Matplotlib

1. Matplotlib 的概要

何謂 Matplotlib

Matplotlib 是 Python 繪製平面圖表的主要函示庫， 相容於各種 OS，與 Jupyter Notebook的相容性極高，在Notebook執行程式碼，就能在Notebook繪製圖表，適合在需以視覺效果呈現資料時使用

Matplotlib 用來繪製圖表的程式碼有兩種風格。MATLAB風格與物件導向風格。

要使用Matplotlib必須先如下匯入函式庫。可使用as關鍵字再以plt呼叫

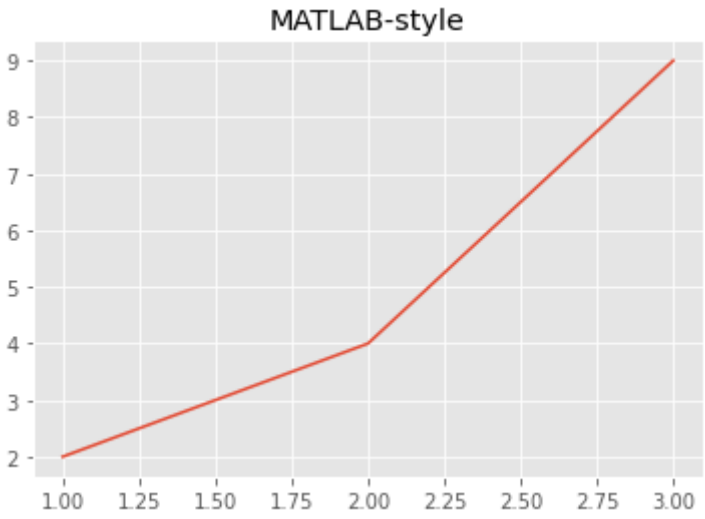
```
import matplotlib.pyplot as plt
```

```
import matplotlib.style

#指定ggplot樣式
matplotlib.style.use('ggplot')
```

MATLAB 風格

MATLAB風格是以類似數值剖析軟體MATLAB相似的格式繪製圖表的方法。此風格會如下對matplotlib.pyplot模型執行各種繪製圖表的函數。

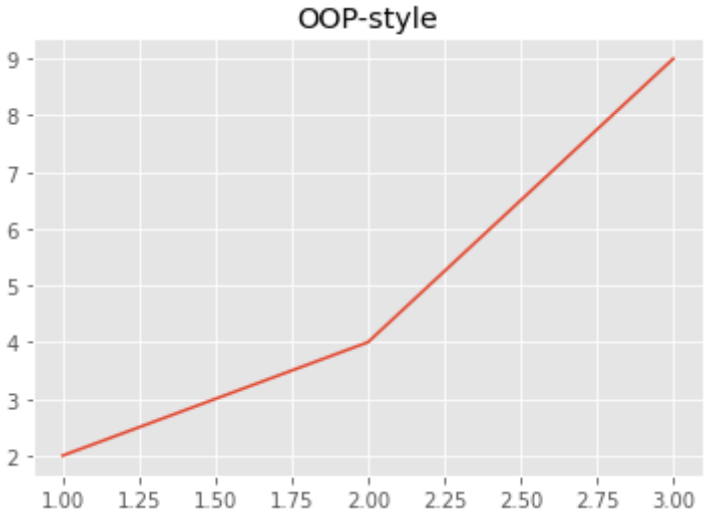
```
#建立資料
x = [1,2,3]
y = [2,4,9]

#繪製折線圖
plt.plot(x,y)

#設定圖表標題
plt.title('MATLAB-style')

#顯示圖表
plt.show()
```



物件導向風格

與MATLAB風格不同，是對繪圖物件新增subplot，再對subplot繪製圖表。物件導向風格可對一個figure物件指定多個subplot，所以可同時顯示多個圖表

```
#建立資料
x = [1,2,3]
y = [2,4,9]

#建立繪圖物件 (fig)與subplot(ax)
fig, ax = plt.subplots()

#繪製折線圖
ax.plot(x,y)

#設定圖表標題
ax.set_title('OOP-style')

#顯示圖表
plt.show()
```



2. 繪圖物件

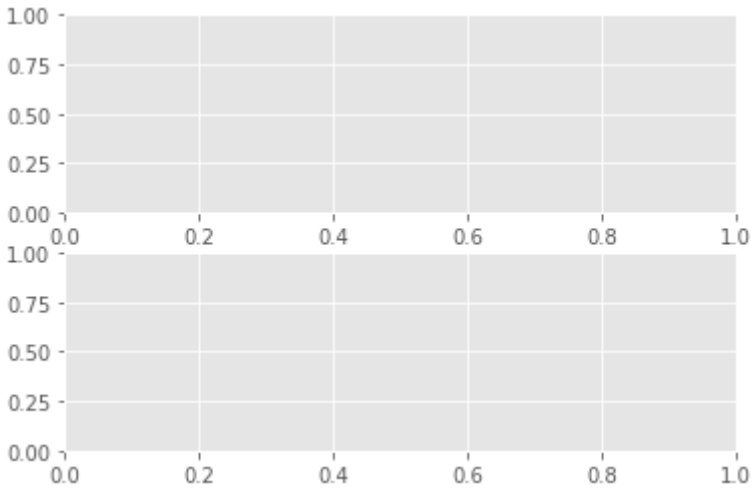
繪圖物件與subplot

利用Matplotlib繪製圖表，可先建立繪圖物件 (figure)，再於繪圖物件配置一個以上的subplot。上述的fig , ax=plt.subplots()程式碼就是

先建立一個繪圖物件，再於繪圖物件配置一個subplot，然後將這兩個部分分別儲存再fig與ax的變數。

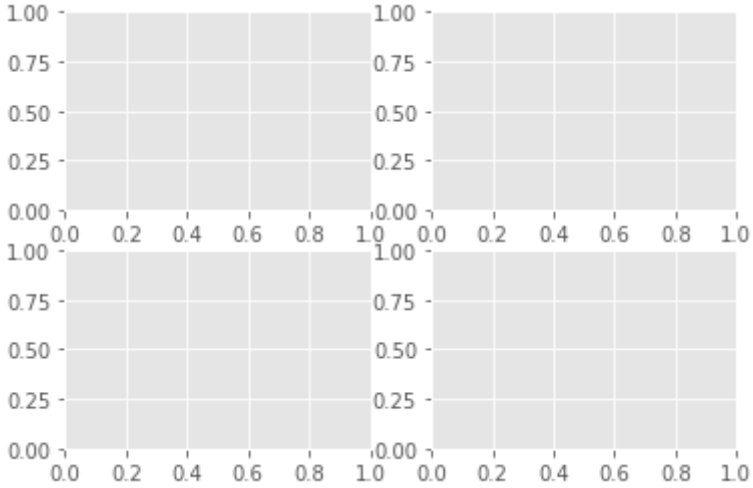
subplots函數的參數若指定為數值。就可以在單一的繪圖物件配置多個subplot。

```
import matplotlib.pyplot as plt

#配置兩個subplot
fig, axes = plt.subplots(2)
plt.show()
```



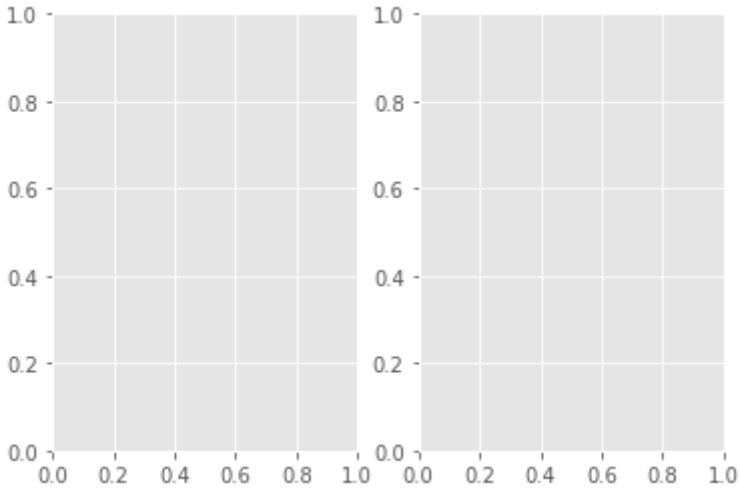
若指定為subplots(2,2)就能配置2列2行

```
#配置2列2行的subplot行的subplot
fig, axes = plt.subplots(2,2)
plt.show()
```



參數還可以利用關鍵字參數nrows，ncols指定

```
#配置1列2行的subplot
fig, axes = plt.subplots(ncols=2)
plt.show()
```

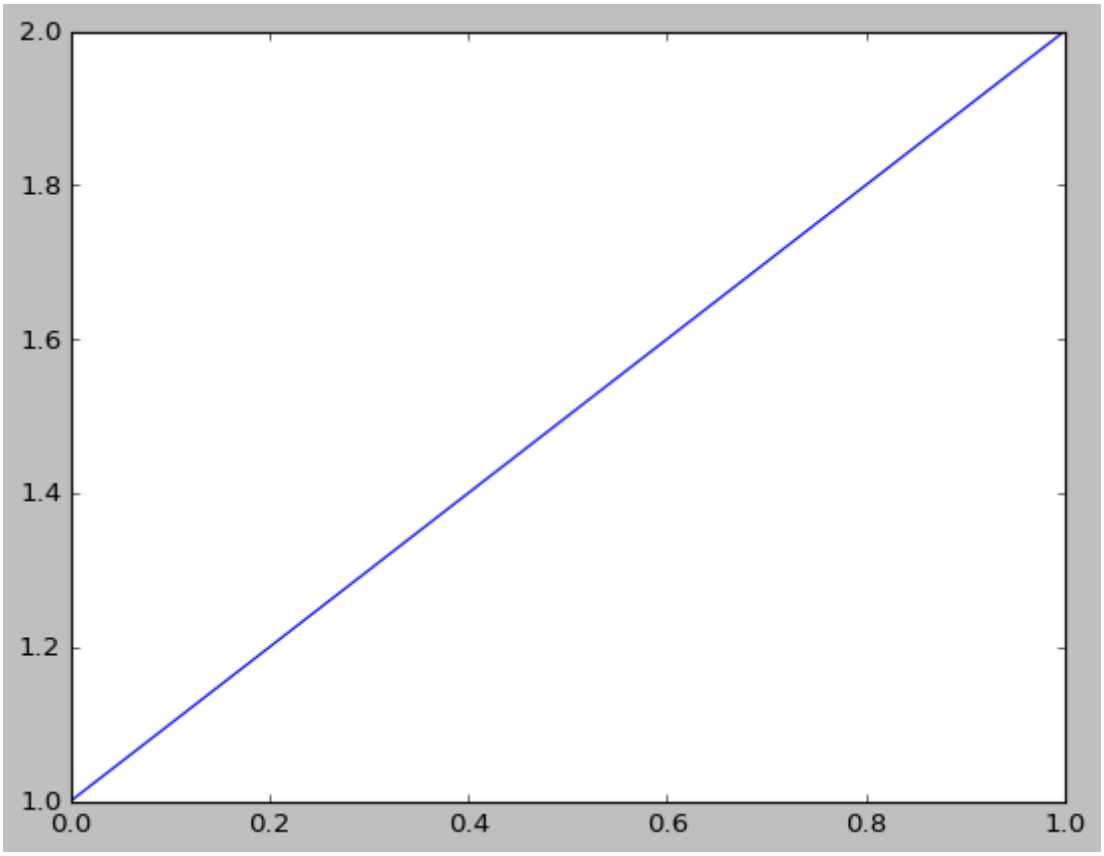


圖表樣式

matplotlib.style可指定圖表的樣式(線條顏色，線條粗細與背景顏色)。可使用的樣式名稱可利用matplotlib.style.available取得。Matplotlib 2.2.2 總共內建了26種圖表樣式。要套用圖表樣式可將樣式名稱的字串指定給matplotlib.style.use()

```
import matplotlib.style

#顯示圖表樣式清單
print(matplotlib.style.available)
```

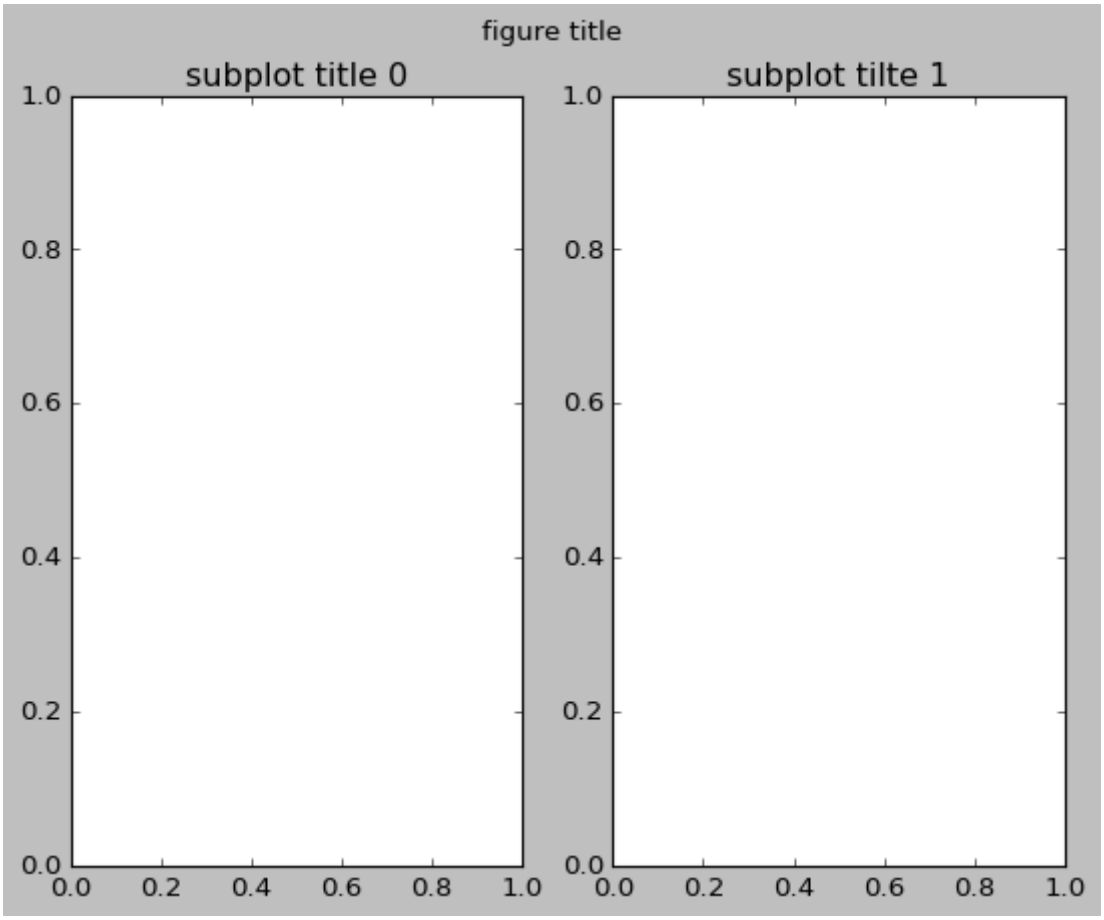
```
#指定為classic圖表樣式
matplotlib.style.use('classic')

fig, ax = plt.subplots()
ax.plot([1,2])

plt.show()
```



