Chapter 4

資料視覺化能力

4-3 圖表繪製其他技巧

4-3-1 多圖表繪製

▶▶ 範例程式 **E4-3-1-1.ipynb**

多個子圖

子圖功能用來並排比較不同的資料圖，以提高說明力。Matplotlib 子圖有四種方式可以在單個圖中一起存在的較小軸組子圖，這些子圖可能是插圖，圖形網格或其他更複雜的佈局。

In[1]	%matplotlib inline import matplotlib.pyplot as plt plt.style.use('seaborn-white') import numpy as np

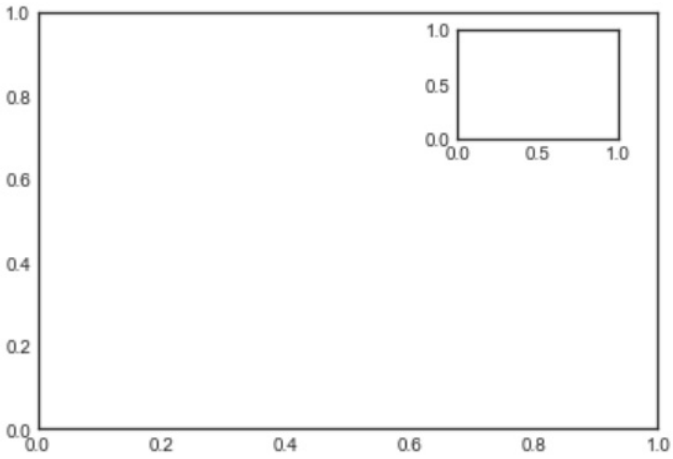
plt.axes 手繪子圖

創建軸的最基本方法是使用 plt.axes 函數。預設情況下，這會創建一個填充整個圖形的標準軸物件。plt.axes 也有可選參數，它是圖形座標系中四個數字的串列。這些數字代表圖形座標系中的「左、底、寬、高」，其範圍從圖的左下角的 0 到圖的右上角的 1。

例如，我們可以通過將 x 和 y 位置設置為 0.65（也就是說，從寬度的 65%和高度的 65%開始，在另一個軸的右上角創建一個插入軸與圖）和 x 和 y 範圍為 0.2（即軸的大小是寬度的 20%和圖的高度的 20%）：

In[2]	ax1 = plt.axes() # standard axes
	ax2 = plt.axes([0.65, 0.65, 0.2, 0.2])

Out[2]

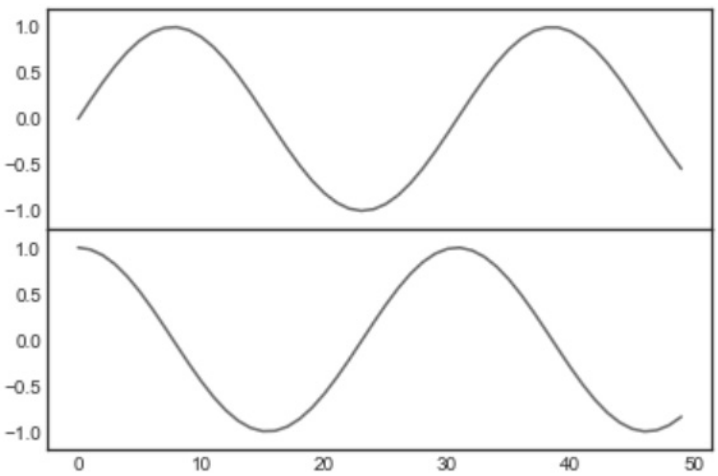


在物件導向的介面中，這個命令的等價指令是 fig.add_axes()。下面的例子，即以此創建兩個垂直堆疊的軸：

In[3]

```
fig = plt.figure()
ax1 = fig.add_axes([0.1, 0.5, 0.8, 0.4],
                   xticklabels=[], ylim=(-1.2, 1.2))
ax2 = fig.add_axes([0.1, 0.1, 0.8, 0.4],
                   ylim=(-1.2, 1.2))

x = np.linspace(0, 10)
ax1.plot(np.sin(x))
ax2.plot(np.cos(x));
```

