Matplotlib畫圖介紹

- matplotlib.pyplot module提供許多繪圖指令(與matlab語法非常相近)，這邊介紹繪製折線圖相關的指令 Reference
- 在使用前，一樣要先import module進來

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
```

畫折線圖

plt.plot(x, y, 'style_code')

- x: position along the x-axis (if missing, than use integers 0,1,2,3.... as default)
- y: position along the y-axis
- 'style code': abbreviation code for line color, line style, and marker style (see tables below)
  還有很多可以設定的選項，詳細請參考

line style and marker code

顏色代碼	
b	藍色(Blue)
c	青藍色(Cyan)
g	綠色(Green)
k	黑色(Black)
m	洋紅色(Magenta)
r	紅色(Red)
w	白色(White)
y	黃色(Yellow)

線條樣式代碼	
-	實線(預設值)
--	虛線
:	點線
-.	點虛線

符號樣式代碼			
o	圓形	s	方形
+	加號	d	菱形
x	叉號	p	五角形
*	星號	.	點
^	向上三角形		
v	向下三角形		
<	向左三角形		
>	向右三角形		

更多使用方法:

- line style
- color code
- marker style

可以使用不同的畫圖風格

plt.style.use('ggplot')
所有作圖風格呈現效果

plt.show display figure 顯示圖形

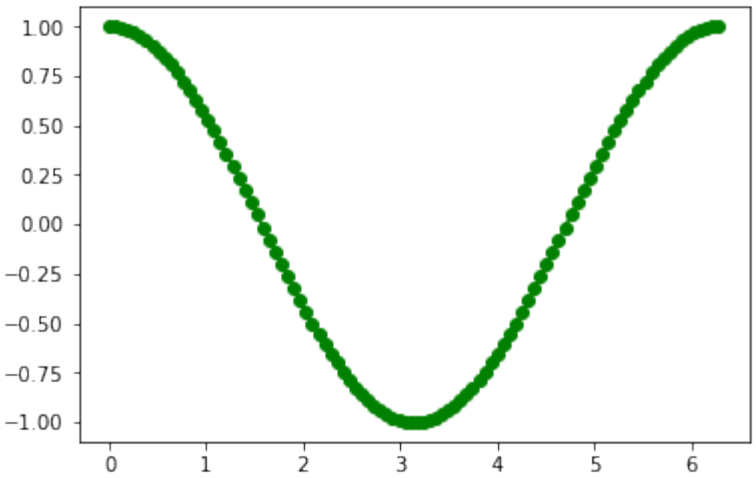
plt.show()

- display/save the current figure in window, and start the next plotting-related command from blank

plt.savefig save figure into file 儲存圖形

- plt.savefig('filename')
- filename: 檔名，可存成.png, .jpg, .gif, .svg, .eps等圖檔格式

In [4]:
```
xx = np.linspace(0, 2 * np.pi, 100) # 0~2pi間 產生100個數字
yy = np.cos(xx)
#plt.figure(figsize=(3, 3), dpi=100) # figsize=(10,10) 為size大小, dpi為解析度
#plt.style.use('ggplot') # 使用不同的style

#####
#fig = plt.gcf()
#fig.set_size_inches(18.5, 10.5) #可以改figure size
#####

plt.plot(xx, yy, 'g-o')
plt.show()
#plt.savefig('fig1.png')
```



一張圖有多條線 Plot multiple lines in one figure

把要畫的x座標陣列與y座標陣列一組一組建立好

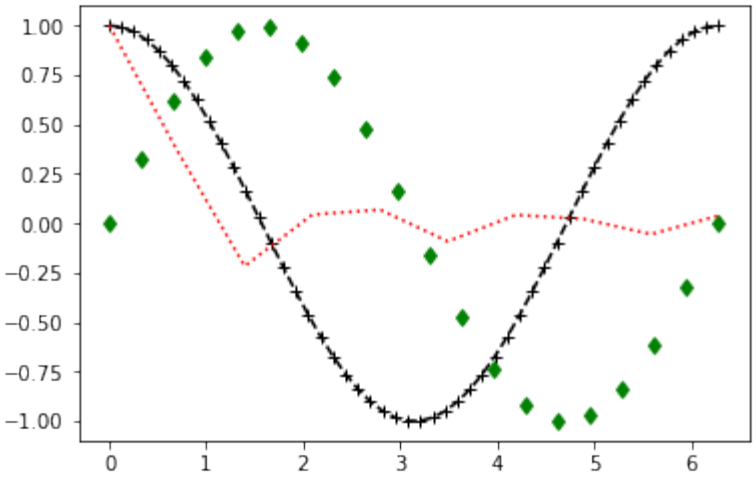
每一組x與y陣列長度要相同，但是各組之間可以不同

依序plot各組x,y陣列（python會自動使用不同顏色的多條曲線，也可自行設定線條顏色樣式）

每組都畫完之後，再用plt.show()顯示在同一張圖中

In [5]:
```
# Example: plot cos(x), sin(x), and sinc(x) in the same figure
```