圖表標題

也可以替繪圖物件與subplot指定圖表標題

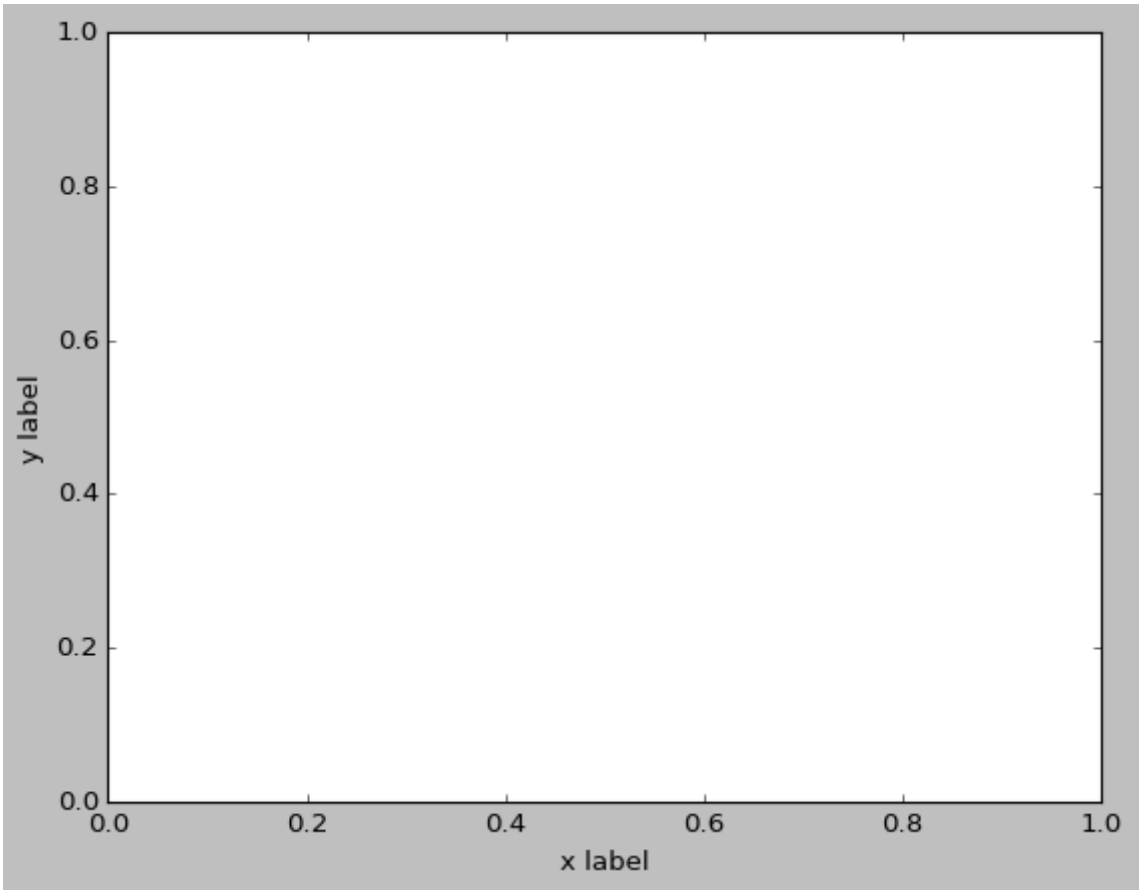
```
fig, axes = plt.subplots(ncols=2)

#替subplot設定圖表標題
axes[0].set_title('subplot title 0')
axes[1].set_title('subplot tilte 1')

#替繪圖物件設定圖表標題
fig.suptitle('figure title')

plt.show()
```



坐標軸標籤

也可以在圖表的座標軸指定標籤

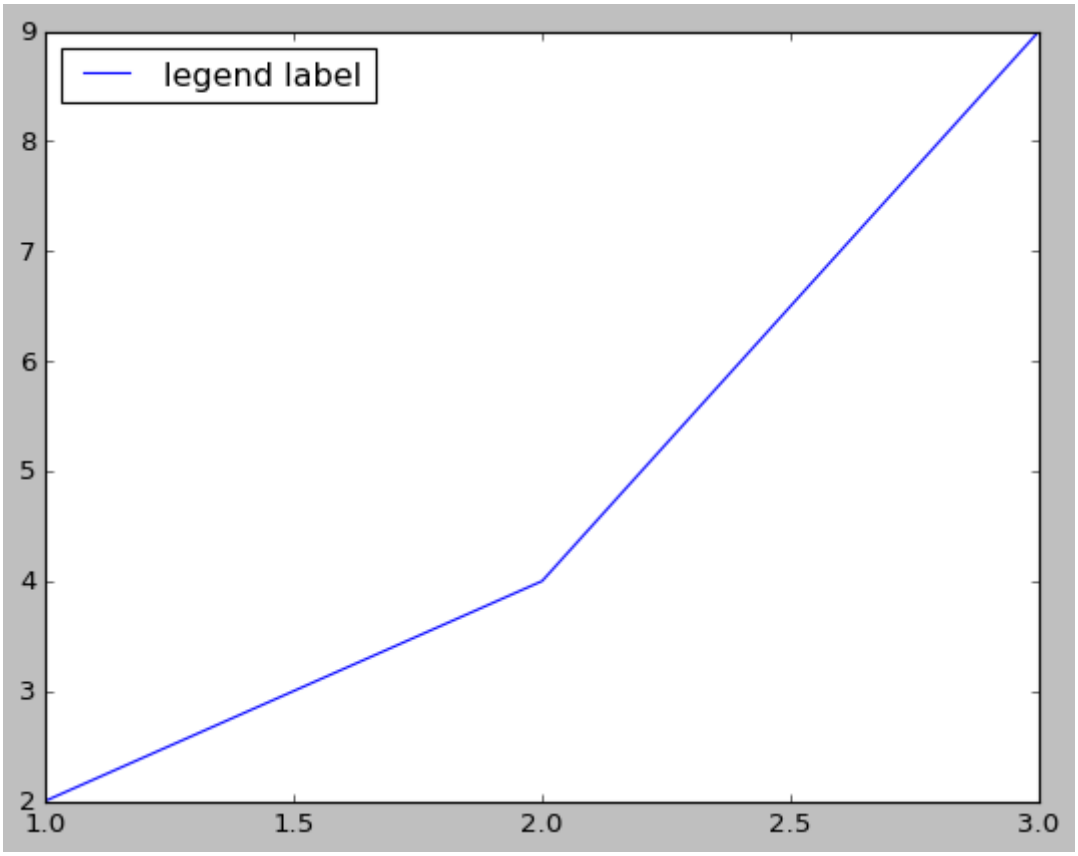
```
fig, ax = plt.subplots()

#在x軸指定標籤
ax.set_xlabel('x label')

#在y軸指定標籤
ax.set_ylabel('y label')

plt.show()
```



圖例

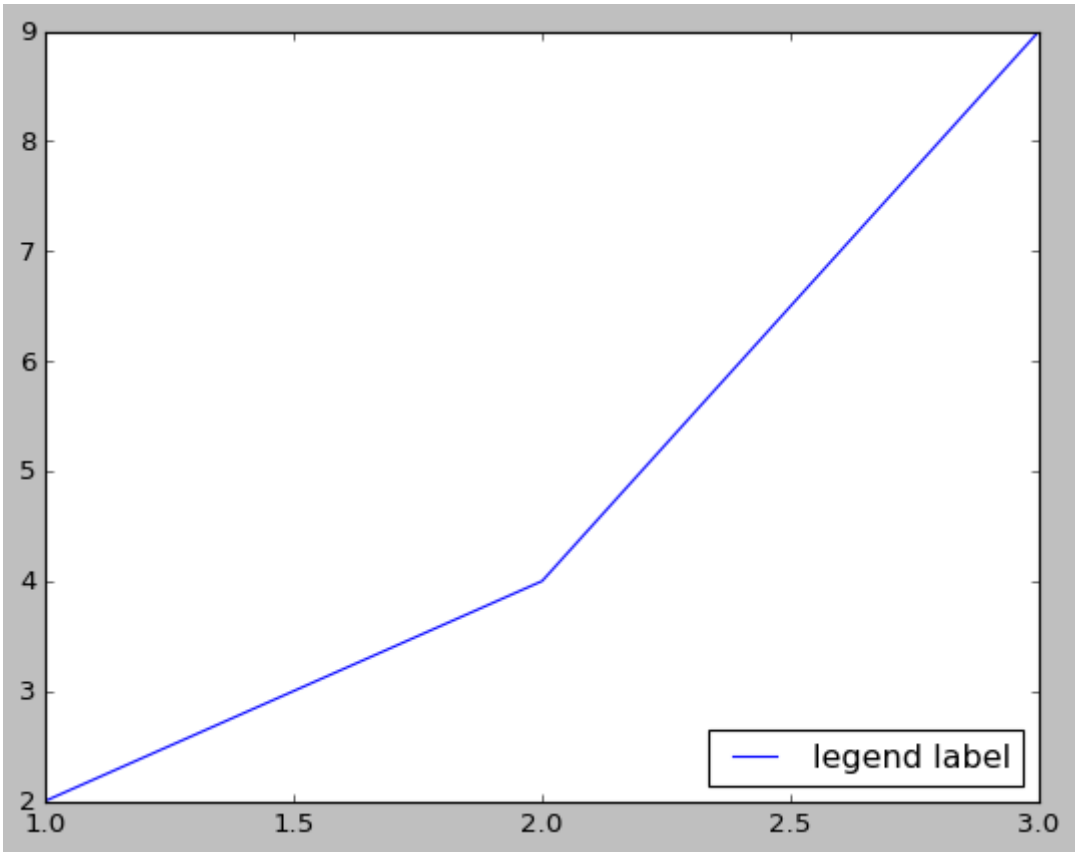
可在subplot顯示圖例。在繪製資料時，將圖例的標籤指定給label參數，就能利用legend方法顯示圖例。指定為loc='best'，則可在與資料重疊最小之處輸出圖例。

```
fig, ax = plt.subplots()

#設定圖利標籤
ax.plot([1,2,3], [2,4,9],label='legend label')

#顯示圖例
ax.legend(loc='best')

plt.show()
```



上述的範例式於左上角顯示圖例。若希望調整圖例的位置，可在loc參數設定位置。下面的範例是於右下角輸出圖例。

```
fig, ax = plt.subplots()
ax.plot([1,2,3],[2,4,9],label='legend label')

#於圖表右下角顯示圖例
ax.legend(loc='lower right')

plt.show()
```



- 圖例的位置共有十種可以指定，其中包含 'upper left'、'center'、'center left'、'lower center'、'best'。若希望圖例在subplot的外側顯示，可利用bbox_to_anchor參數指定座標

輸出檔案

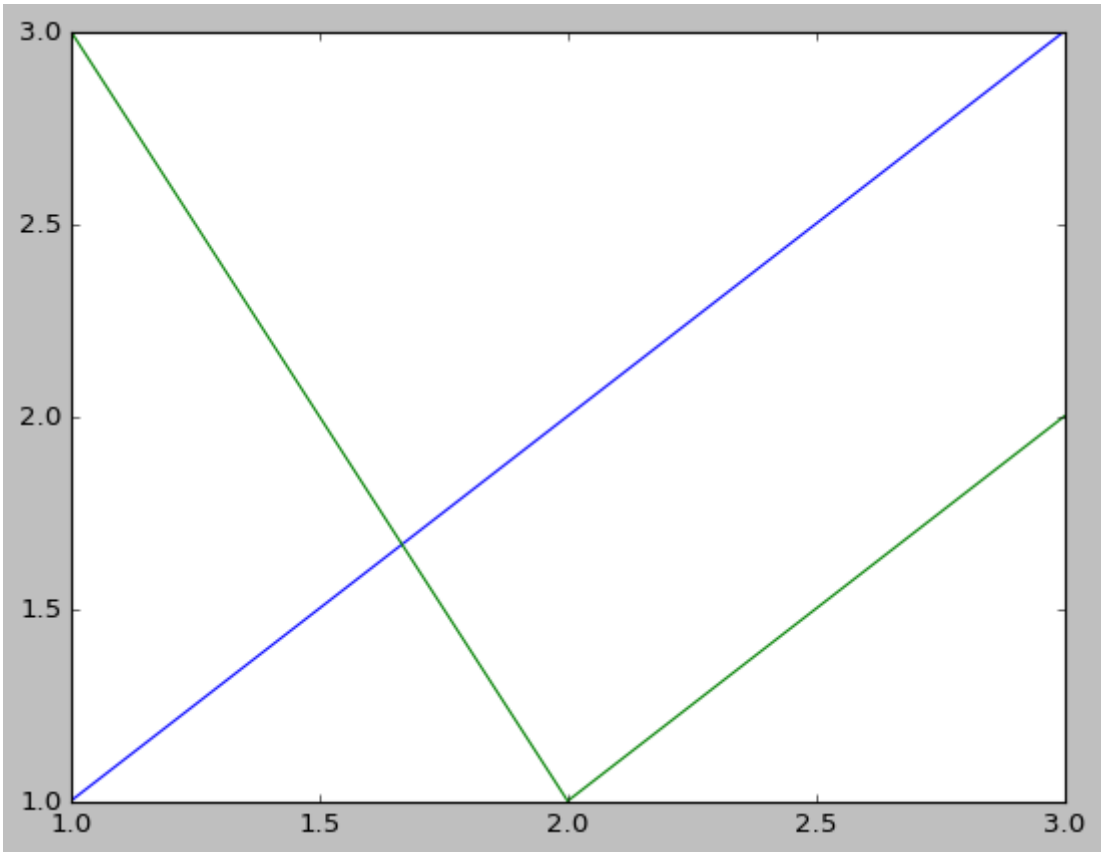
圖表可利用savefig方法轉存為檔案。可選用的檔案格式包含 png、pdf、ps、eps、svg，會根據附檔名自動決定檔案格式(也可利用format參數指定)

```
fig,ax = plt.subplots()
ax.set_title('subplot title')

#以png格式儲存
fig.savefig('sample-figure.png')

#以vg格式儲存
fig.savefig('sample-figure.svg')
```

3. 圖表種類與輸出方法

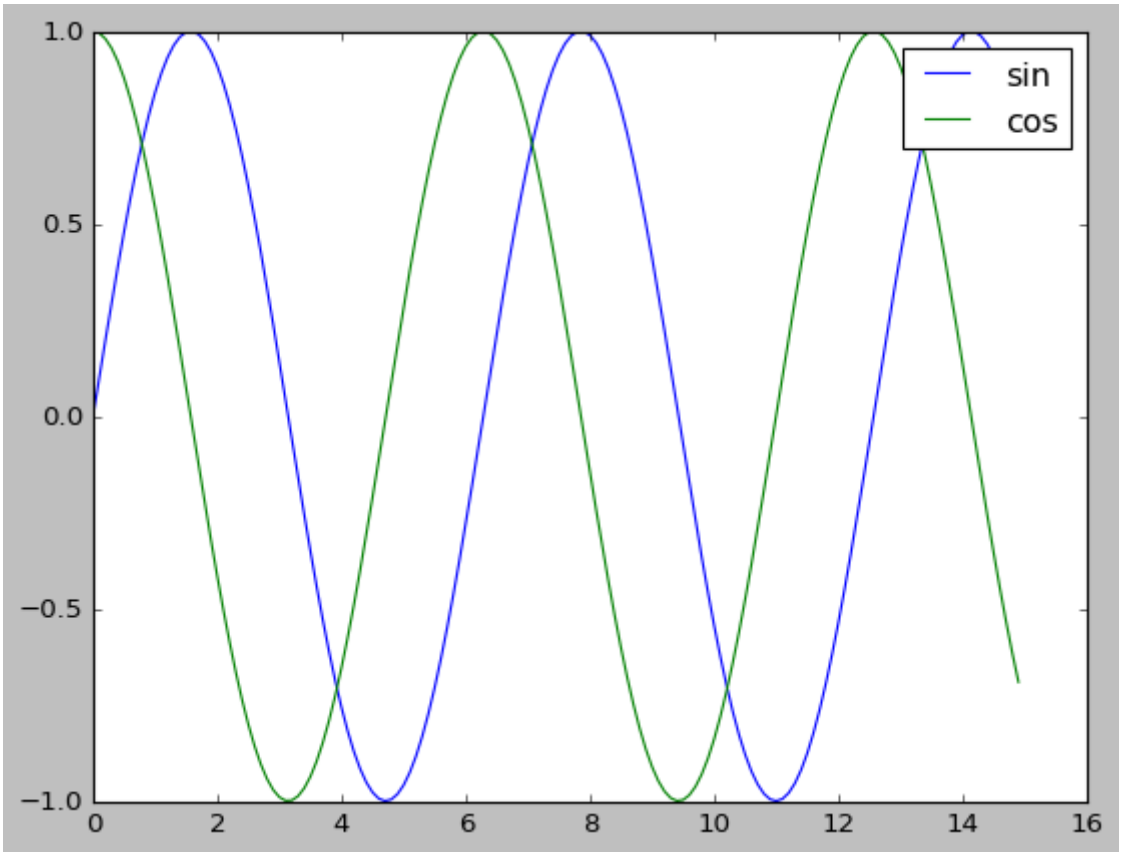
折線圖

```
fig, ax = plt.subplots()

x = [1,2,3]
y1 = [1,2,3]
y2 = [3,1,2]

#繪製折線圖
ax.plot(x,y1)
ax.plot(x,y2)

plt.show()
```