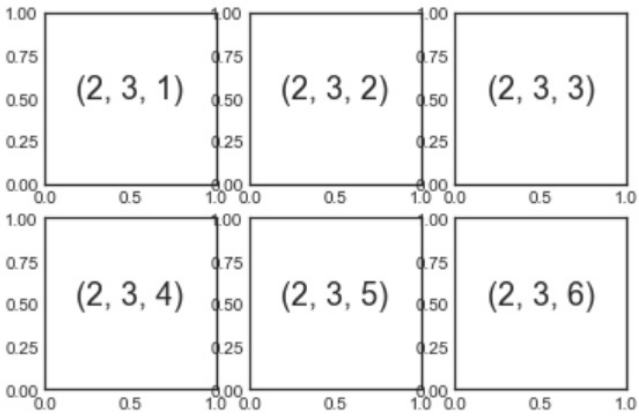
Out[3]



plt.subplot 子圖的簡單網格

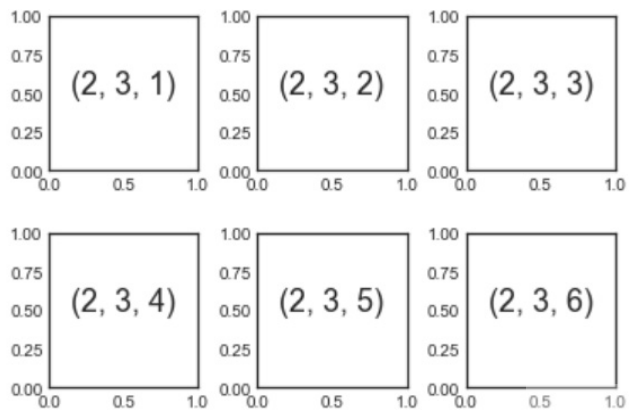
這是對齊列或行的子圖，利用 plt.subplot()，在網格中創建一個子圖。採用三個整數參數，列數、行數和要在此創建的繪圖的索引，從左上角到右下角：

In[4]	for i in range(1, 7):
	    plt.subplot(2, 3, i)
	    plt.text(0.5, 0.5, str((2, 3, i)),
	             fontsize=18, ha='center')

Out[4]	

命令 plt.subplots_adjust 可用於調整這些圖之間的間距。下面的代碼使用等效的物件導向命令 fig.add_subplot()：

In[5]	fig = plt.figure()
	fig.subplots_adjust(hspace=0.4, wspace=0.4)
	for i in range(1, 7):
	    ax = fig.add_subplot(2, 3, i)
	    ax.text(0.5, 0.5, str((2, 3, i)),
	           fontsize=18, ha='center')

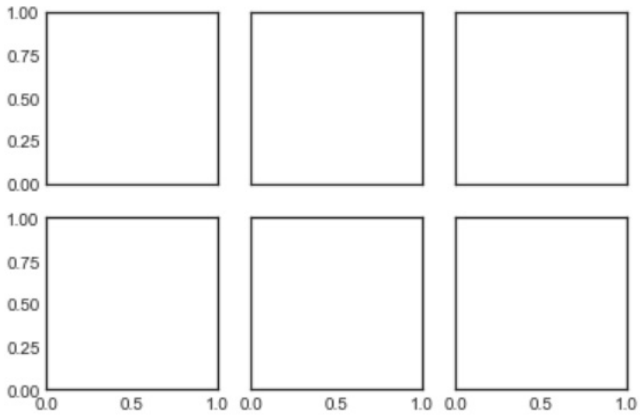
Out[5]	