```
x1 = np.linspace(0, 2*np.pi)    # 50x1 array between 0 and 2*pi
y1 = np.cos(x1)
x2 = np.linspace(0, 2*np.pi,20) # 20x1 array
y2 = np.sin(x2)
x3 = np.linspace(0, 2*np.pi,10) # 10x1 array
y3 = np.sinc(x3)

plt.plot(x1, y1, 'k--+')   # black dashed line, with "+" markers
plt.plot(x2, y2, 'gd')     # green dimonds (no line)
plt.plot(x3, y3, 'r:')     # red dotted line (no marker)
plt.show()
```



Add title, axis lables 加標題、座標軸文字

利用以下指令，可以在圖的不同位置加上文字

plt.title('string',fontsize=n): 加標題

plt.xlabel('string',fontsize=n): 加x軸文字

plt.ylabel('string',fontsize=n): 加y軸文字

plt.text(xp, yp, 'string',fontsize=n): 在圖上xp, yp的位置寫字

在這些指令中，要顯示的字串要用"夾起

在字串中插入希臘字母、上標、下標

\α

\$\alpha \beta$　　（$$之間的空格會被忽略）

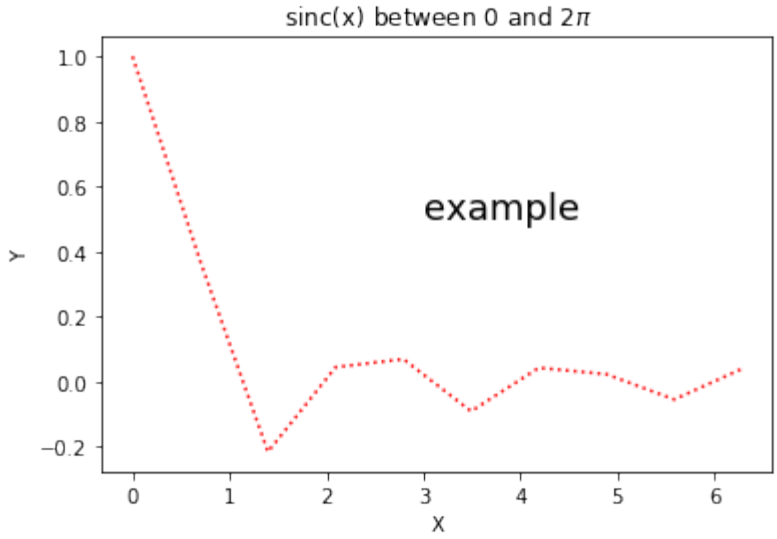
\$\alpha$ β　　（字母間如果要插空格，必須分別用$$夾住）

上標superscript：\T^{20}

下標subscript：　\T_{20}

In [6]:
```
#Example of adding title, axies labels, and text string
```