雖是折線圖，但只要讓值的間隔變小，就能繪製出類似曲線圖的圖表。下列是繪製sin、cos圖表範例

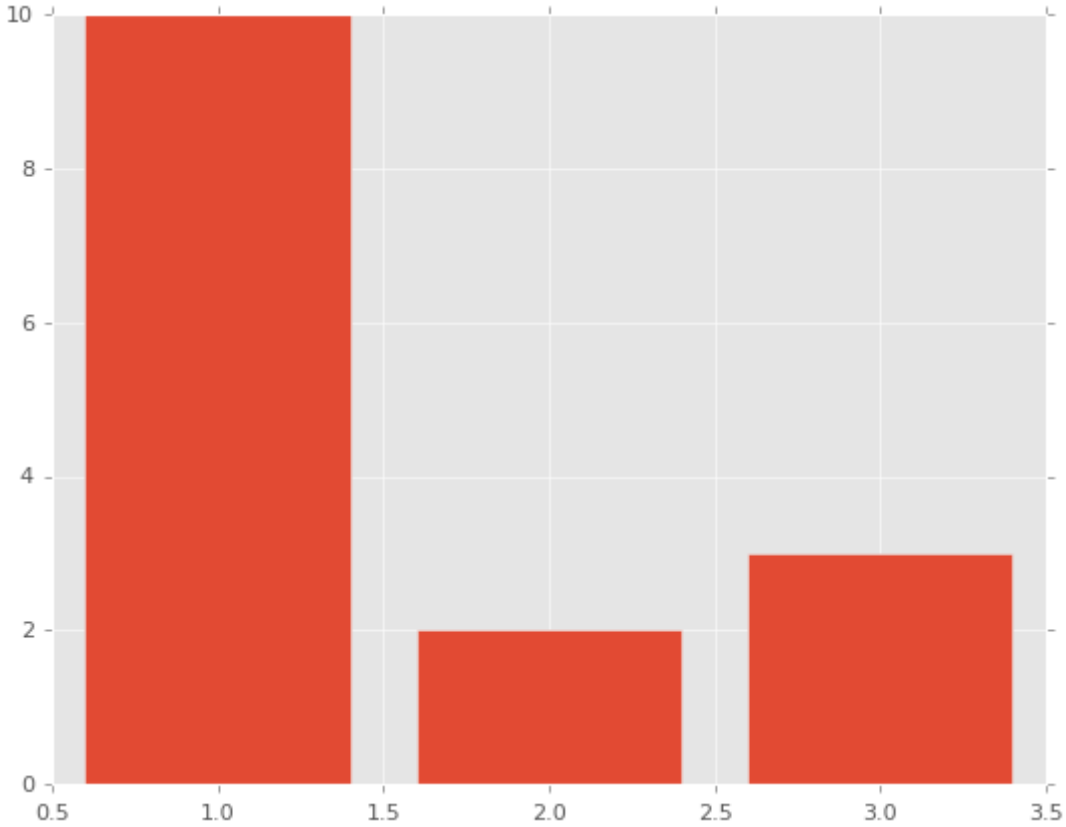
```
import numpy as np

x = np.arange(0.0, 15.0, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

fig, ax = plt.subplots()
ax.plot(x, y1, label='sin')
ax.plot(x, y2, label='cos')
ax.legend()

plt.show()
```



長條圖

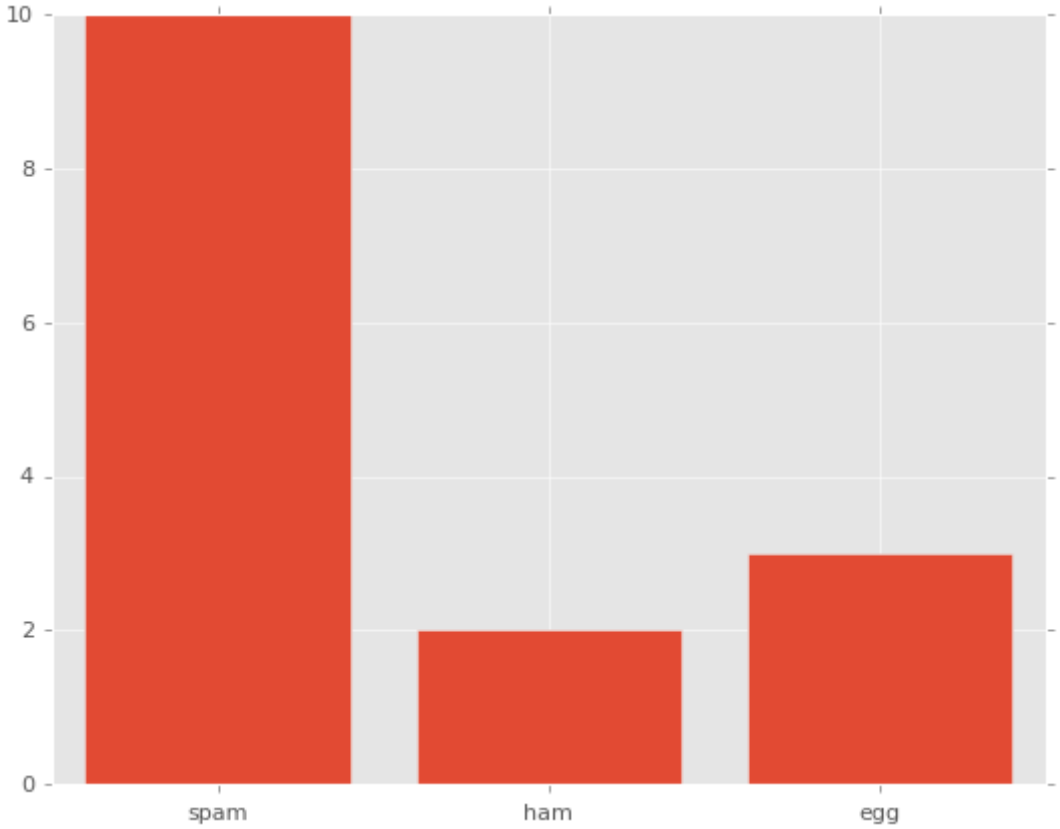
長條圖是以bar方法繪製

```
fig, ax = plt.subplots()

x = [1,2,3]
y = [10,2,3]

#繪製條狀圖
ax.bar(x,y)

plt.show()
```



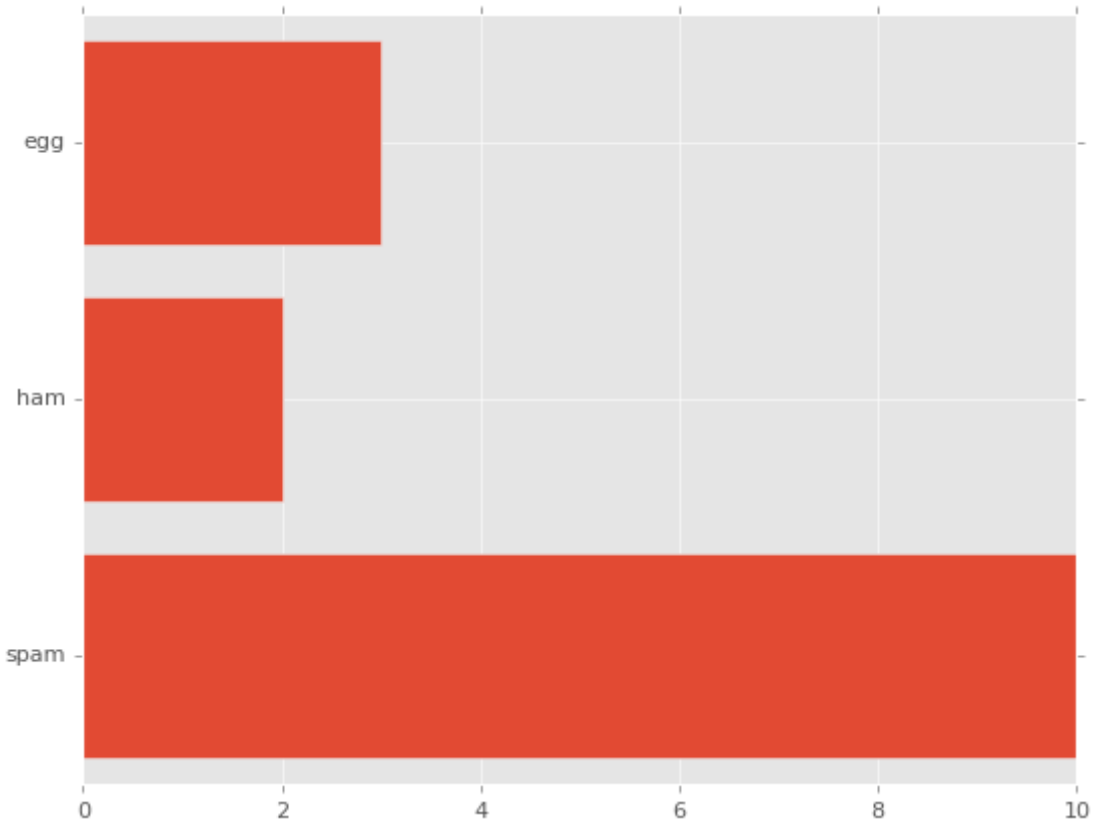
bar 方法的 tick_label參數可於刻度指定標籤

```
fig, ax = plt.subplots()

x = [1,2,3]
y = [10,2,3]
labels = ['spam','ham','egg']

#指定標籤
ax.bar(x,y,tick_label=labels)

plt.show()
```



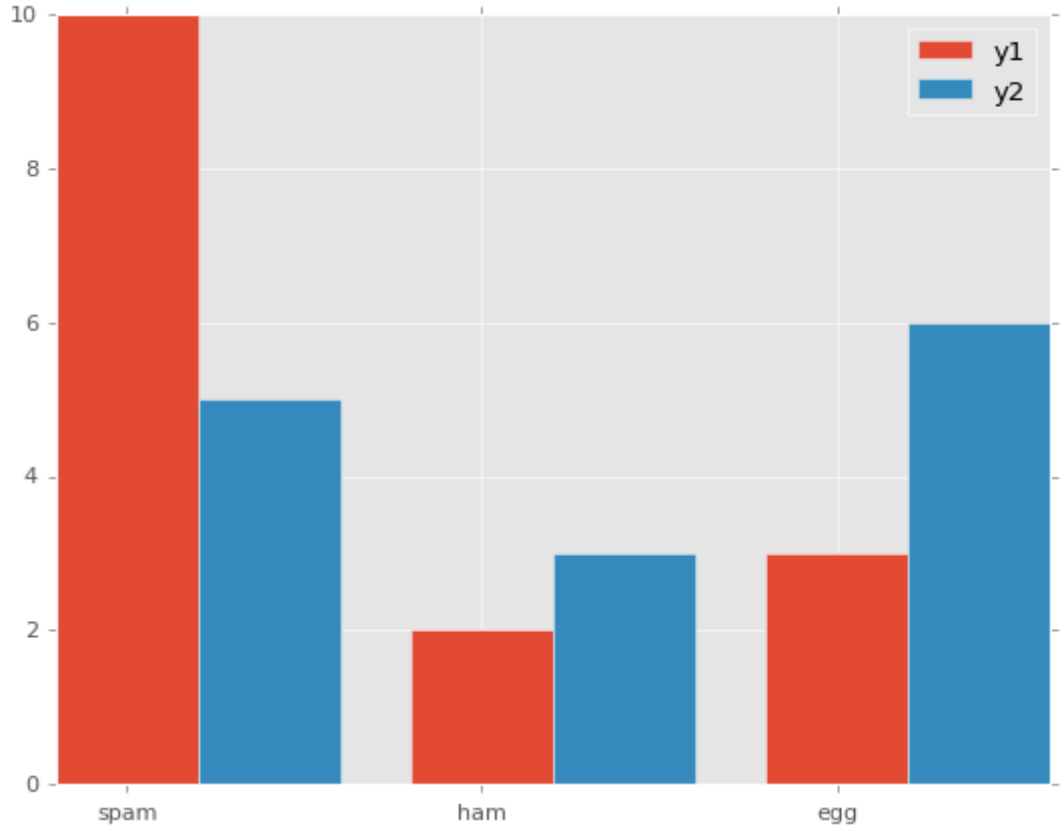
若想繪製橫條圖可使用barh方法。基本的使用方法與bar方法一樣，將前一段程式碼的bar部分換成barh

```
fig, ax = plt.subplots()

x = [1,2,3]
y = [10,2,3]
labels = ['spam','ham','egg']

#指定標籤
ax.barh(x,y,tick_label=labels)

plt.show()
```



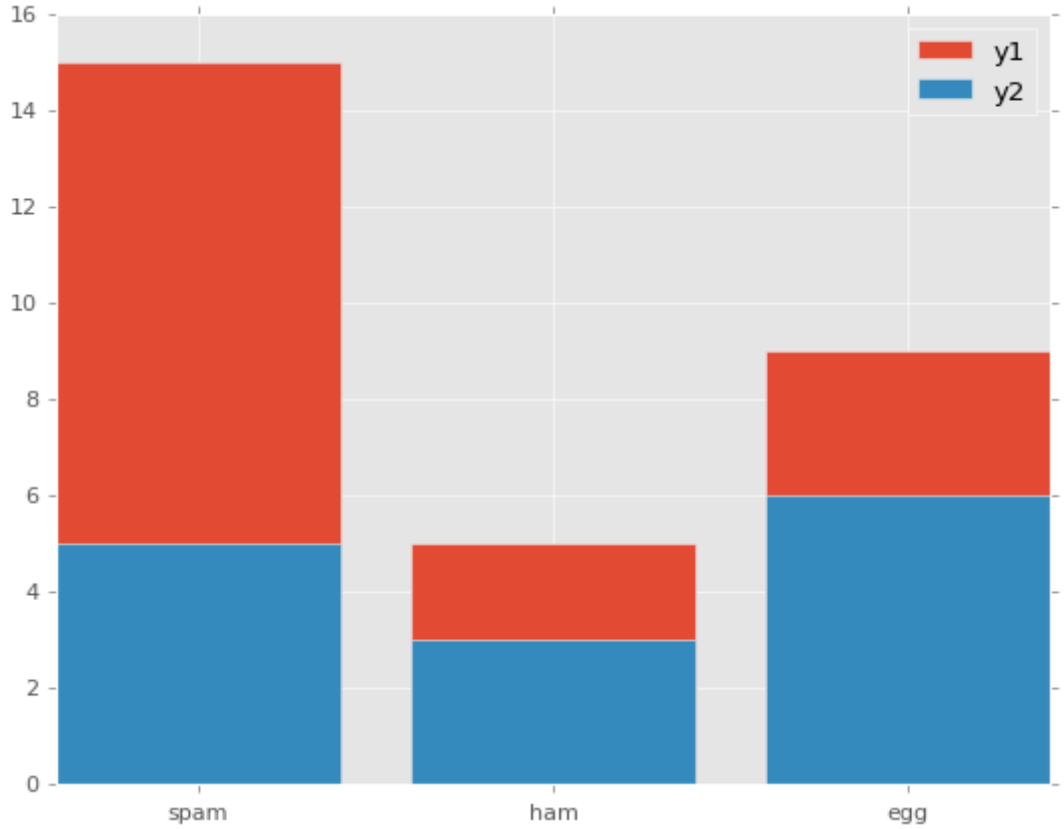
若想讓多個直條圖同時顯示，必須自行指定長條圖的寬度，也必須自行移動長條圖的位置。

```
fig, ax = plt.subplots()

x = [1,2,3]
y1 = [10,2,3]
y2 = [5,3,6]
labels = ['spam','ham','egg']

#將長條圖的寬度設定為0.4
width = 0.4

#指定寬度
ax.bar(x,y1,width=width, tick_label = labels,label='y1')

#錯開長條位置
x2 = [num + width for num in x]
ax.bar(x2, y2, width=width, label='y2')

ax.legend()

plt.show()
```