上面例子中也用了 plt.subplots_adjust 的 hspace 和 wspace 參數，它們以子圖大小為單位指定沿圖的高度和寬度的間距。如上圖，間距是子圖寬度和高度的 40%。

plt.subplots 整體網格一氣呵成

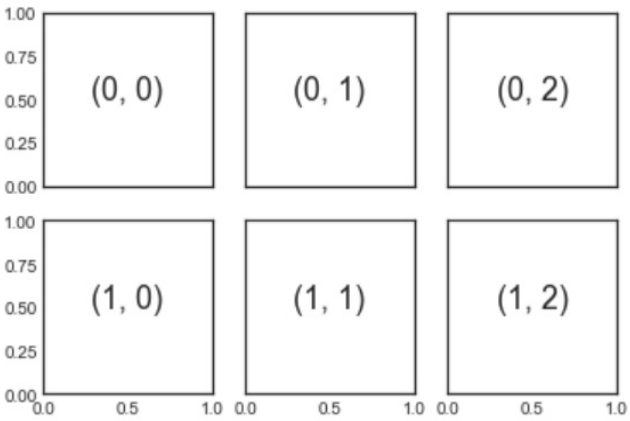
在創建大型子圖網格時，建議使用 plt.subplots()（注意 subplots 末尾的 s）。該函數不是創建單個子圖，而是在一行中創建完整的子圖網格，並將它們返回到 NumPy 陣列中。其參數是列數和行數，以及可選關鍵字 sharex 和 sharey，允許您設定不同軸之間的關係。

下面的例子創建子圖的網格，其中同一行中的所有軸共用其 y 軸刻度，並且同一列中的所有軸共用其 x 軸刻度：

In[6]	`fig, ax = plt.subplots(2, 3, sharex='col', sharey='row')`
Out[6]	

通過指定 sharex 和 sharey，可以自動刪除網格上的內部標籤，以使繪圖更清晰。生成的軸實例網格在 NumPy 陣列中返回，允許使用標準陣列索引標記法方便地指定所需的軸：

```
In[7]
# axes are in a two-dimensional array, indexed by [row, col]
for i in range(2):
    for j in range(3):
        ax[i, j].text(0.5, 0.5, str((i, j)),
                      fontsize=18, ha='center')
fig
```

Out[7]	

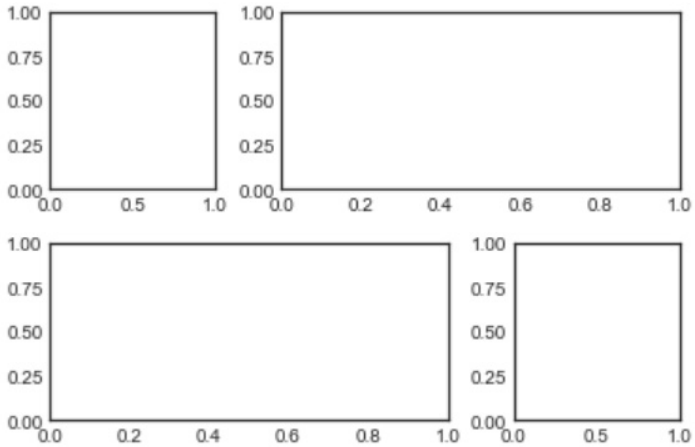
plt.GridSpec 更複雜的安排

要超越常規網格到跨越多行和列的子圖，可以使用 plt.GridSpec()。plt.GridSpec() 物件本身不會創建一個圖，它只是一個方便的介面，可通過 plt.subplot()命令識別。例如，具有一些指定寬度和高度空間的兩行和三列網格的 gridspec 如下所示：

In[8]	`grid = plt.GridSpec(2, 3, wspace=0.4, hspace=0.3)`

從這裡我們可以使用 familiary Python 切片語法指定子圖位置和範圍：

In[9]	`plt.subplot(grid[0, 0])`
	`plt.subplot(grid[0, 1:])`
	`plt.subplot(grid[1, :2])`
	`plt.subplot(grid[1, 2]);`