```
plt.plot(x3, y3, 'r:')     # red dotted line (no marker)
plt.xlabel('X')
plt.ylabel('Y')
plt.title(r'sinc(x) between 0 and 2$\pi$')
plt.text(3,0.5,'example',fontsize=18)
plt.show()
```



Change the range of axes and tick marks 設定座標軸範圍、刻度

自訂座標軸刻度與刻度上的標記文字

plt.xticks([array of tick marks],[list of tick mark labels]) ： x軸刻度、標記

plt.yticks([array of tick marks],[list of tick mark labels]) ： y軸刻度、標記

例如： plt.xticks(np.linspace(0,5,3),[' zero ', ' 5/2', ' five'])

注意：plt.xticks要寫在plt.xlim或plt.ylim之前，否則會被xlim的設定蓋過

座標軸範圍

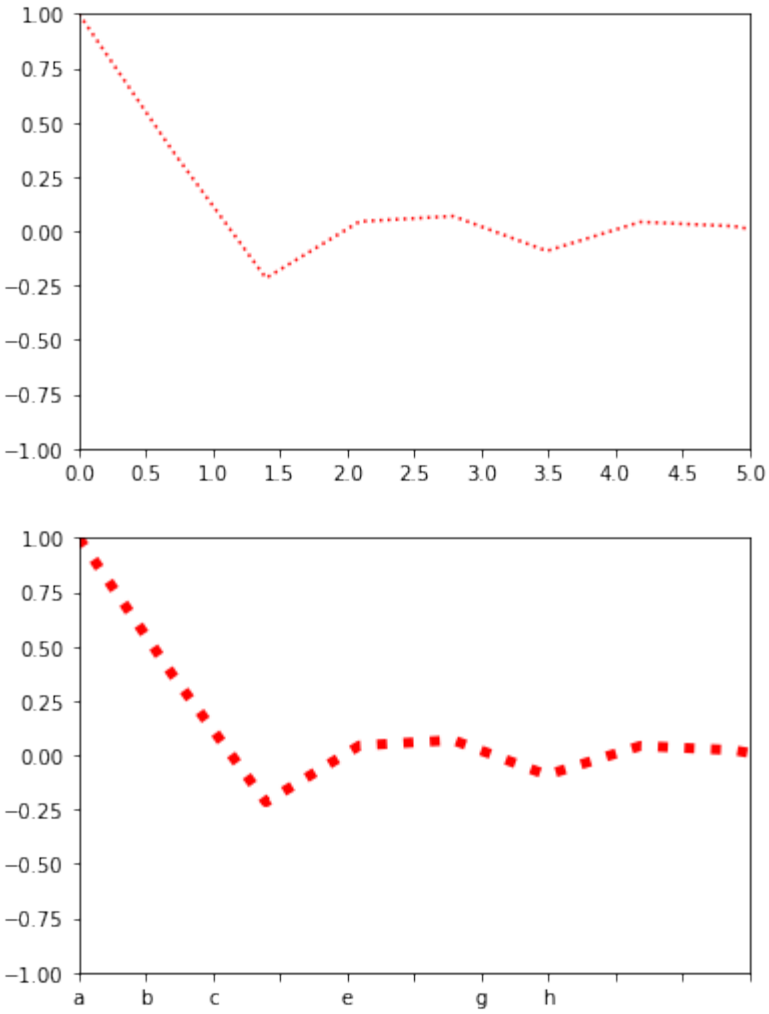
plt.xlim([xmin,xmax]) x軸的上、下界

plt.ylim([ymin,ymax]) y軸的上、下界

注意：**Python** 會自動根據上下界的順序自動反轉座標，例如：**plt.xlim([xmax,xmin])** 會自動讓**X**軸高值在左側、低值在右側

In [7]:
```
# Example of setting axes range and tick marks
plt.plot(x3, y3, 'r:')     # red dotted line (no marker)
plt.xticks(np.linspace(0,5,11))
plt.xlim([0,5])
plt.ylim([-1,1])
plt.show()
```