繪製堆疊直條圖的方法

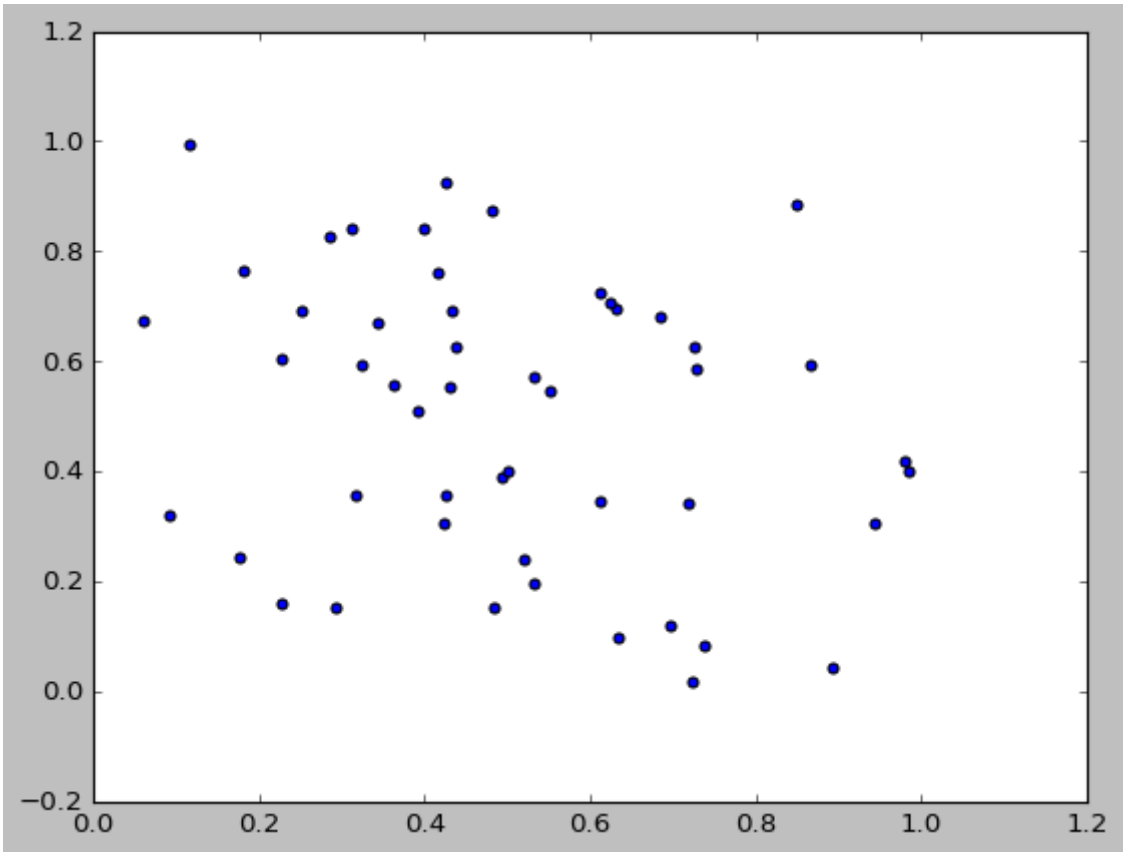
```
fig, ax = plt.subplots()

x = [1,2,3]
y1 = [10,2,3]
y2 = [5,3,6]
labels = ['spam','ham','egg']

#儲存y1與y2的加總值
y_total = [num1 + num2 for num1, num2 in zip(y1,y2)]

#以y1與y2的加總值繪製高度對應的長條圖
ax.bar(x, y_total, tick_label = labels, label ='y1')
ax.bar(x, y2, label='y2')
ax.legend()

plt.show()
```



散佈圖

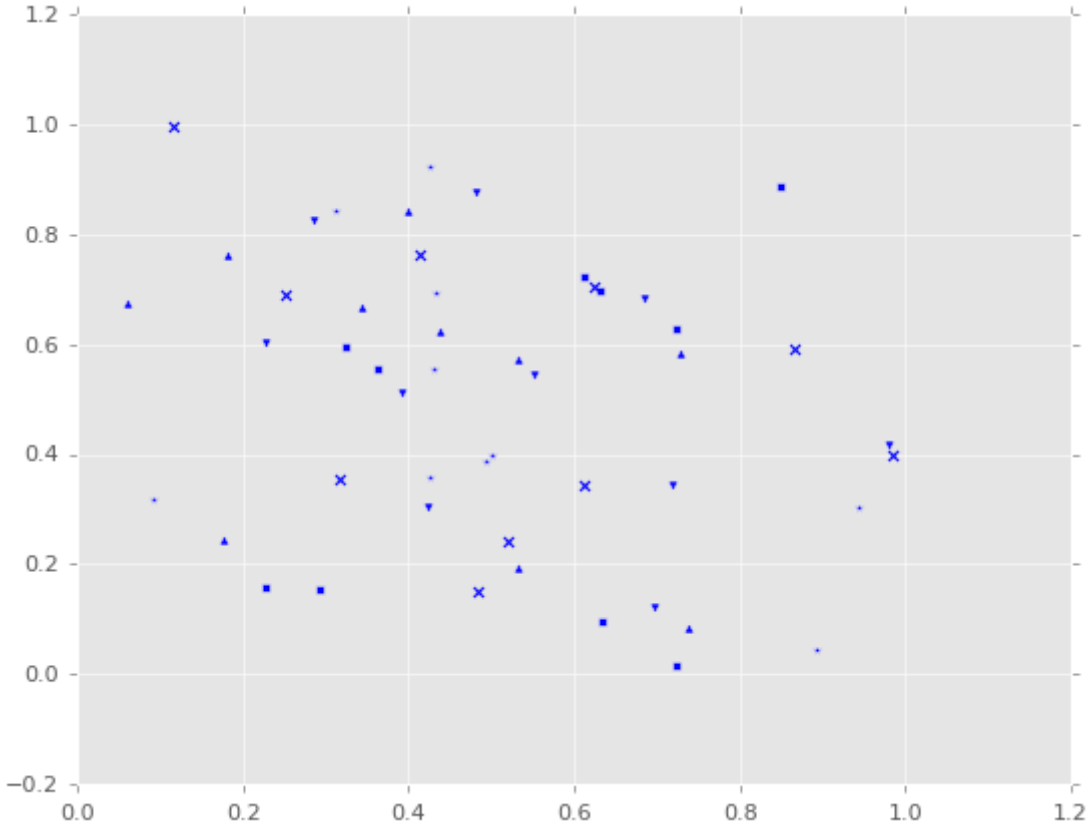
可以利用scatter方法繪製。下列是將50個隨機產生的元素繪製成散佈圖的範圍。範例為了重視與書籍相同的散佈圖，指定了亂數的seed

```
fig, ax = plt.subplots()

#隨機產生50個元素
np.random.seed(123)
x = np.random.rand(50)
y = np.random.rand(50)

#繪製散佈圖
ax.scatter(x,y)

plt.show()
```



預設的資料標記為圓形，若想使用其他形狀，可於marker參數指定。

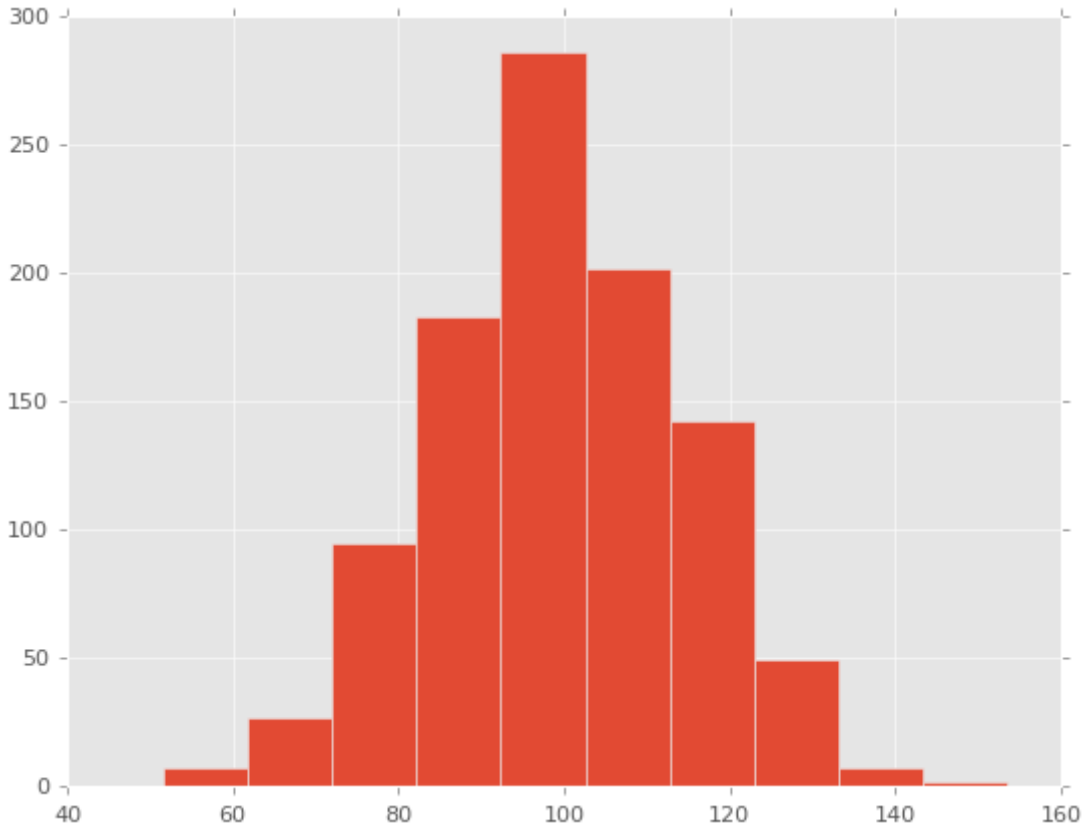
```
fig, ax = plt.subplots()

#隨機產生50個元素
np.random.seed(123)
x = np.random.rand(50)
y = np.random.rand(50)

#下三角形
ax.scatter(x[0:10],y[0:10],marker='v',label='triangle down')

#上三角形
ax.scatter(x[10:20],y[10:20], marker='^',label='triangle up')

#正方形
ax.scatter(x[20:30],y[20:30],marker='s',label='square')

#星形
ax.scatter(x[30:40],y[30:40],marker='*',label='star')

#X
ax.scatter(x[40:50],y[40:50],marker='x',label='x')

plt.show()
```



直方圖

可利用hist方法繪製

```
#產生資料
np.random.seed(123)

#平均
mu = 100

#標準差
sigma = 15

x = np.random.normal(mu,sigma,1000)

fig, ax = plt.subplots()

#繪製直方圖
n, bins, patches = ax.hist(x)

plt.show()
```



hist方法的傳回值可說明次數分佈表的資料，n儲存了各直方的次數(元素數量)，bins儲存了直方的邊界值，patches則儲存了繪製直方所需的資訊，使用n與bins就能以下列程式碼輸出次數分佈表

```
#輸出次數分佈表
for i, num in enumerate(n):
  print('{:.2f} - {:.2f}: {}'.format(bins[i], bins[i+1], num))
```

51.53 - 61.74: 7.0

61.74 - 71.94: 27.0

71.94 - 82.15: 95.0

82.15 - 92.35: 183.0

92.35 - 102.55: 286.0

102.55 - 112.76: 202.0

112.76 - 122.96: 142.0

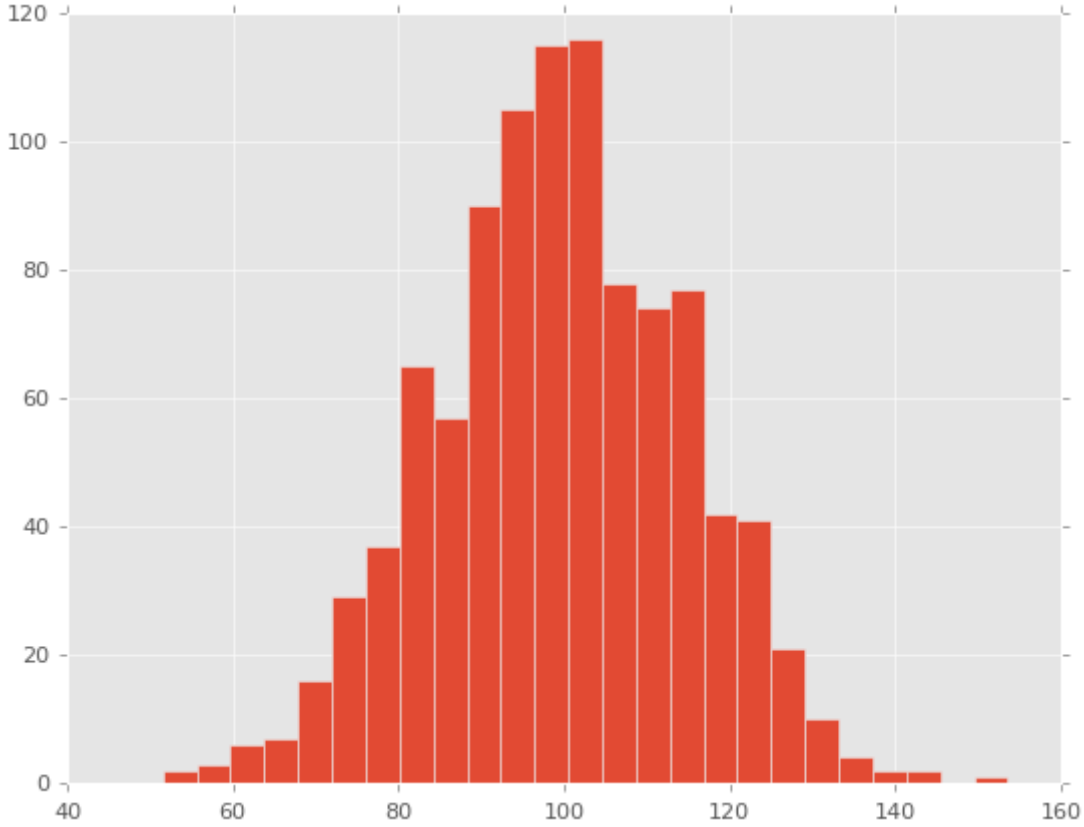
122.96 - 133.17: 49.0

133.17 - 143.37: 7.0

143.37 - 153.57: 2.0

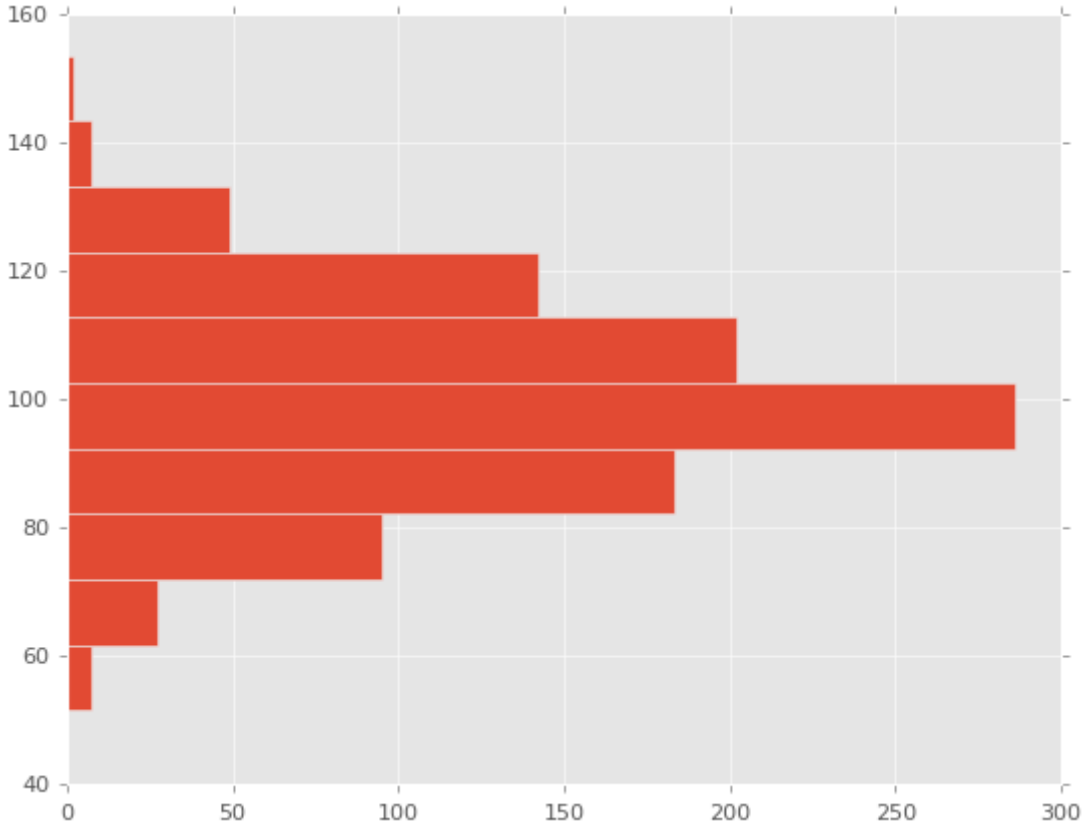
於hist方法的bins參數指定任意值，就能變更直方的數量。預設值為10，下列的程式碼可將同一筆資料繪製成更為細膩的直方圖

```
fig, ax = plt.subplots()

#指定直方的數量
ax.hist(x, bins=25)

plt.show()
```



