend(loc=2)
plt.show()
```



11 Subgráficos de matplotlib

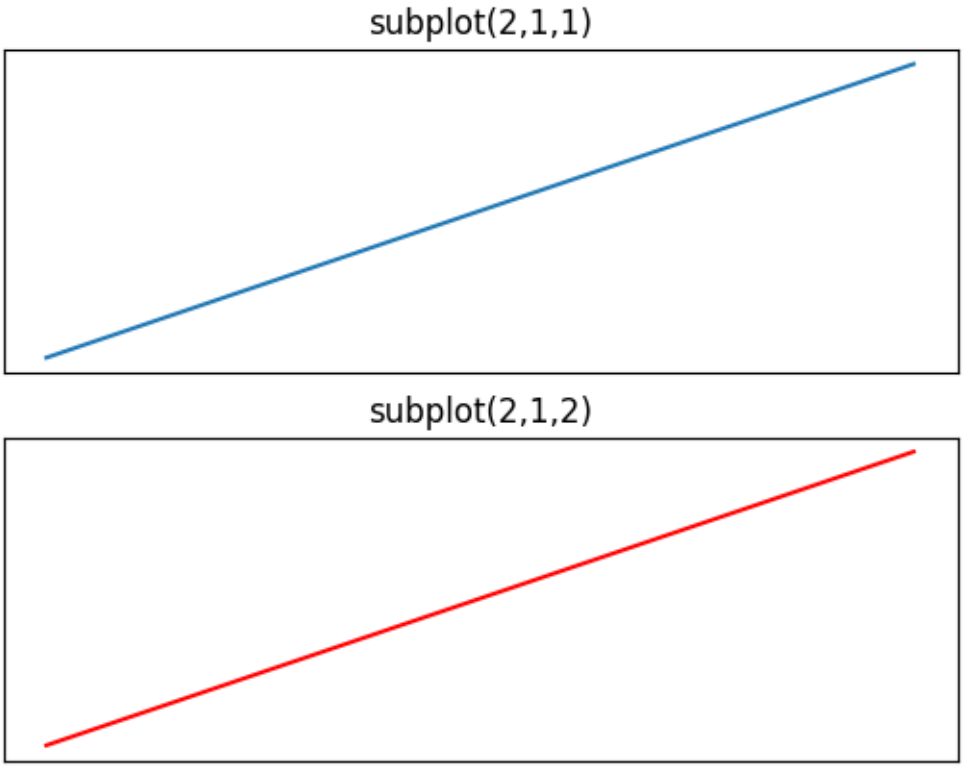
La función `subplot()` puede ser llamada para graficar dos o más gráficos en una figura. la librería soporta todo tipo de sub-gráficos incluyendo horizontales 2x1, verticales 2x1 o cuadriculados de 2x2.

11.1 Sub-gráfico horizontal

Fíjate en el uso de la función `subplot()` en el siguiente ejemplo:

```
In [106]: from pylab import *

          t = arange(0.0, 20.0, 1)
          s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
```

```
subplot(2,1,1)
xticks([]), yticks([])
title('subplot(2,1,1)')
plot(t,s)

subplot(2,1,2)
xticks([]), yticks([])
title('subplot(2,1,2)')
plot(t,s,'r-')