```
# change x tick mark lables
plt.plot(x3, y3, 'r:', linewidth=5.0)     # red dotted line (no marker), a
dd linewidth
plt.xticks(np.linspace(0,5,11),['a','b','c','','e','','g','h'])
plt.xlim([0,5])
plt.ylim([-1, 1])
plt.show()
```



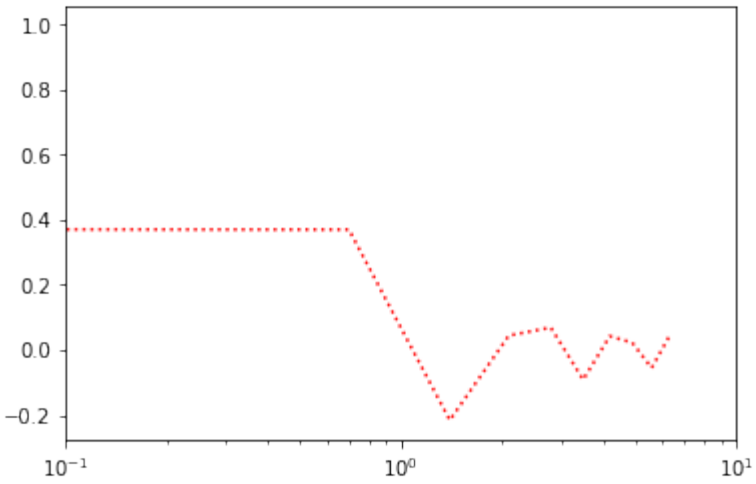
Use logarithmic axes 使用對數座標軸

Python 預設使用線性座標(linear)，要調整為對數座標(log)使用:

plt.xscale('log') 對數x軸

plt.yscale('log') 對數y軸

In [8]:
```
# Example of setting log axis
plt.plot(x3, y3, 'r:')     # red dotted line (no marker)
plt.xlim([0.1,10])
plt.xscale('log')          # log x-axis
plt.show()
```



plt.legend -- add a legend 加上圖例說明

圖中有許多線條的時候，在圖上加入圖例說明（**legend**）會更清楚瞭解每條線的意義。

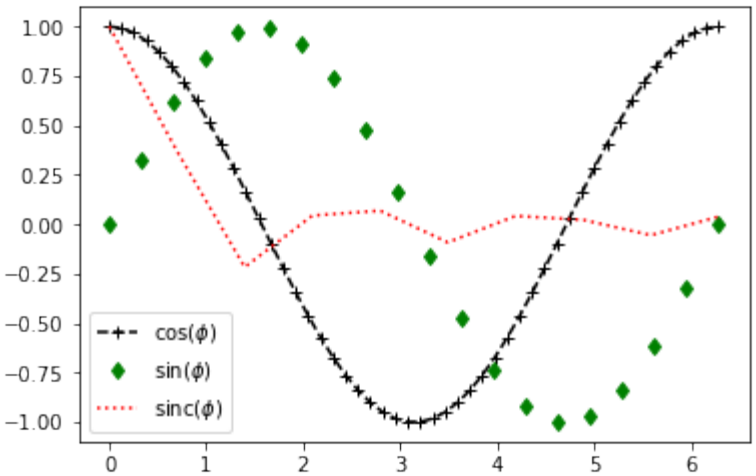
plt.legend([list of legend text],loc='location')

'location'設定legend在圖中的位置:

- 'best',
- 'upper right', 'upper left', 'lower left', 'lower right', 'right', 'center left', 'center right', 'lower center', 'upper center', 'center'