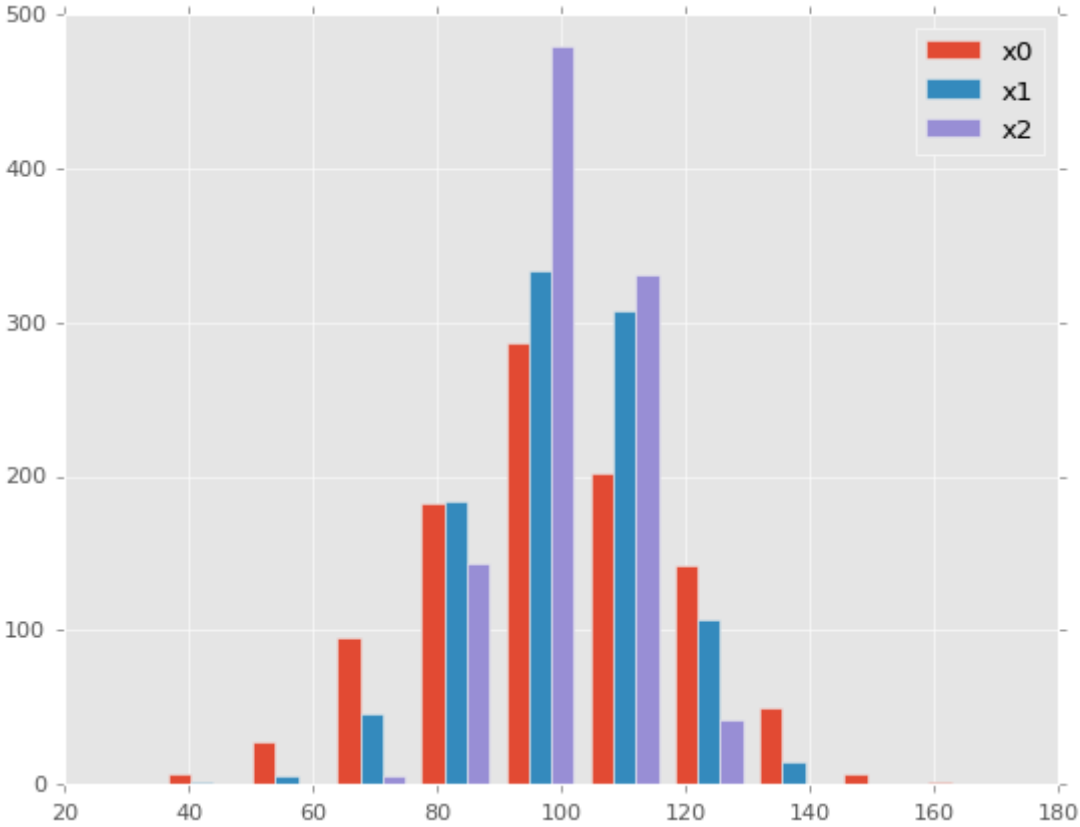
hist 方法的參數若指定為orientation='horizontal'，就能繪製水平的直方圖

```
fig,ax = plt.subplots()

#繪製水平的直方圖
ax.hist(x, orientation='horizontal')

plt.show()
```



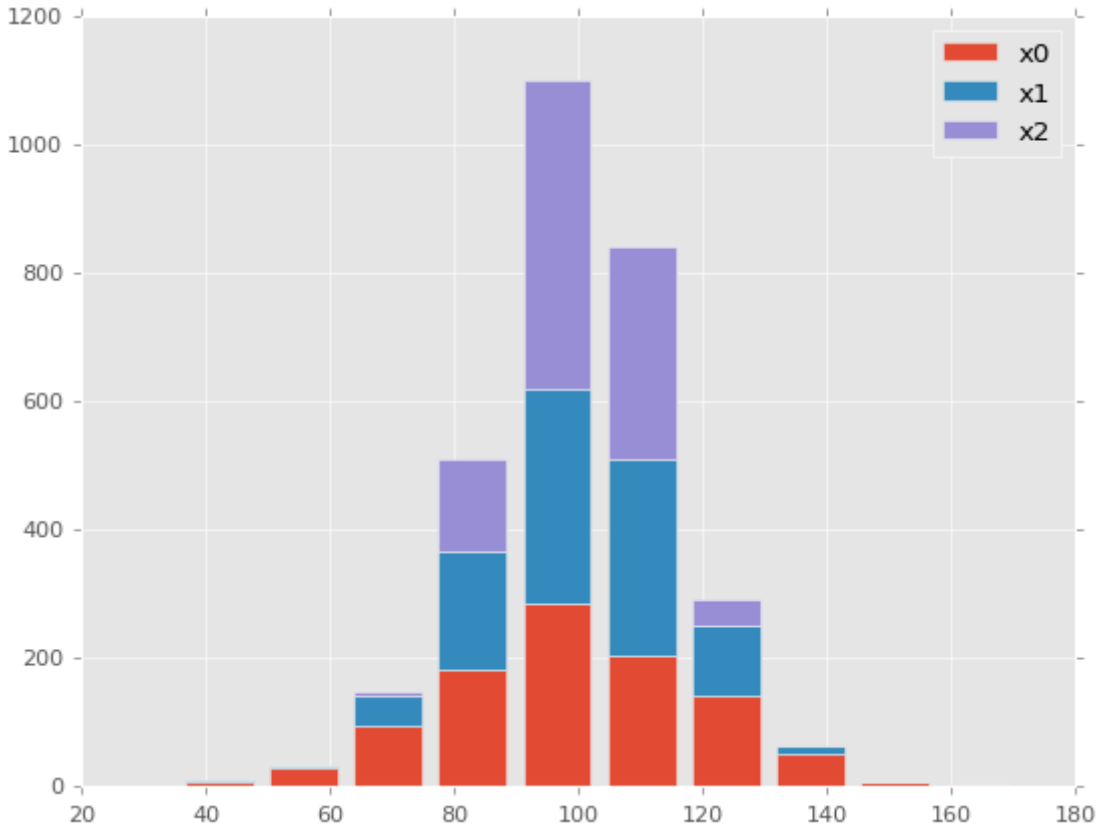
直方圖與長條圖不同的時，一旦指定多個值，直方就會自動水平排列。下列的範例會先建立符合常態分佈的三筆亂數資料 (x0, x1, x2)，在繪製三筆亂數資料並列的直方圖

```
#產生資料
np.random.seed(123)

#平均值
mu = 100
x0 = np.random.normal(mu,20,1000)

#以不同的標準差產生資料
x1 = np.random.normal(mu,15,1000)
x2 = np.random.normal(mu,10,1000)

fig, ax = plt.subplots()

labels = ['x0','x1','x2']

#根據這三筆資料繪製直方圖
ax.hist((x0, x1, x2), label=labels)
ax.legend()

plt.show()
```



hist方法的參數若指定為 stacked=True，則可繪製堆疊直方圖

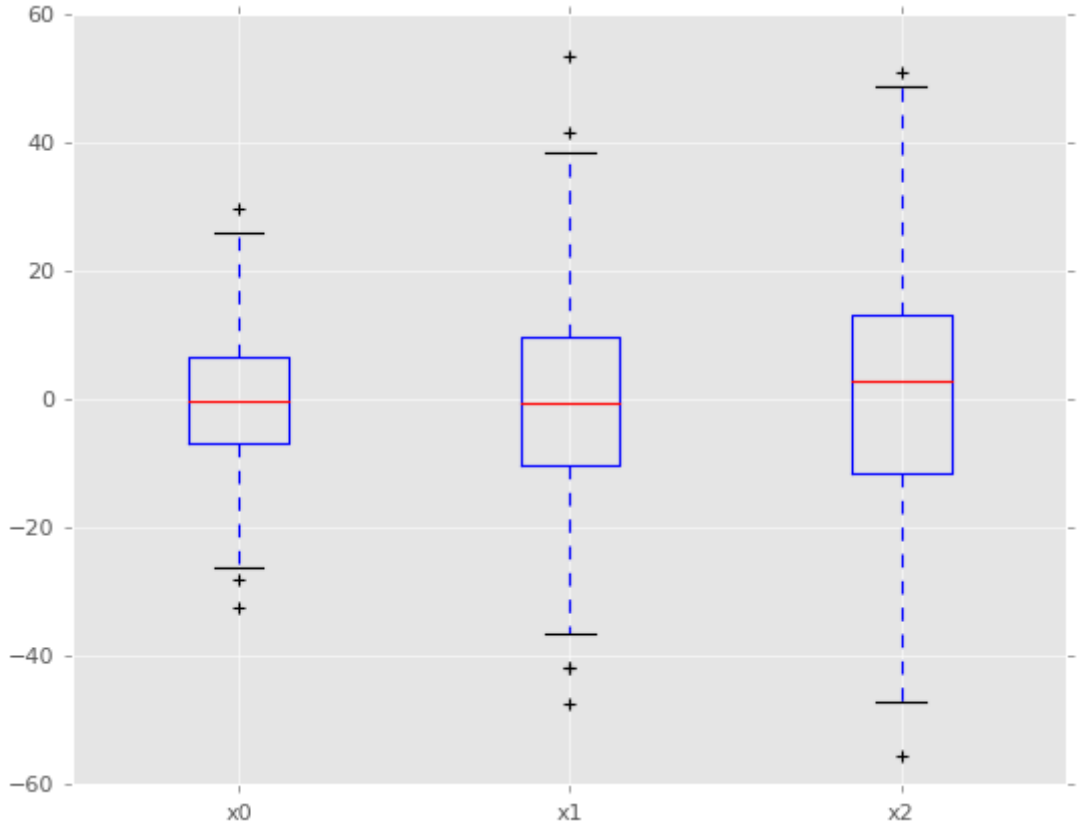
```
fig, ax = plt.subplots()

labels = ['x0', 'x1', 'x2']

#繪製堆疊直方圖
ax.hist((x0, x1, x2), label=labels, stacked=True)
ax.legend()

plt.show()
```



盒鬚圖

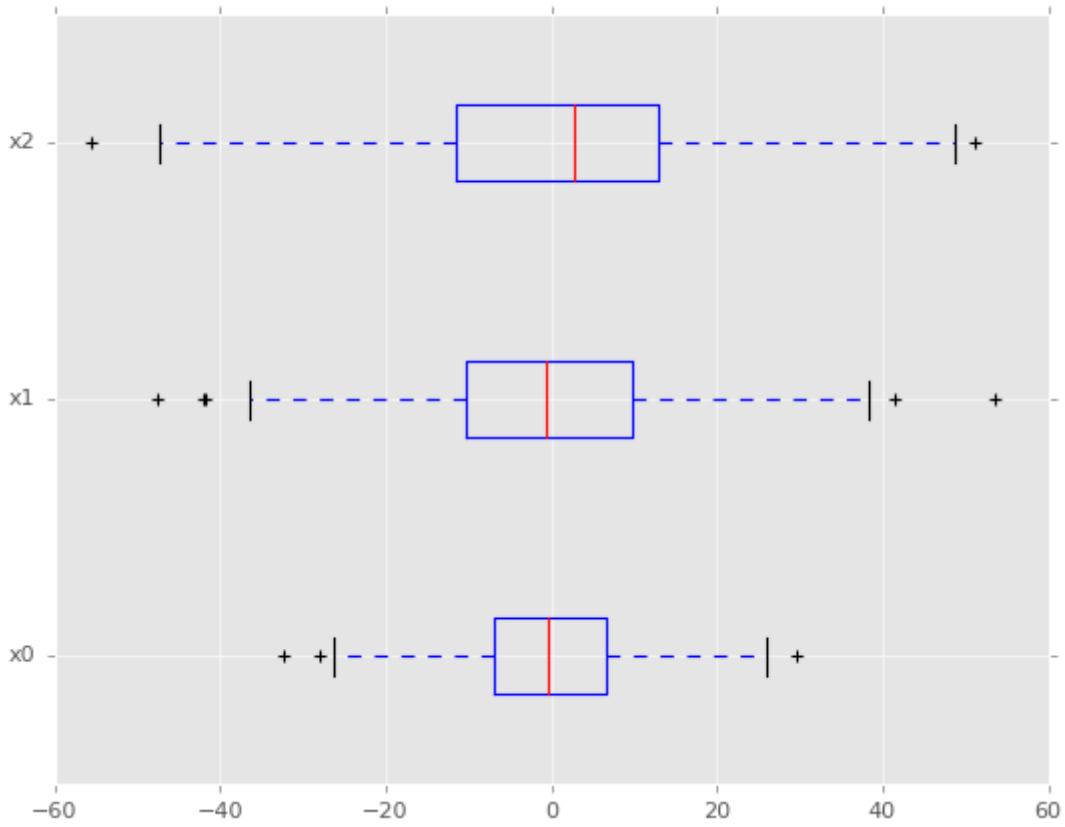
盒鬚圖可使用boxplot方法繪製。下列的程式碼將繪製三種資料 (x0、x1、x2)的盒鬚圖

```
#以不同的標準差產生資料
np.random.seed(123)
x0 = np.random.normal(0,10,500)
x1 = np.random.normal(0,15,500)
x2 = np.random.normal(0,20,500)

fig, ax = plt.subplots()
labels = ['x0','x1','x2']

#繪製盒鬚圖
ax.boxplot((x0,x1,x2),labels=labels)

plt.show()
```



在 boxplot方法的參數指定 vert=False，就能繪製水平的盒鬚圖

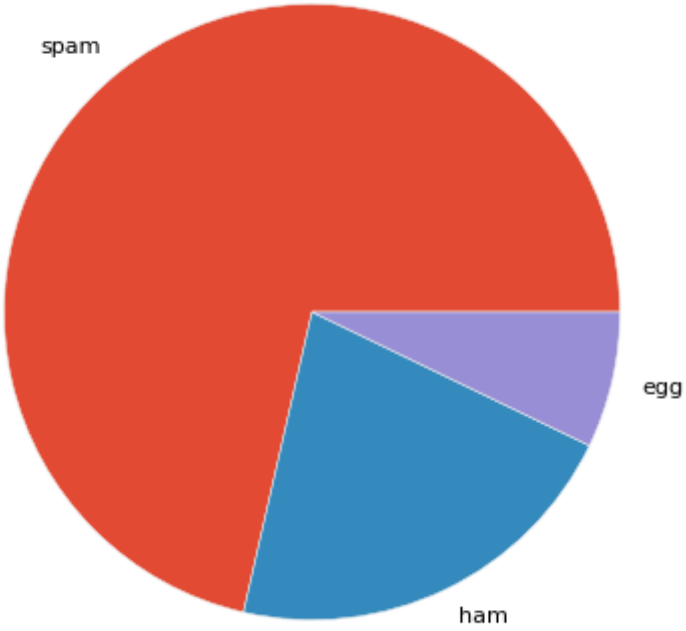
```
fig, ax = plt.subplots()
labels = ['x0','x1','x2']

#繪製水平的盒鬚圖
ax.boxplot((x0,x1,x2),labels=labels, vert=False)

plt.show()
```