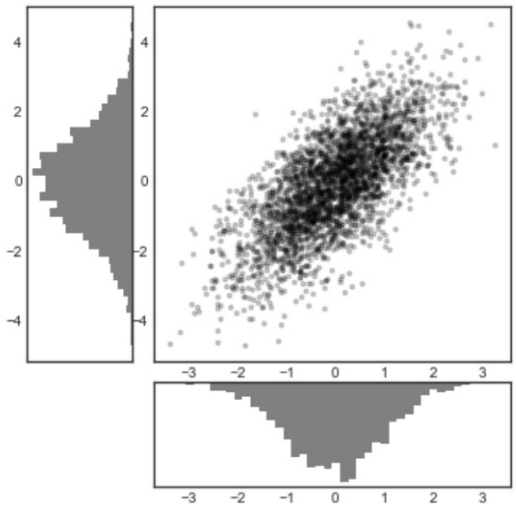
Out[9]	

這種類型的柔性網格對齊具有廣泛用途，在創建多軸長條圖時使用，如下圖所示：

In[10]

```
# Create some normally distributed data
mean = [0, 0]
cov = [[1, 1], [1, 2]]
x, y = np.random.multivariate_normal(mean, cov, 3000).T

# Set up the axes with gridspec
fig = plt.figure(figsize=(6, 6))
grid = plt.GridSpec(4, 4, hspace=0.2, wspace=0.2)
main_ax = fig.add_subplot(grid[:-1, 1:])
y_hist = fig.add_subplot(grid[:-1, 0], xticklabels=[],
sharey=main_ax)
x_hist = fig.add_subplot(grid[-1, 1:], yticklabels=[],
sharex=main_ax)

# scatter points on the main axes
main_ax.plot(x, y, 'ok', markersize=3, alpha=0.2)

# histogram on the attached axes
x_hist.hist(x, 40, histtype='stepfilled',
            orientation='vertical', color='gray')
x_hist.invert_yaxis()

y_hist.hist(y, 40, histtype='stepfilled',
            orientation='horizontal', color='gray')
y_hist.invert_xaxis()
```

Out[10]



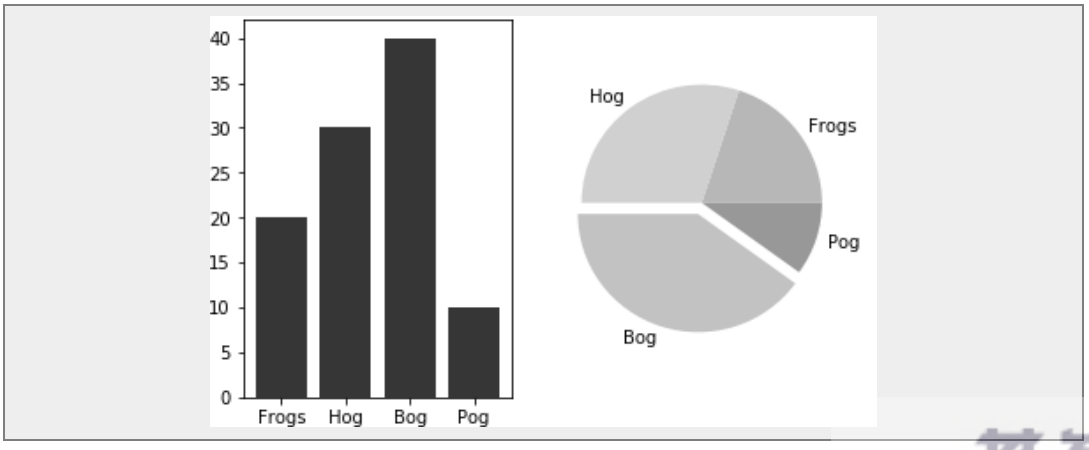
範例程式 E4-3-1-2.py

```
1   import matplotlib.pyplot as plt
2   labels = 'Frogs', 'Hog', 'Bog', 'Pog'
3   sizes = [20, 30, 40, 10]
4   colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
5   explode = (0, 0, 0.1, 0)
6   plt.subplot(1, 2, 1)
7   plt.bar(labels, sizes, color="red")
8   plt.subplot(1, 2, 2)
9   plt.pie(sizes, explode=explode, labels=labels, colors=colors)
10  plt.axis('equal')
11  plt.show()
```