plt.legend還有許多可以調整的設定，詳細請參考

In [9]:
```
#example of adding legend
plt.plot(x1, y1, 'k--+')   # black dashed line, with "+" markers
plt.plot(x2, y2, 'gd')     # green dimonds (no line)
plt.plot(x3, y3, 'r:')     # red dotted line (no marker)
plt.legend(['cos($\phi$)','sin($\phi$)','sinc($\phi$)'])
plt.show()
```



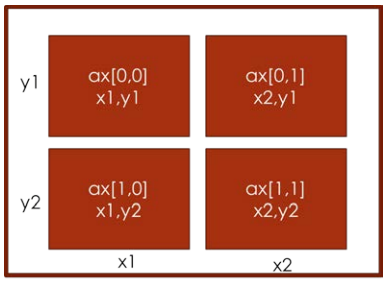
plt.subplot -- split window to show multiple figures 切割視窗顯示多張圖

在同一個視窗畫出mxn張小圖

f,ax=**plt.subplots**(m, n, sharex='...', sharey='...') 分割畫面並設定共用座標軸

- m: 垂直方向的圖形數目
- n: 水平方向的圖形數目
- sharex & sharey: 圖形之間是否要共用x軸或y軸，'...'內可填入的選項有
  - 'col': 直行的圖共用
  - 'row': 橫列的圖共用
  - 'all': 所有的小圖都共用
  - 'false': 不共用 (default setting)

ax[a,b].plot(...) 會在[a,b]區塊畫圖，排列方式如下



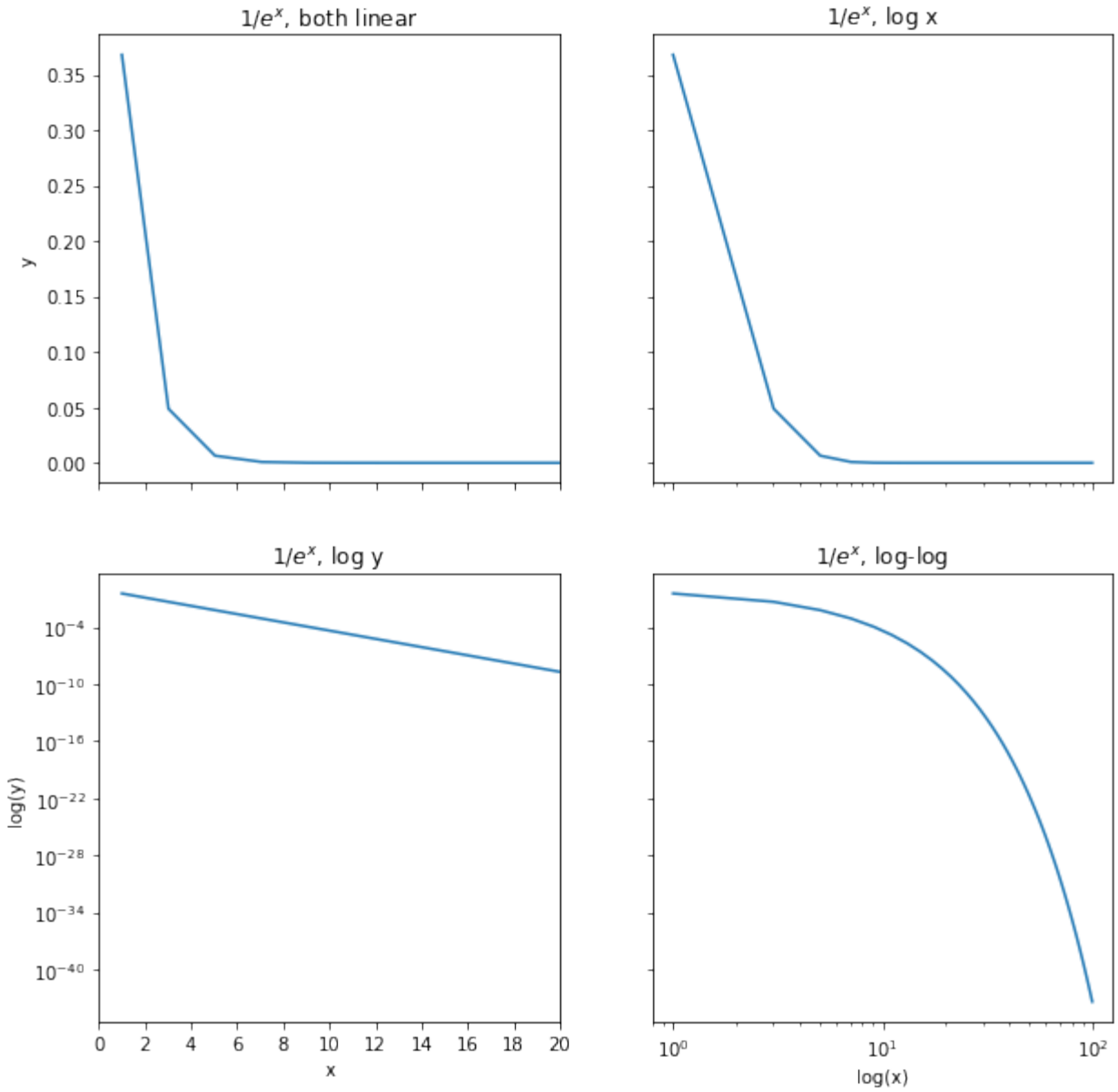
ax[a,b].set_title('....') 在 [a,b]區塊加標題

ax[a,b].set_xlabel('....') 在 [a,b]區塊加x軸文字

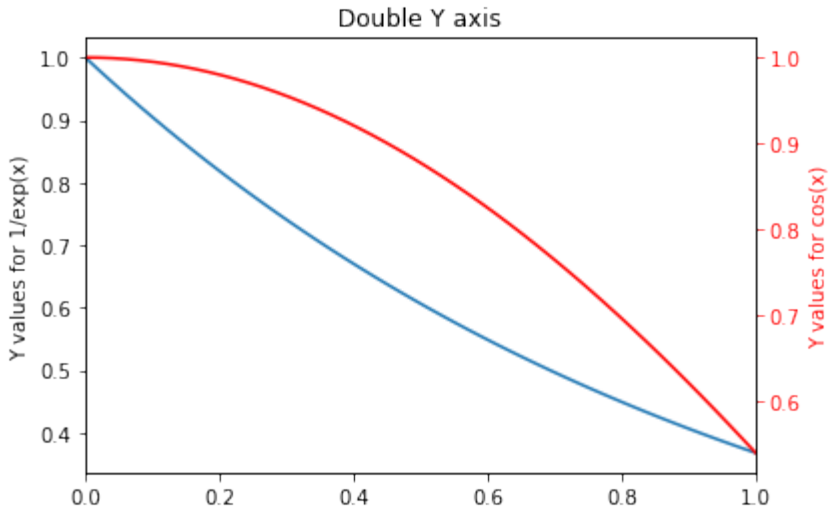
In [10]:
```
# example of plot 2x2 figures in the same window