圓形圖

圓形圖可利用pie方法繪製

```
labels = ['spam','ham','egg']
x = [10,3,1]

fig, ax = plt.subplots()

#繪製圓形圖
ax.pie(x,labels=labels)

plt.show()
```

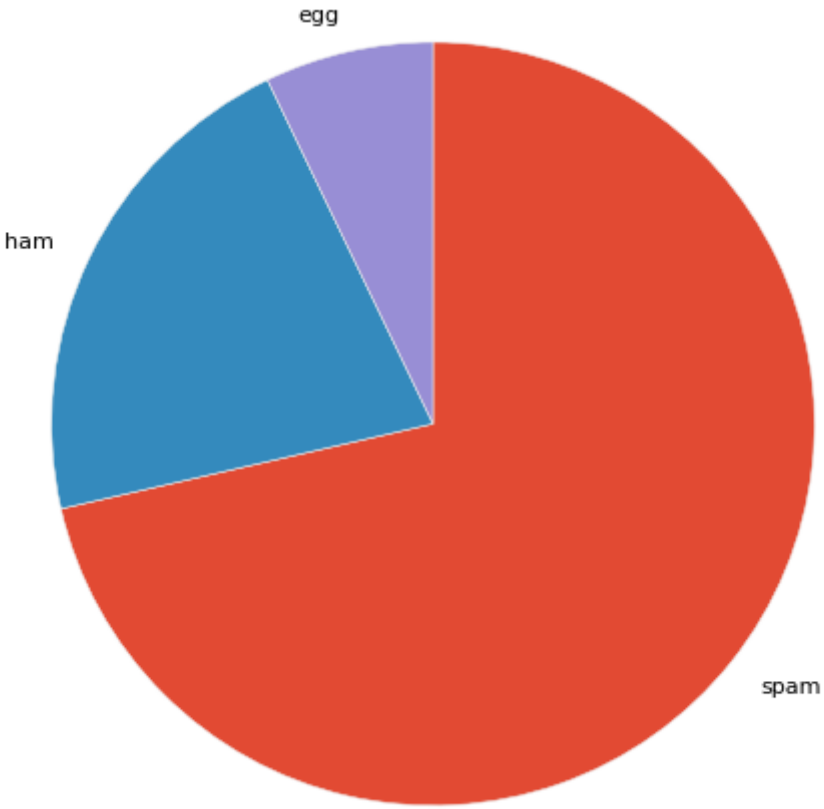


上面範例的圓形圖受繪圖範圍所限而變形成橢圓形，若想保持圓形，可對subplot指定axis('equal')，維持長寬比

```
fig, ax = plt.subplots()
ax.pie(x, labels=labels)

#維持長寬比例的圓形圖
ax.axis('equal')

plt.show()
```

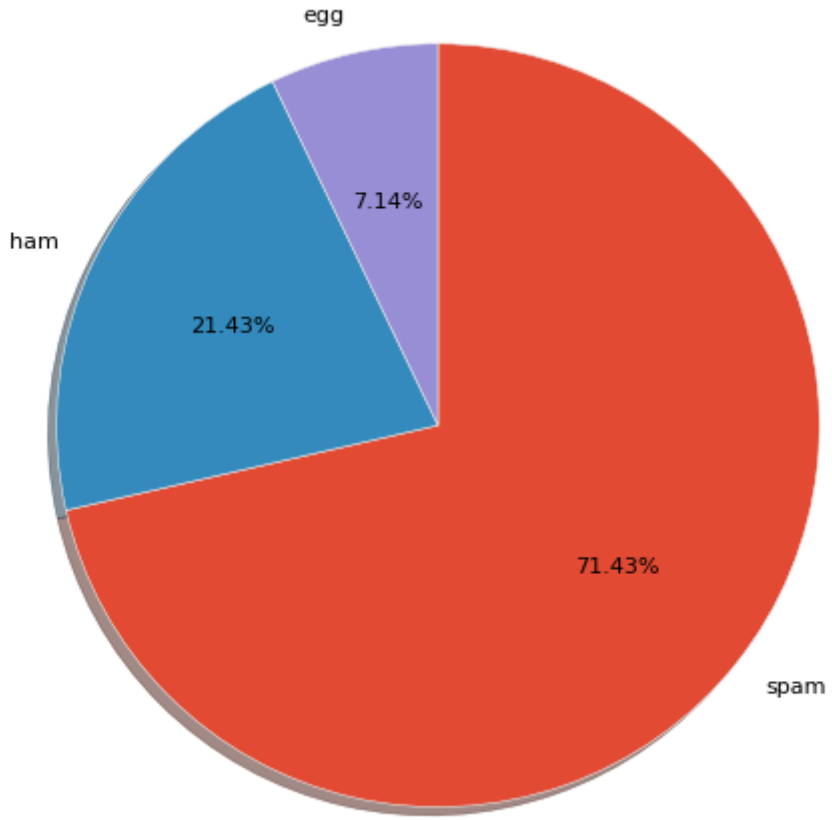
圓形圖預設是從右側 (三點鐘方向)開始沿逆時針方向依序繪製每個元素，若想從上方(十二點方向)開始繪製，可指定startangle=90 (90度的位置)，若要沿著順時針方向繪製元素，則可指定counterclock=False

```
fig,ax = plt.subplots()

#從上方開始，順時針配置元素
ax.pie(x, labels=labels, startangle=90, counterclock=False)
ax.axis('equal')

plt.show()
```



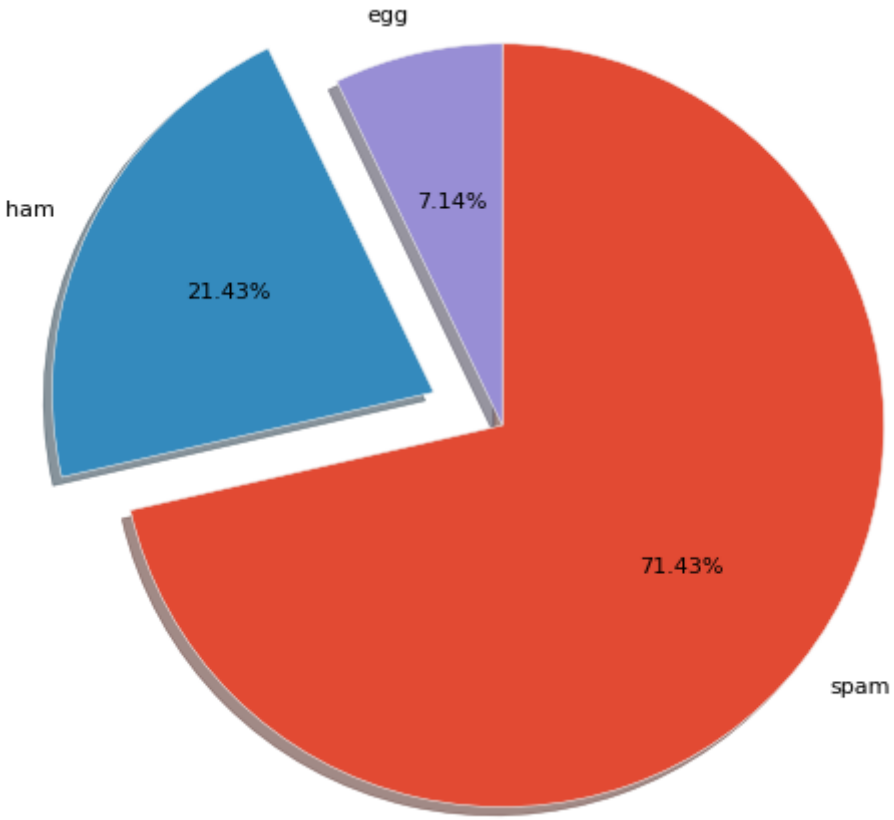
若希望替圓形圖增加陰影，可指定shadow=True。假設指定為autopct='%1.2f%%'，還能追加值的百分比。autopct可指定要顯示幾位數的百分比

```
fig, ax = plt.subplots()

#追加陰影與%符號
ax.pie(x,labels=labels,startangle=90,counterclock=False,shadow=True,
autopct='%1.2f%%')
ax.axis('equal')

plt.show()
```



若希望突顯圓形圖的某個值，可在explode參數指定值，讓元素分割出來。下列的程式碼讓第一個元素分割出來

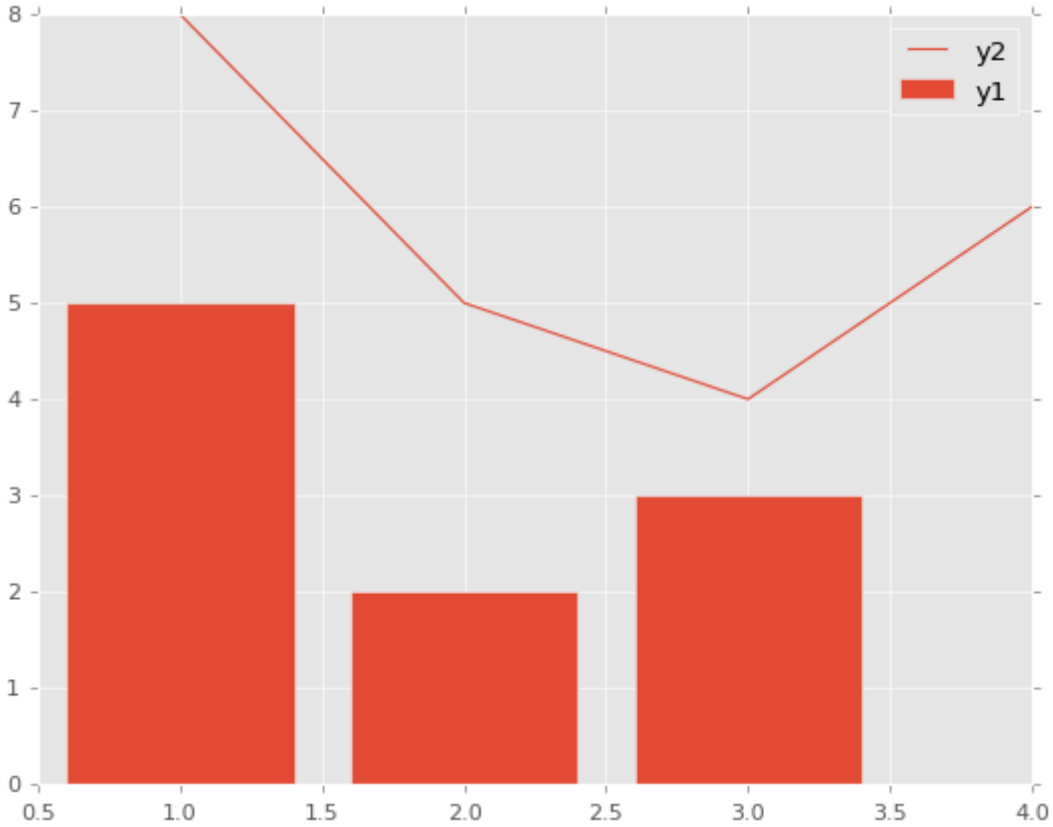
```
#讓第一個元素(ham)分割出來
explode = [0, 0.2, 0]

fig, ax = plt.subplots()

#指定explode
ax.pie(x, labels=labels,
startangle=90,counterclock=False,shadow=True,autopct='%1.2f%%',explode=explode)

ax.axis('equal')

plt.show()
```



組合多個圖表

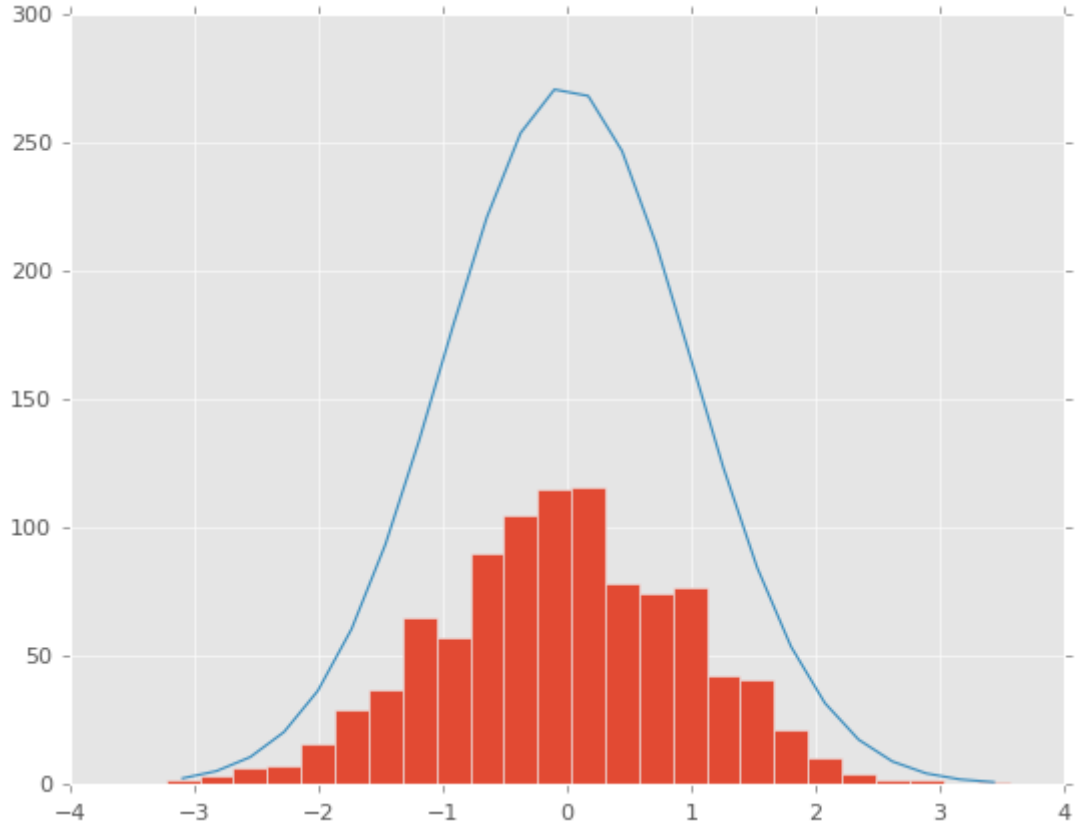
可同時繪製不同的圖表。下列的程式在單一的subplot繪製長條圖與折線圖

```
fig, ax = plt.subplots()

x1 = [1,2,3]
y1 = [5,2,3]
x2 = [1,2,3,4]
y2 = [8,5,4,6]

#繪製長條圖
ax.bar(x1,y1,label='y1')
ax.plot(x2,y2,label='y2')
ax.legend()

plt.show()
```



也可以在直方圖追加折線圖，繪製趨勢線

```
np.random.seed(123)

#產生正規亂數
x = np.random.randn(1000)

fig, ax = plt.subplots()

#繪製直方圖
counts, edges, patches = ax.hist(x, bins=25)

#求出用於趨勢線的點 (位於直方的中點)
x_fit = (edges[:-1] + edges[1:]) / 2

#繪製趨勢線
y = 1000 * np.diff(edges) * np.exp(-x_fit**2/2)
ax.plot(x_fit,y)

plt.show()
```