範例程式說明

- 1 行匯入 matplotlib.pyplot 套件。
- 2 行輸入標籤資料。
- 3 行輸入圓形圖中各扇形大小，每個扇形的分數面積由 x/sum(x)給出。扇形是逆時針繪製的，預設情況下從 x 軸開始。
- 4 行輸入圓形圖中各扇形顏色。
- 5 行指定用於偏移每個扇形的半徑的分數。
- 6-7 行指定第一個子圖的位置，繪製長條圖。
- 8-9 行指定第二個子圖的位置，繪製圓形圖。
- 10 行設定座標軸尺度相等。
- 11 行要求顯示圖形。

輸出結果



範例程式 **E4-3-1-3.py**

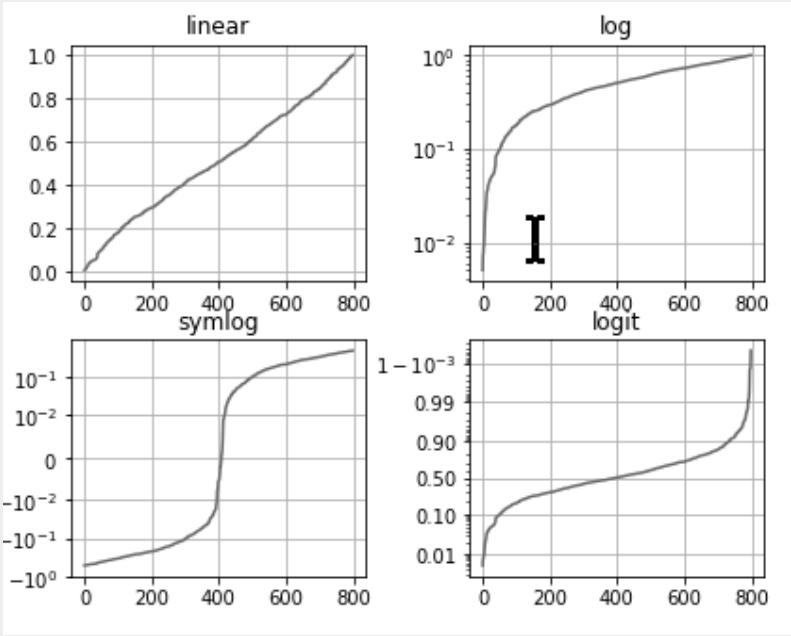
```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   from matplotlib.ticker import NullFormatter  # useful for `logit` scale
4   np.random.seed(19680801)
5   y = np.random.normal(loc=0.5, scale=0.4, size=1000)
6   y = y[(y > 0) & (y < 1)]
7   y.sort()
8   x = np.arange(len(y))
9   plt.figure(1)
10  plt.subplot(221)
11  plt.plot(x, y)
12  plt.yscale('linear')
13  plt.title('linear')
14  plt.grid(True)
15  plt.subplot(222)
16  plt.plot(x, y)
17  plt.yscale('log')
18  plt.title('log')
19  plt.grid(True)
20  plt.subplot(223)
21  plt.plot(x, y - y.mean())
22  plt.yscale('symlog', linthreshy=0.01)
23  plt.title('symlog')
24  plt.grid(True)
25  plt.subplot(224)
26  plt.plot(x, y)
27  plt.yscale('logit')
28  plt.title('logit')
29  plt.grid(True)
30  plt.gca().yaxis.set_minor_formatter(NullFormatter())
31  plt.subplots_adjust(top=0.99, bottom=0.01, left=0.10, right=0.95,
    hspace=0.25,  wspace=0.35)
32  plt.show()
```

範例程式說明

- 1-3 行 import 所需套件。
- 4 行運用 numpy 的 random.seed 方法重設隨機數起始種子值。
- 5-7 行運用 numpy 的 random.normal 方法取得標準化的亂數資料串列，並將之排序後，存成 y 串列。