import numpy as np
import matplotlib.pyplot as plt

# generate x and y array
x=np.linspace(1,100)
y=1/np.exp(x)
```

In [11]:
```
# split window
f, ax = plt.subplots(2,2,sharex='col',sharey='row', figsize=(10, 10))

#upper left
ax[0,0].plot(x,y)
ax[0,0].set_title(r'1/$e^{x}$, both linear')
ax[0,0].set_ylabel('y')

#upper right
ax[0,1].plot(x,y)
ax[0,1].set_title(r'1/$e^{x}$, log x')
```

```
#lower left
ax[1,0].plot(x,y)
ax[1,0].set_yscale('log')
ax[1,0].set_title(r'1/$e^{x}$, log y')
ax[1,0].set_xlabel('x')
ax[1,0].set_ylabel('log(y)')
ax[1,0].set_xticks(np.linspace(0,20,11))
ax[1,0].set_xlim([0,20])

#lower right
ax[1,1].plot(x,y)
ax[1,1].set_xscale('log')
ax[1,1].set_title(r'1/$e^{x}$, log-log')
ax[1,1].set_xlabel('log(x)')

#save figure and show
#plt.savefig('2x2_subplot.png')
plt.show()
```



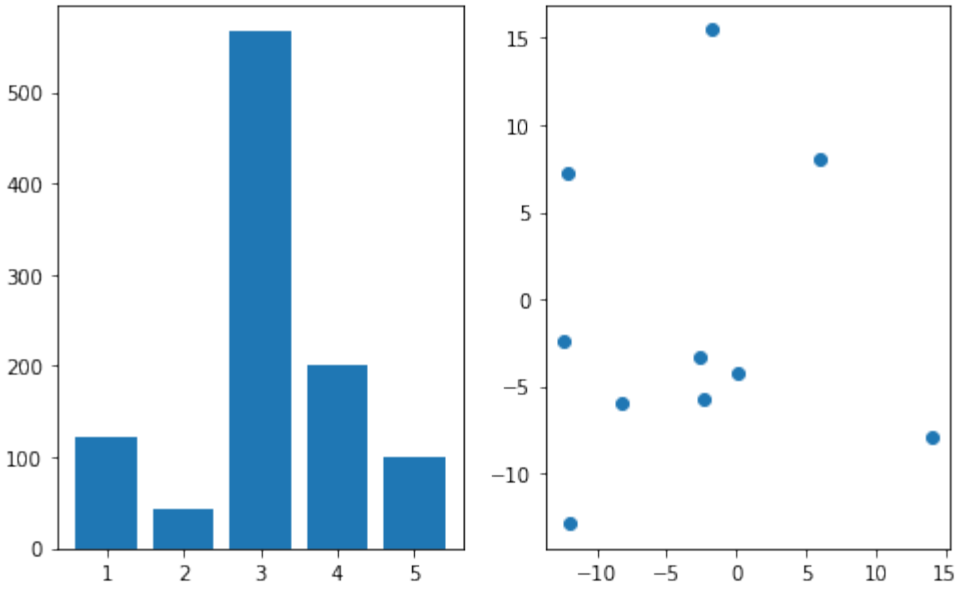
In [12]:
```
### 右側坐標軸

x = np.linspace(0, 1)
y1 = 1/np.exp(x)
y2 = np.cos(x)