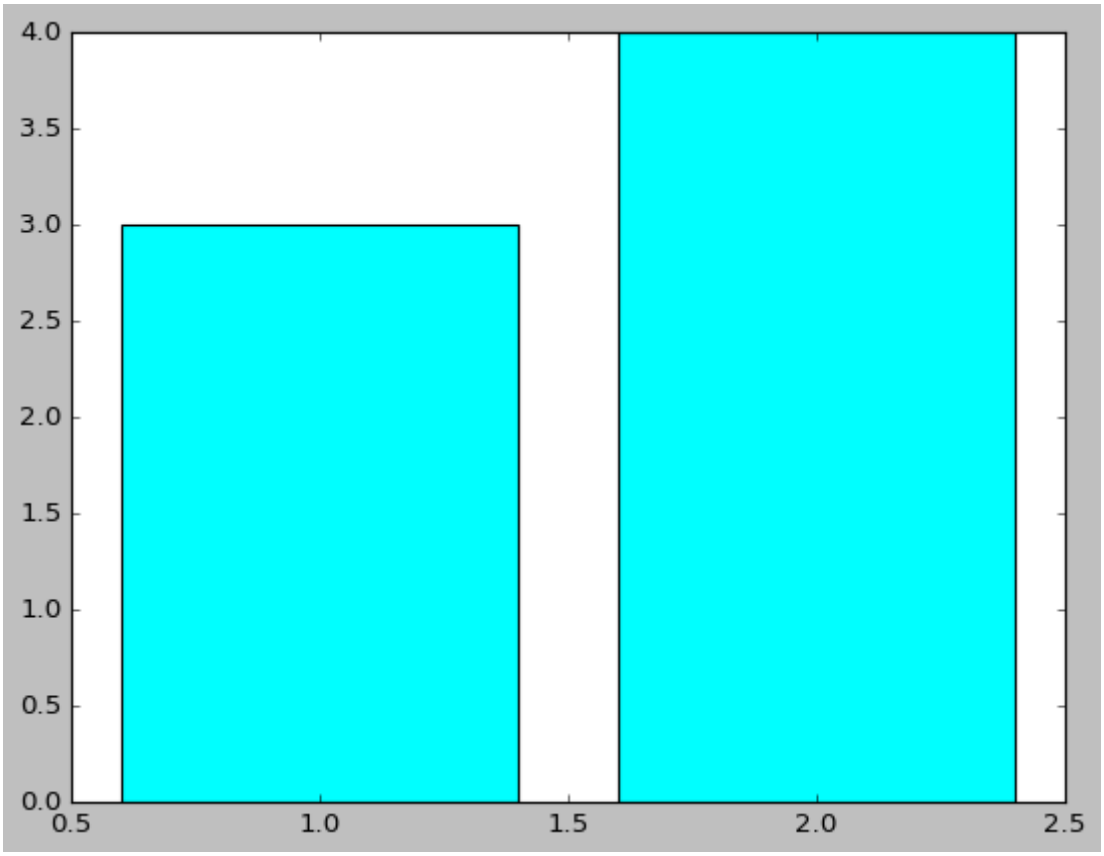


4. 樣式

設定顏色

長條圖或散佈圖可利用color參數或edgecolor參數指定顏色。color參數可指定填色，edgecolor參數可指定框線顏色。下列的程式碼只指定了填色的長條圖以及連框線顏色都一併指定的長條圖。

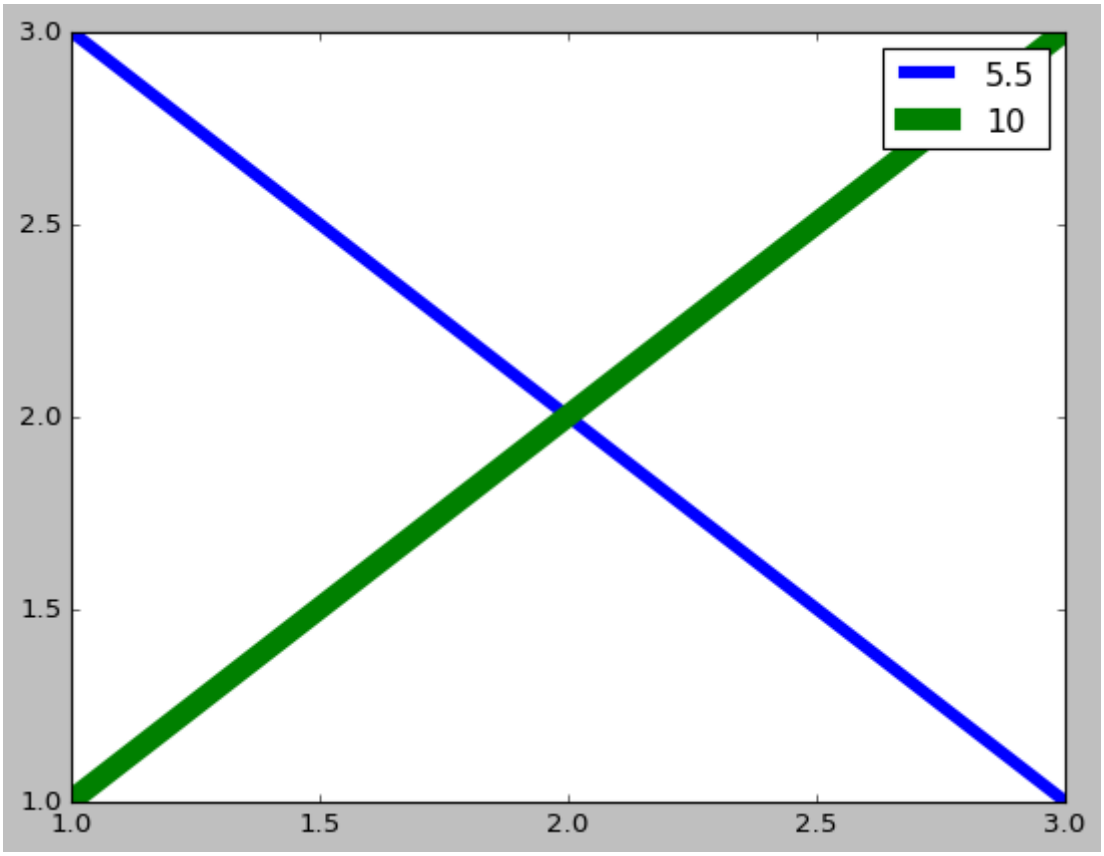
```
fig, ax = plt.subplots()

#指定填色
ax.bar([1],[3], color='aqua')

#指定填色與框線顏色
ax.bar([2],[4], color='aqua', edgecolor='black')
plt.show()
```



線條樣式

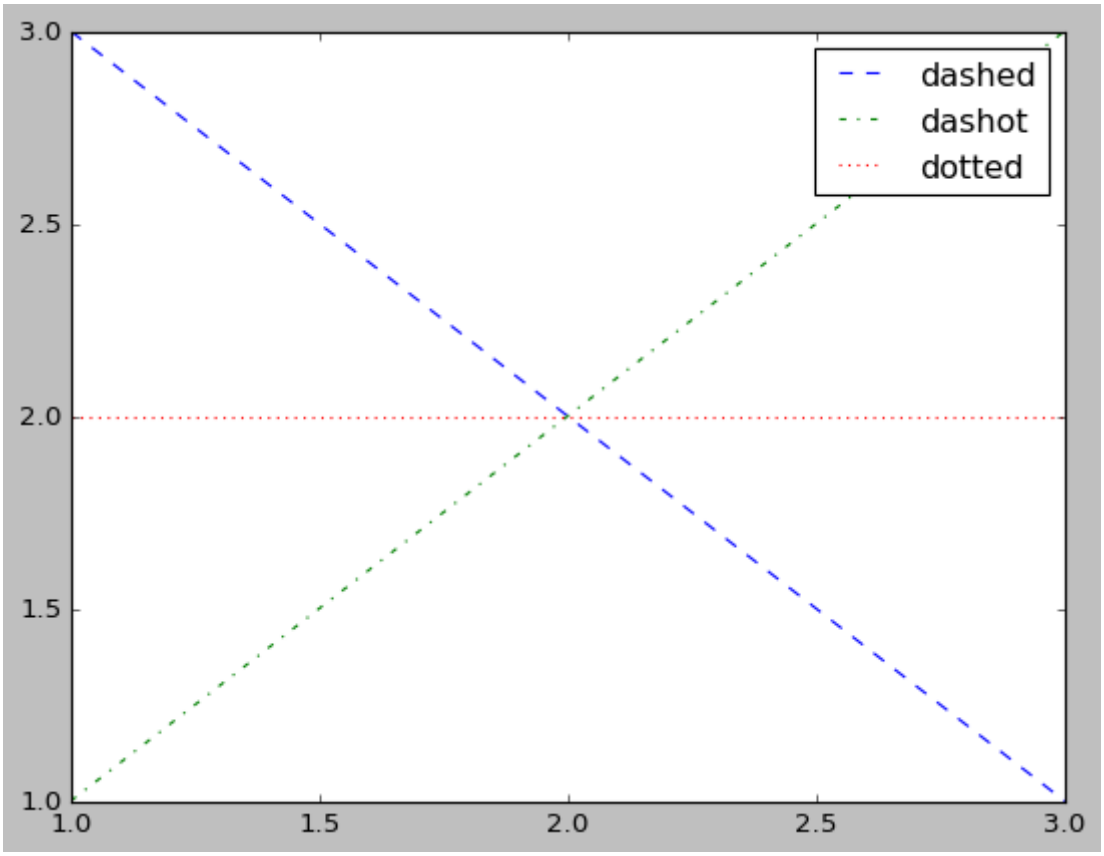
可在折線圖或圖表框線、分割線或其他線條套用樣式。linewidth參數可調整線條的寬度，調整的單位為點

```
fig, ax = plt.subplots()

#繪製5.5點寬的線條
ax.plot([1,3],[3,1],linewidth=5.5, label='5.5')

#繪製10點寬線條
ax.plot([1,3],[1,3],linewidth=10, label='10')

ax.legend()

plt.show()
```



linestyle參數可指定線條種類。下列程式碼繪製了虛線(--)、鎖鍊線(-.)、點線(:)

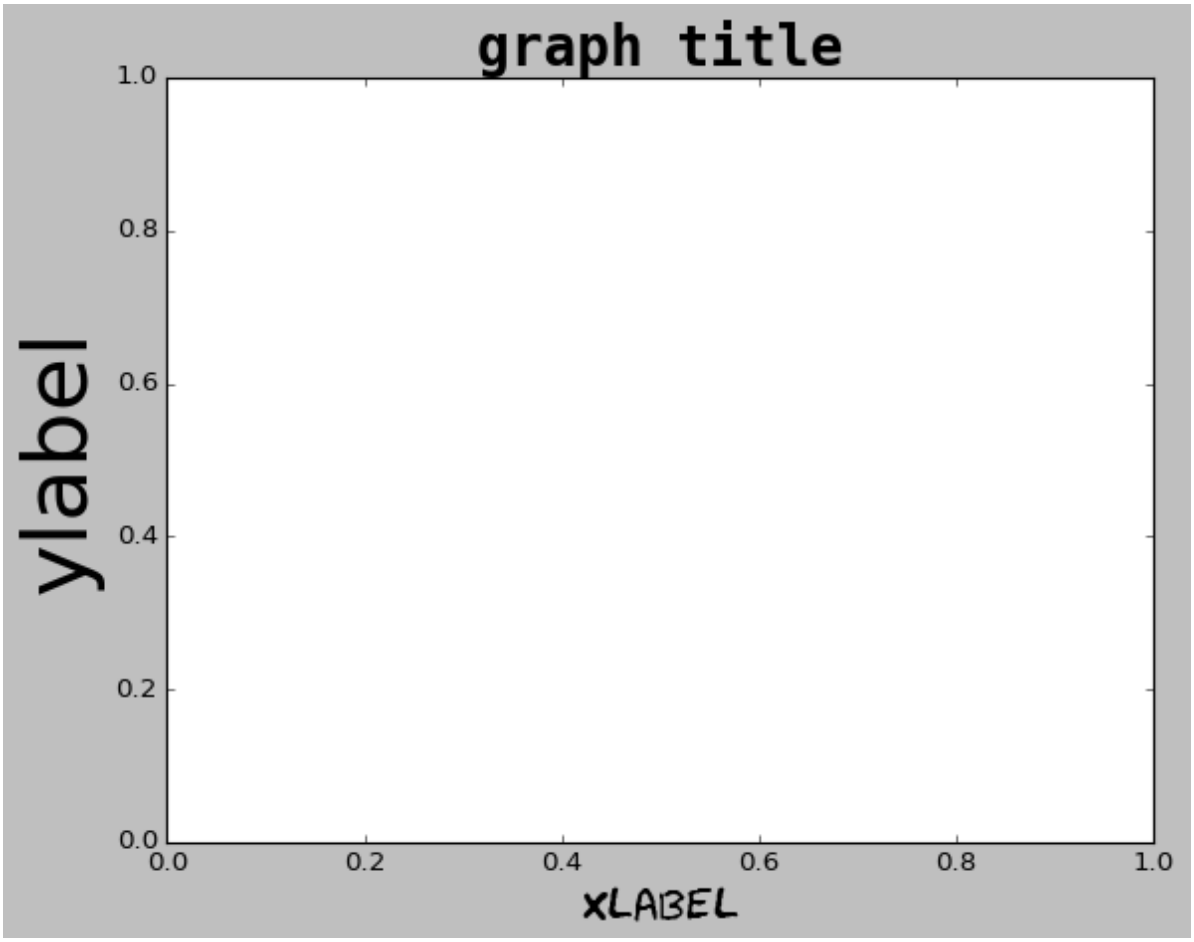
```
fig, ax = plt.subplots()

#繪製虛線
ax.plot([1,3],[3,1],linestyle='--',label='dashed')

#繪製鎖鏈線
ax.plot([1,3],[1,3],linestyle='-.',label='dashot')

#繪製點線
ax.plot([1,3],[2,2],linestyle=':',label='dotted')
ax.legend()

plt.show()
```



字型

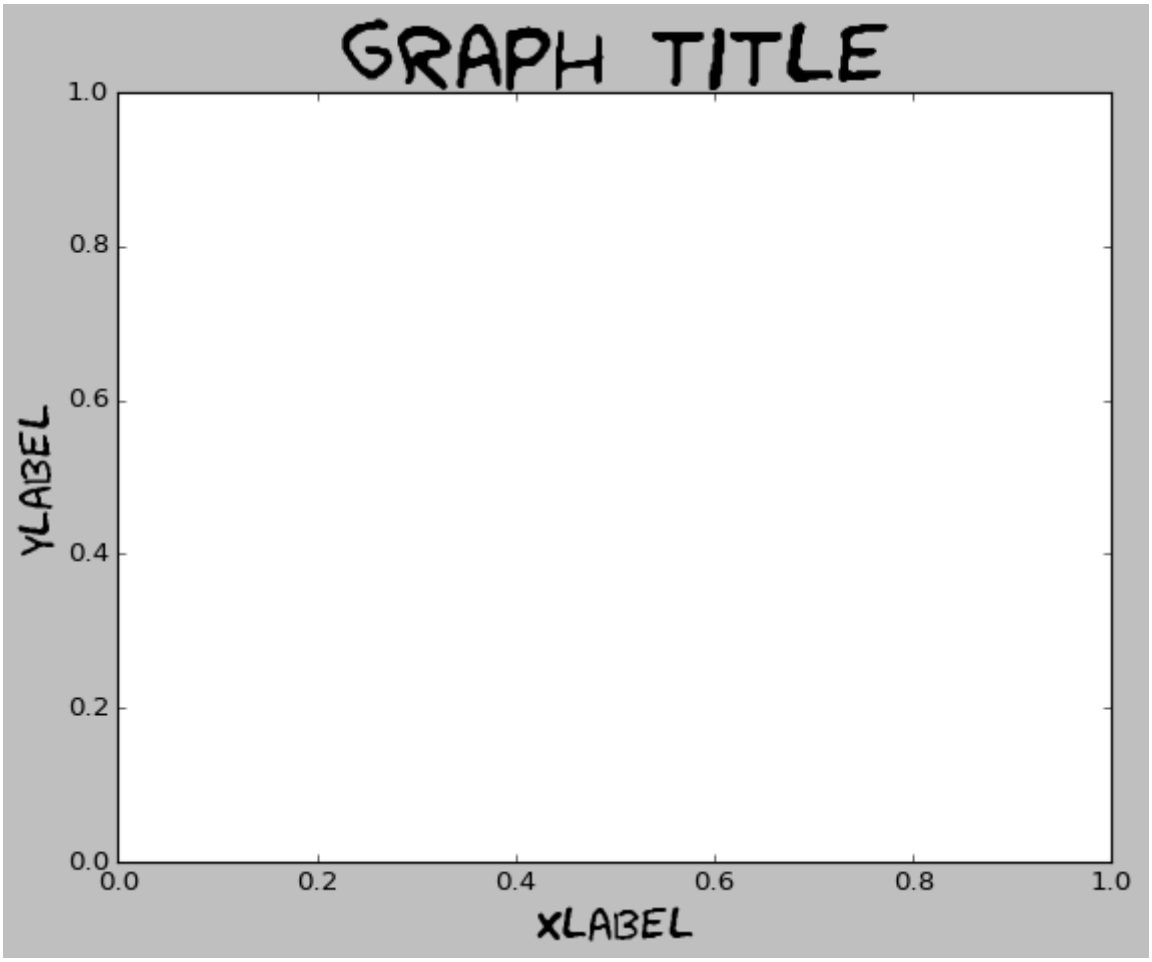
標題、圖例、座標軸標籤這類文字都可設定樣式。size參數可設定字型大小 (單位 : 點)，weight參數可設定字型的粗細 (light、bold)。此外，family參數可設定字型種類，預設可使用serif、sans-serif、cursive、fantasy、monospace

```
fig, ax = plt.subplots()

ax.set_xlabel('xlabel', family='fantasy', size=20, weight='bold')

ax.set_ylabel('ylabel', family='cursive', size=40, weight='light')

ax.set_title('graph title', family='monospace', size=25, weight='heavy')

plt.show()
```