show()
```

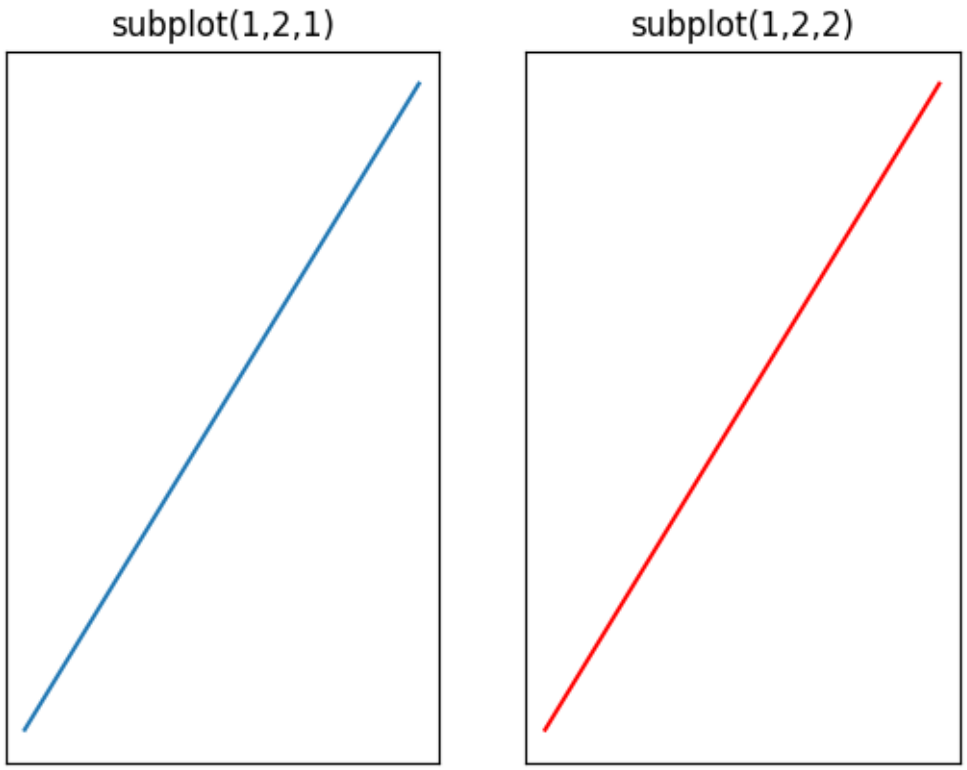


11.2 Sub-gráfico vertical

Cambiando los parámetros de subplot podemos crear un gráfico vertical

```
In [107]: from pylab import *

          t = arange(0.0, 20.0, 1)
          s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

          subplot(1,2,1)
```

```
xticks([]), yticks([])
title('subplot(1,2,1)')
plot(t,s)

subplot(1,2,2)
xticks([]), yticks([])
title('subplot(1,2,2)')
plot(t,s,'r-')

show()
```

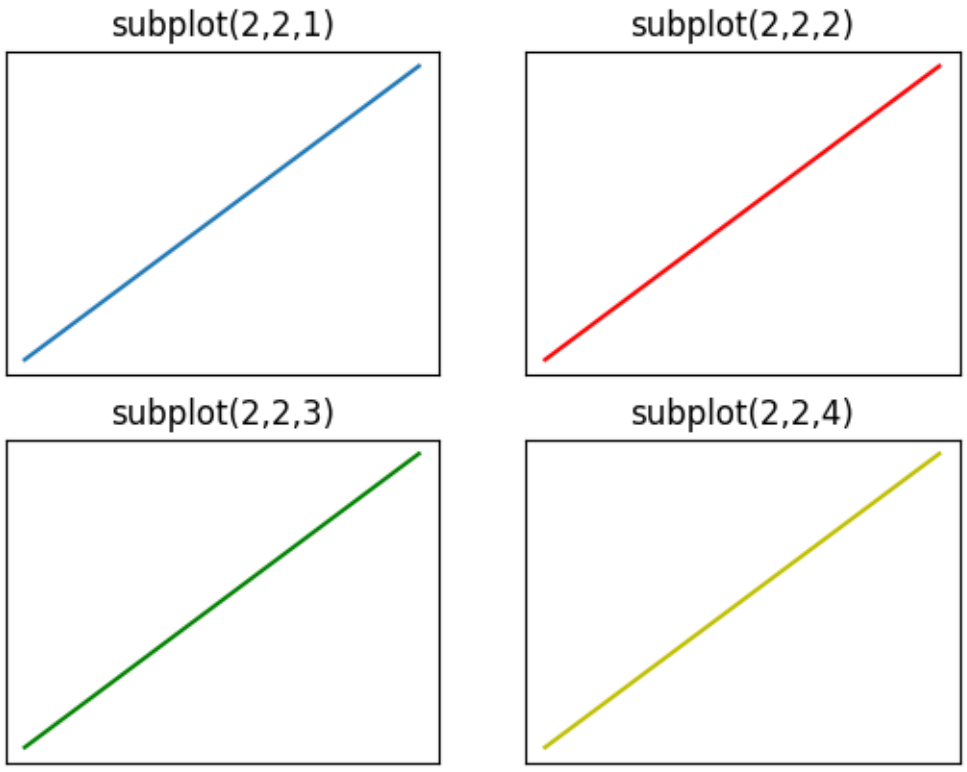


11.3 Cuadrícula de subgráficos

Para crear una cuadrícula de gráficos de2x2, usa el siguiente código:

```
In [108]: from pylab import *

          t = arange(0.0, 20.0, 1)
          s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

          subplot(2,2,1)
          xticks([]), yticks([])
```

```
title('subplot(2,2,1)')
plot(t,s)

subplot(2,2,2)
xticks([]), yticks([])
title('subplot(2,2,2)')
plot(t,s,'r-')

subplot(2,2,3)
xticks([]), yticks([])
title('subplot(2,2,3)')
plot(t,s,'g-')

subplot(2,2,4)
xticks([]), yticks([])
title('subplot(2,2,4)')
plot(t,s,'y-')

show()
```



12 Matriz de correlación

Un diagrama de correlación puede ser creado usando matplotlib.

12.1 Definición de la matriz

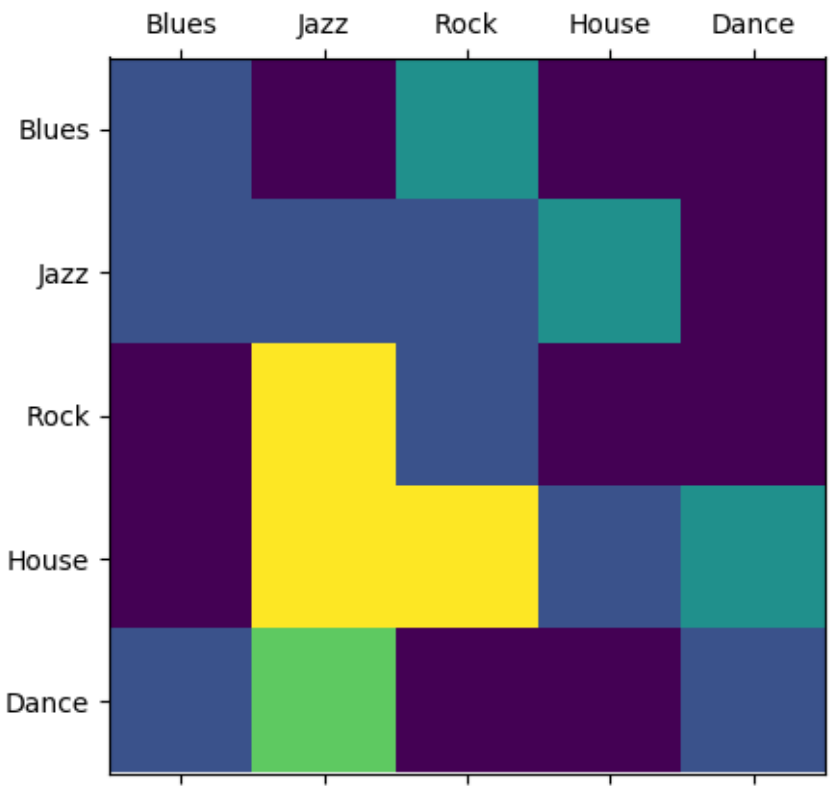
Para empezar, definimos una matriz 2x2 y una lista llamada groups. La matriz es definida dentro de los corchetes:

```
In [109]: m = [
          [1,0,2,0,0],
          [1,1,1,2,0],
          [0,4,1,0,0],
          [0,4,4,1,2],
          [1,3,0,0,1],
          ]

          groups = ['Blues','Jazz','Rock','House','Dance']
```

12.2 Matriz de correlación

El siguiente código genera un diagrama de matriz de correlación:

```
In [110]: import matplotlib.pyplot as plt
          import numpy as np

          m = [
          [1,0,2,0,0],
          [1,1,1,2,0],
          [0,4,1,0,0],
          [0,4,4,1,2],
          [1,3,0,0,1],
          ]

          plt.matshow(m)

          groups = ['Blues','Jazz','Rock','House','Dance']

          x_pos = np.arange(len(groups))
          plt.xticks(x_pos,groups)

          y_pos = np.arange(len(groups))
          plt.yticks(y_pos,groups)

          plt.show()
```



Inicialmente definimos una matriz (`m`) y la lista (`groups`). Definimos que el largo será igual al largo de los grupos. En el eje x e y definimos el nombre de los grupos.