fig = plt.figure()

ax1 = fig.add_subplot(111)
ax1.plot(x, y1)
ax1.set_ylabel('Y values for 1/exp(x)')
ax1.set_title("Double Y axis")

ax2 = ax1.twinx()  # this is the important function
ax2.plot(x, y2, 'r')
ax2.set_xlim([0, 1])
ax2.set_ylabel('Y values for cos(x)', color='red')
ax2.tick_params(axis='y', colors='red')
ax2.set_xlabel('Same X')

plt.show()
```



長條圖、散佈圖

In [13]:
```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure(figsize=(8,5))
plt.subplot(121)
x=[1,2,3,4,5]
y=[123,44,567,201,100]

plt.bar(x,y,align='center')

## 散佈圖
x = np.random.randn(10)*10
y = np.random.randn(10)*10

plt.subplot(122)
```

```
plt.scatter(x,y)

plt.show()
```



2D plot

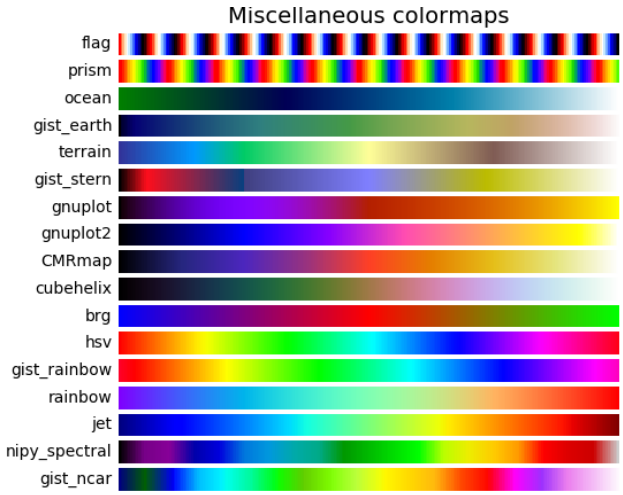
plt.contour 畫等值線

CS=plt.contour(x,y,z,cmap=...,levels=...,colors=...)

- x: array (can be 1D or 2D) of x to evaluate z
- y: array (can be 1D or 2D) of y to evaluate z
- z: 2-D array of data to plot contour
- cmap: color map (remember to import matplotlib.cm)
- levels: contour levels
- colors: line color

plt.contour: change color map 換等值線色階

CS=plt.contour(x,y,z,cmap=...,levels=...,colors=...)



cmap: color map (remember to import matplotlib.cm)

plot.label:add contour label 加等值線標籤

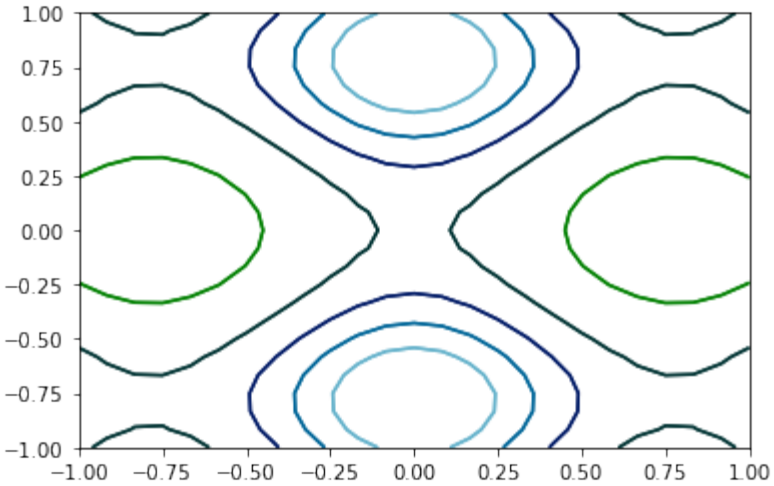
plt.clabel(CS, inline=1, fontsize=10)

- CS: plotting object
- inline: location of label
  - 1=lines do not cross over label text (default)
  - 0=lines cross over label text
- fontsize: size of label text

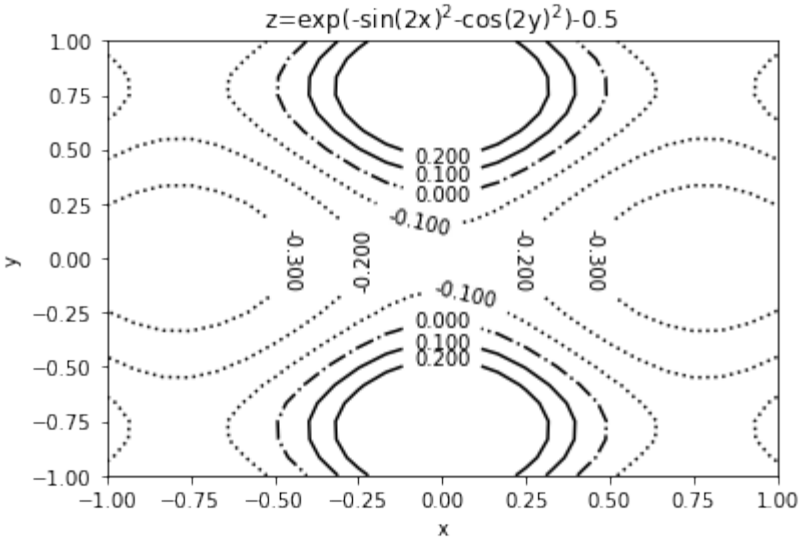
In [14]:
```
import matplotlib.cm as cm
import numpy as np
import matplotlib.pyplot as plt
# Now we use finer grid (larger arrays) to continue our example:
x=np.linspace(-1,1,25)
y=np.linspace(-1,1,25)
xx,yy=np.meshgrid(x,y)
print(xx.shape)