要在不同的位置設定相同的字型樣式，卻得每次都用參數指定的話，一定非常麻煩，此時可以將字型的設定製作成字典資料，然後指定給fontdict 參數即可。

```
#以字典定義字型樣式
fontdict = {
    'family': 'fantasy',
    'size': 20,
    'weight': 'normal',
}

fig, ax = plt.subplots()

#以字典格式指定字型樣式
ax.set_xlabel('xlabel', fontdict=fontdict)
ax.set_ylabel('ylabel', fontdict=fontdict)

#可利用個別指定的size覆寫設定
ax.set_title('graph title', fontdict=fontdict, size=40)

plt.show()
```

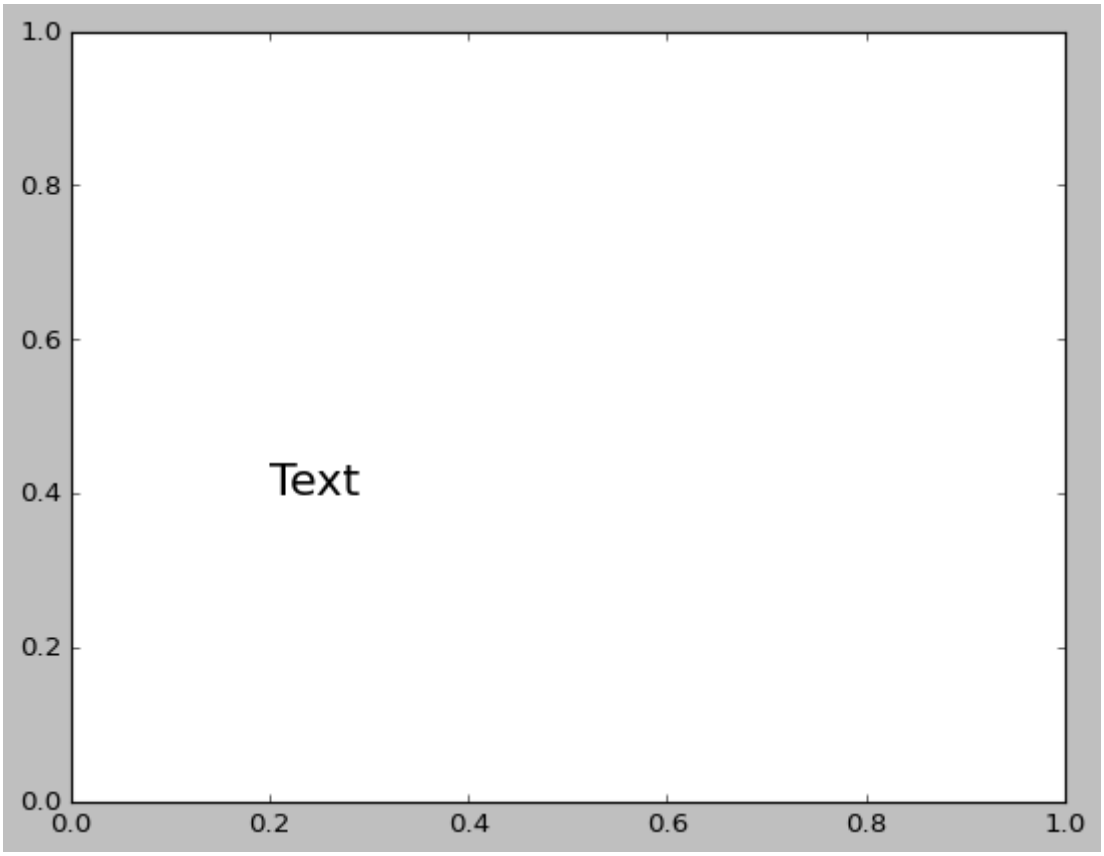


- 上述範例直接以set_title方法指定size參數，所以subplot的標題未套用字典定義的20點，而是40點

繪製文字

text方法可在圖表繪製任何內容的文字，第一、二參數可指定文字的左下角 X、Y座標，也可指定與字型樣式相同的參數

```
fig, ax = plt.subplots()

#繪製Text這個文字
ax.text(0.2,0.4,'Text',size=20)

plt.show()
```



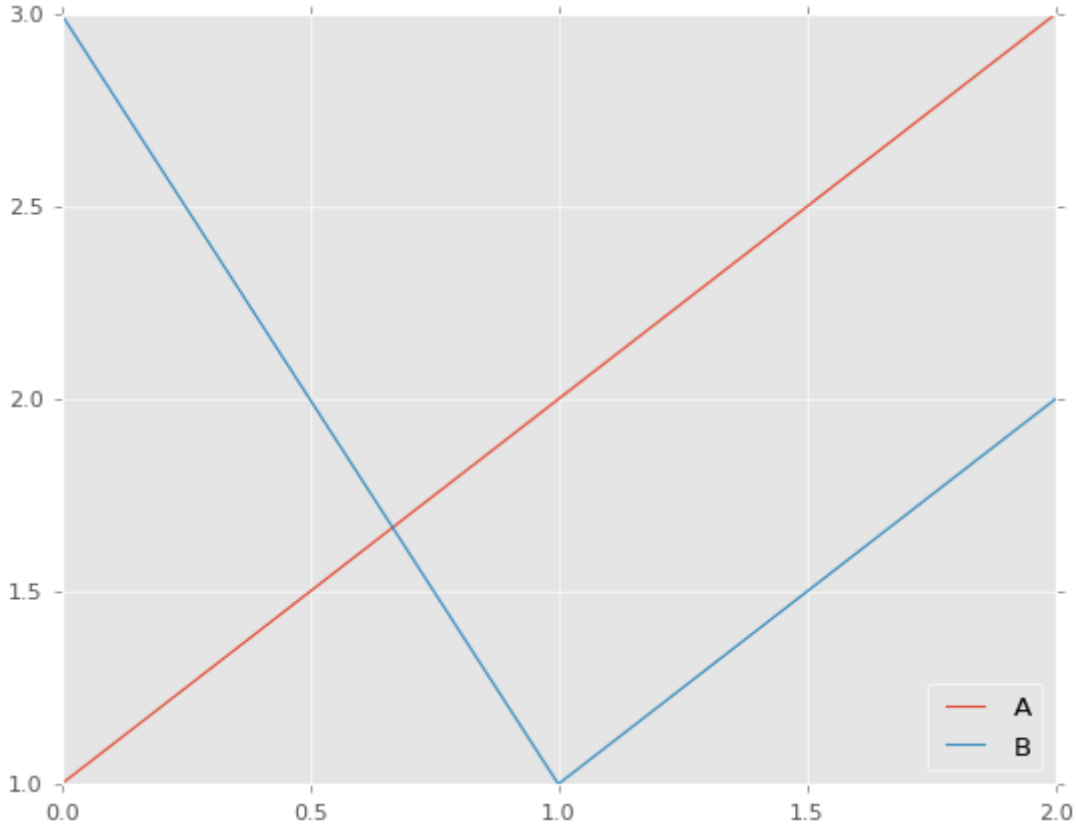
利用 pandas 的物件繪製圖表

可利用 pandas的DataFrame、Series繪製圖表。這種繪製方法的內部也使用了Matplotlib。

這種繪製圖表的方式雖然無法微調圖表的內容，也無法同時繪製多個圖表，卻能以圖表呈現DataFrame或Series的資料。

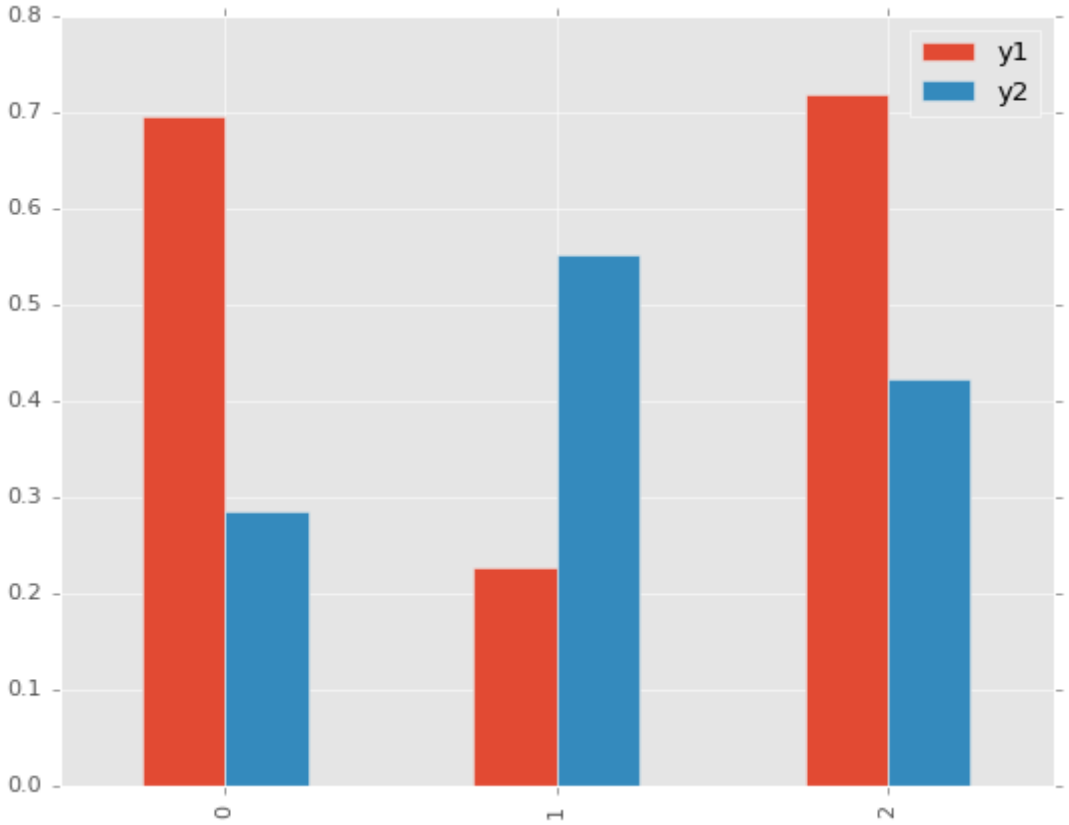
對DataFrame呼叫plot方法，即可繪製折線圖

```
import pandas as pd
import matplotlib.style
import matplotlib.pyplot as plt

#指定樣式
matplotlib.style.use('ggplot')

#建立DataFrame
df = pd.DataFrame({'A': [1,2,3], 'B': [3,1,2]})

#繪製折線圖
df.plot()

plt.show()
```



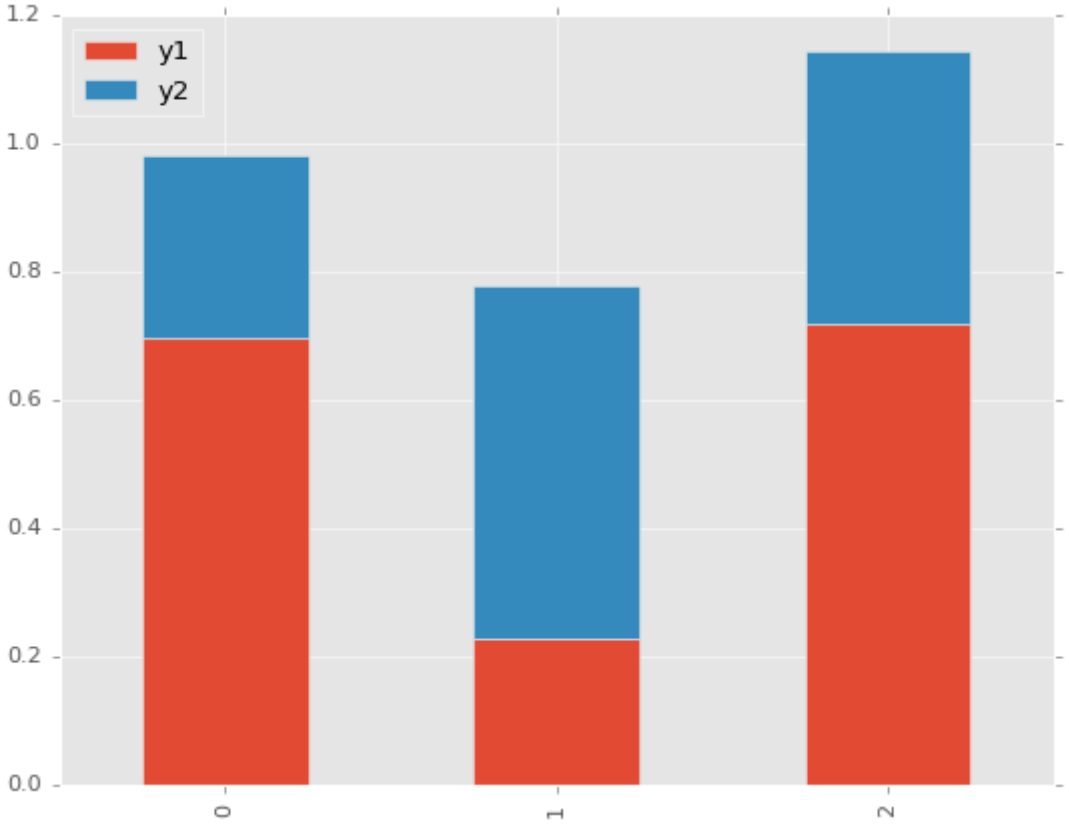
長條圖可利用plot.bar繪製。假設資料有很多筆，則會自動排成一列

```
import numpy as np

#隨機產生3列2行的資料，藉此建立DataFrame
np.random.seed(123)
df2 = pd.DataFrame(np.random.rand(3,2),columns=['y1','y2'])

#繪製長條圖
df2.plot.bar()
plt.show()
```



若參數指定為 stacked=True，則可繪製堆疊長條圖

```
#繪製堆疊長條圖
df2.plot.bar(stacked=True)
plt.show()
```



- 利用plot.barh繪製橫條圖
- 利用plot.scatter繪製散佈圖
- 利用plot.hist縮製直方圖
- 利用plot.box繪製盒鬚圖
- 利用plot.pie繪製圓形圖