- 8 行運用 numpy 的 arange 方法取得相等間隔資料串列。
- 9 行設定 figure 物件。
- 10-14 行繪製第一個子圖，此為線性 scale 的長條圖。
- 15-19 行繪製第二個子圖，此為 log scale 的長條圖。
- 20-24 行繪製第三個子圖，此為 symmetric log scale 的長條圖。
- 25-29 行繪製第四個子圖，此為 logit scale 的長條圖。
- 30 行設定較小的 y 軸刻度標記。
- 31 行調整子圖的外觀使個子圖大小接近。
- 32 行要求顯示圖形。

▶▶ 輸出結果



4-3-2 CSV 檔案繪製圖表

▶▶ 範例程式 **E4-3-2-1.py**

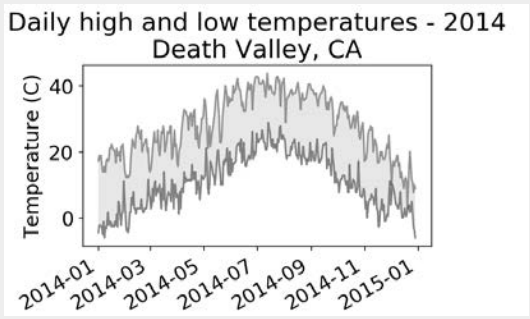
```
1  import csv
2  from datetime import datetime
3  from matplotlib import pyplot as plt
```

```
4   filename = 'death_valley_2014.csv'
5   with open(filename) as f:
6       reader = csv.reader(f)
7       header_row = next(reader)
8       dates, highs, lows = [], [], []
9       for row in reader:
10          try:
11              current_date = datetime.strptime(row[0], "%Y-%m-%d")
12              high = (int(row[1])-32)*5/9
13              low = (int(row[3])-32)*5/9
14          except ValueError:
15              print(current_date, 'missing data')
16          else:
17              dates.append(current_date)
18              highs.append(high)
19              lows.append(low)
20  fig = plt.figure(dpi=128, figsize=(10, 6))
21  plt.plot(dates, highs, c='red', alpha=0.5)
22  plt.plot(dates, lows, c='blue', alpha=0.5)
23  plt.fill_between(dates, highs, lows, facecolor='blue', alpha=0.1)
24  title = "Daily high and low temperatures - 2014\nDeath Valley, CA"
25  plt.title(title, fontsize=20)
26  plt.xlabel('', fontsize=16)
27  fig.autofmt_xdate()
28  plt.ylabel("Temperature (C)", fontsize=16)
29  plt.tick_params(axis='both', which='major', labelsize=16)
30  plt.show()
```

範例程式說明

- 1-3 行 import 所需套件。
- 4-19 行讀取'death_valley_2014.csv'，建立 dates、highs、slows 三個串列儲存 2014 美國死谷每天高低溫資料，並將之轉換為攝氏溫度。
- 20 行設定 figure 物件。
- 21 行繪製每日高溫折線圖。
- 22 行繪製每日低溫折線圖。
- 23 行在高、低溫折線間塗上顏色。
- 24-29 行設定相關圖形標記。
- 30 行要求顯示圖形。