# compute function z using the grids constructed by xx and yy
z=np.exp(-np.sin(2*xx)**2-np.cos(2*yy)**2)-0.5
print(z.shape)
```

```
(25, 25)
(25, 25)
```

In [17]:
```
# simple plot
CS = plt.contour(x, y, z, cmap=cm.ocean)
plt.show()
```

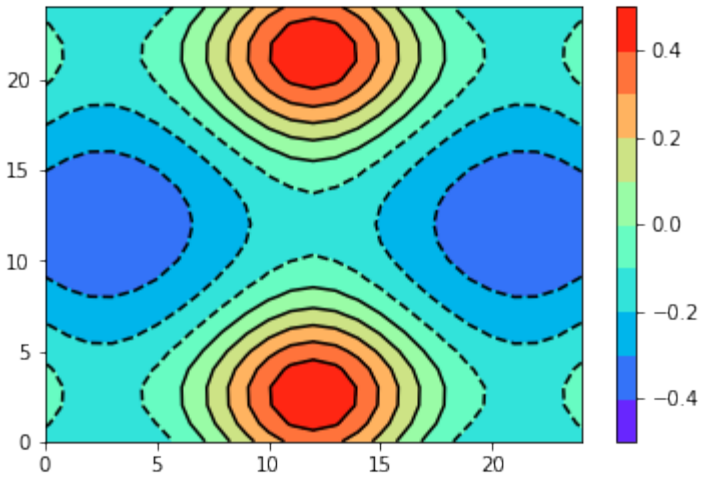


In [15]:
```
## negative values dotted, 0 dashdot, positive value solid
CS0 = plt.contour(x, y, z, levels=np.arange(-0.3, 0, 0.1), linestyles='dotted', colors='k')
plt.clabel(CS0, inline=1, fontsize=10)

CS1 = plt.contour(x, y, z, levels=[0], linestyles='dashdot', colors='k')
plt.clabel(CS1, inline=1, fontsize=10)

CS2 = plt.contour(x, y, z, levels=np.arange(0.1, 0.3, 0.1), linestyles='solid', colors='k')
plt.clabel(CS2, inline=1, fontsize=10)

##
plt.title(r'z=exp(-sin(2x)$^2$-cos(2y)$^2$)-0.5')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



plt.contourf: filled contour 色塊等值線（等值線內填入相同顏色）

In [19]:
```
plt.contour(z, levels= np.linspace(-0.5,0.5,11), colors='k')
CS = plt.contourf(z, levels= np.linspace(-0.5,0.5,11), cmap=cm.rainbow)
plt.colorbar()
plt.show()
```



部分內容是**from** 維婷老師 大二程設課程

Sample plots in Matplotlib

Here you'll find a host of example plots with the code that generated them