輸出結果



4-3-3 Numpy 模組應用

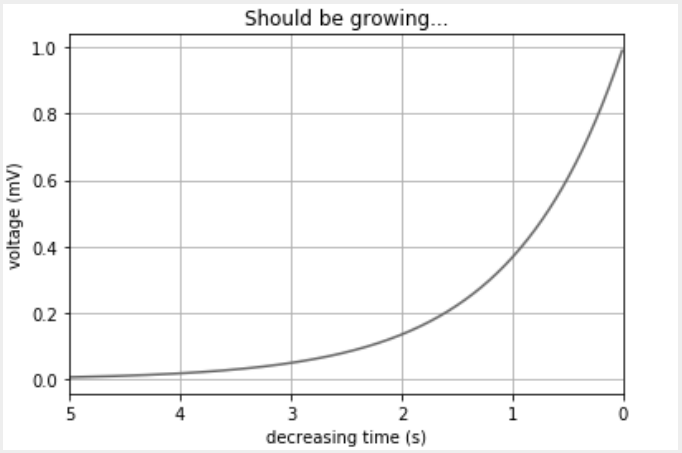
範例程式 **E4-3-3-1.py**

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  t = np.arange(0.01, 5.0, 0.01)
4  s = np.exp(-t)
5  plt.plot(t, s)
6  plt.xlim(5, 0)  # decreasing time
7  plt.xlabel('decreasing time (s)')
8  plt.ylabel('voltage (mV)')
9  plt.title('Should be growing...')
10 plt.grid(True)
11 plt.show()
```

範例程式說明

- 1-2 行 import 所需套件。
- 3 行運用 numpy 的 arange 方法取得間隔大小相同的一個串列，存為 t 物件。
- 4 行運用 numpy 的 exp 方法取得 t 串列物件對應的自然指數函數值，存為 s 物件。
- 5 行以 t 與 s 繪製折線圖。
- 6-8 行設定座標軸的限制與標記。
- 9 行設定整個圖的標題。
- 10 行要求顯示格線。
- 11 行要求顯示圖形。

輸出結果



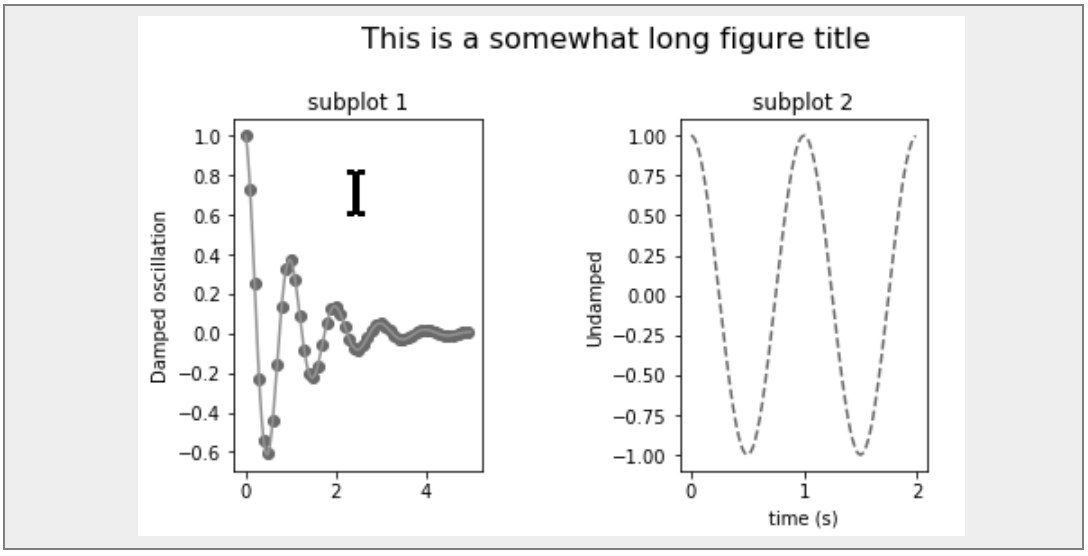
範例程式 **E4-3-3-2.py**

```
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
import numpy as np
def f(t):
    s1 = np.cos(2*np.pi*t)
    e1 = np.exp(-t)
    return s1 * e1
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
t3 = np.arange(0.0, 2.0, 0.01)
plt.subplot(121)
plt.plot(t1, f(t1), 'o', t2, f(t2), '-')
plt.title('subplot 1')
plt.ylabel('Damped oscillation')
plt.suptitle('This is a somewhat long figure title',
fontsize=16)
plt.subplot(122)
plt.plot(t3, np.cos(2*np.pi*t3), '--')
plt.xlabel('time (s)')
plt.title('subplot 2')
plt.ylabel('Undamped')
plt.subplots_adjust(left=0.01, wspace=0.8, top=0.8)
plt.show()
```

範例程式說明

- 1-3 行 import 所需套件。
- 4-7 行定義函數 f，能對輸入值計算 cos 與 exp 函數值，將之相乘後回傳。
- 8-10 行運用 numpy 的 arange 方法取得間隔大小相同的三個串列，存為 t1、t2、t3 物件。
- 11-15 行繪製第一個子圖，以 t1、f(t1)與 t2、f(t2)繪製折線圖，並處理相關圖形標記。
- 16-21 行繪製第二個子圖，以 t3、2*np.pi*t3 繪製折線圖，並處理相關圖形標記。
- 22 行要求顯示圖形。

輸出結果



4-3-4　隨機數的應用

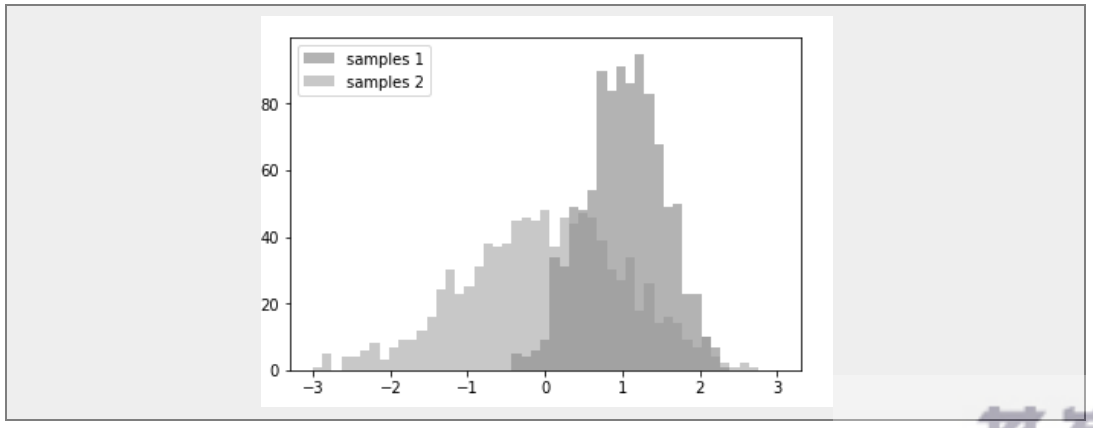
範例程式 E4-3-4-1.py

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  samples_1 = np.random.normal(loc=1, scale=.5, size=1000)
4  samples_2 = np.random.standard_t(df=10, size=1000)
5  bins = np.linspace(-3, 3, 50)
6  plt.hist(samples_1, bins=bins, alpha=0.5, label='samples 1')
7  plt.hist(samples_2, bins=bins, alpha=0.5, label='samples 2')
8  plt.legend(loc='upper left')
9  plt.show()
```

範例程式說明

- 1-2 行 import 所需套件。
- 3 行運用 numpy 的 random.normal 方法取得標準分配的函數值 1000 個，存為 samples_1 物件。
- 4 行運用 numpy 的 standard_t 方法取得標準 t 分配的函數值 1000 個，存為 samples_2 物件。
- 5 行運用 numpy 的 arange 方法取得間隔大小相同的串列，存為 bins 物件。
- 6-7 行針對 samples_1 與 samples_2 繪製兩個直方圖，並處理相關圖形標記。
- 8 行指定圖例位置。
- 9 行要求顯示圖形。

輸出結果



範例程式 E4-3-4-2.py

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  samples_1 = np.random.normal(loc=1, scale=.5, size=1000)
4  samples_2 = np.random.standard_t(df=10, size=1000)
5  bins = np.linspace(-3, 3, 50)
6  plt.scatter(samples_1, samples_2, alpha=0.2);
7  plt.show()
```

範例程式說明

- 1-2 行 import 所需套件。
- 3 行運用 numpy 的 random.normal 方法取得標準分配的函數值 1000 個，存為 samples_1 物件。
- 4 行運用 numpy 的 standard_t 方法取得標準 t 分配的函數值 1000 個，存為 samples_2 物件。
- 5 行運用 numpy 的 arange 方法取得間隔大小相同的串列，存為 bins 物件。
- 6 行針對 samples_1 與 samples_2 繪製兩個散佈圖。
- 7 行要求顯示圖形。

輸出結果