用python畫圖 - matplotlib –

要用matplotlib來做數據處理和畫圖的話，基本上最好有裝numpy, scipy, 和 matplotlib

第一步：畫二維數據圖:

假設要畫sin圖....

#coding=utf-8

import matplotlib.pyplot as plt

import numpy as np

#準備數據 ... 假設我要畫一個sin波 從0~180度

x = np.arange(0,360)

y = np.sin(x \* np.pi / 180.0)

# 開始畫圖

# 設定要畫的的x,y數據list....

plt.plot(x,y)

plt.grid(True)#幫圖形加上格線

# 設定圖的範圍, 不設的話，系統會自行決定

plt.xlim(-30,390)

plt.ylim(-1.5,1.5)

# 照需要寫入x 軸和y軸的 label 以及title

plt.xlabel("x-axis")

plt.ylabel("y-axis")

plt.title("The Title")

# 在這個指令之前，都還在做畫圖的動作

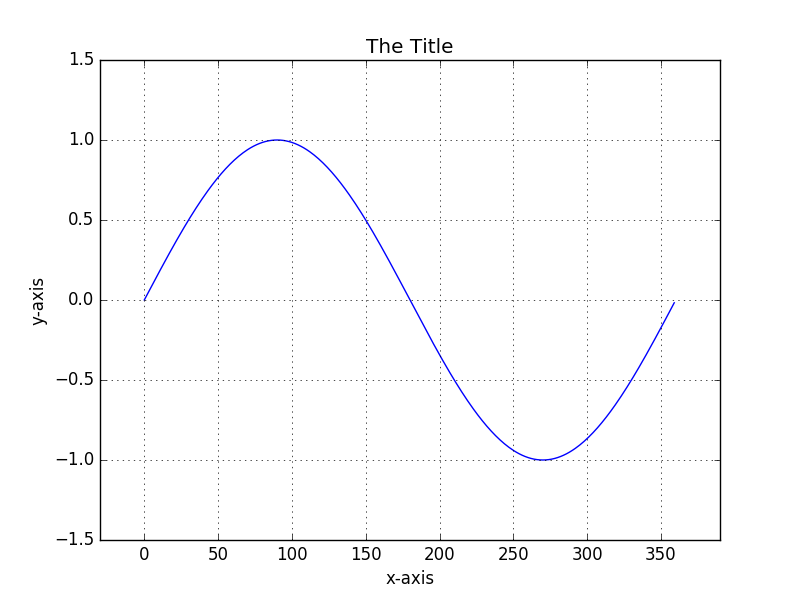
# 這個指令算是 "秀圖"

plt.show()

# 如果要存成圖形檔:

# 把 pyplot.show() 換成下面這行:

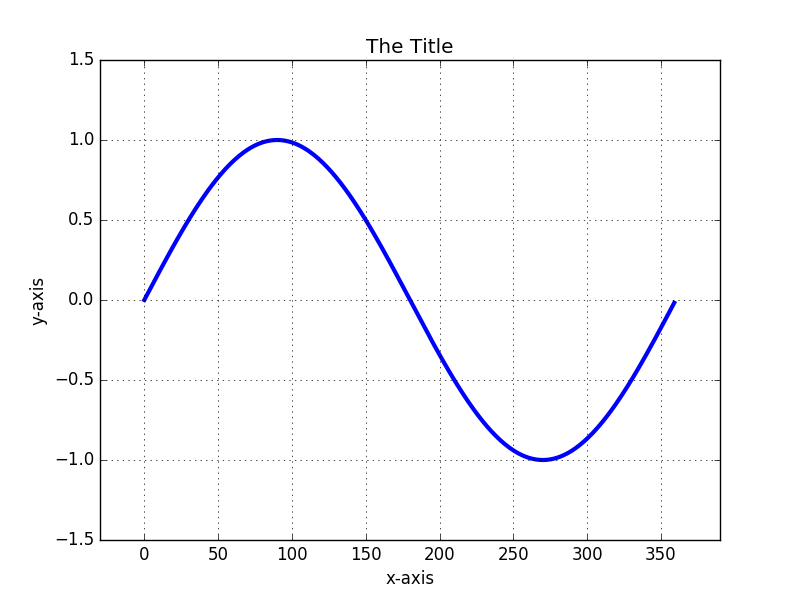
plt.savefig("filename.png",dpi=300,format="png")



而調整線寬的方式就是再多加一個lw的參數:

plt.plot(x,y,lw=3)#這是利用lw參數來決定線的大小

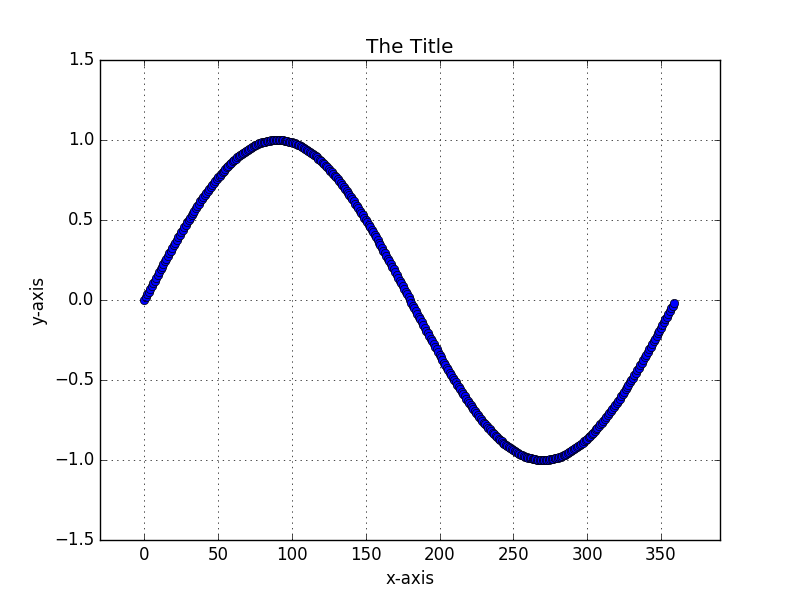
出圖後



如果不想畫線而想畫點的話(改變linestyle)，就在把 plt.plot(x,y) 這一行的引數多加"o"，變成下面這行:

plt.plot(x,y,"o")#這是利用字串決定點的種類

然後就會得到下圖:



當然還有很多其他的style可選，像是"ro"就是紅點，"--" 會是虛線, "r--" 就是紅色虛線。

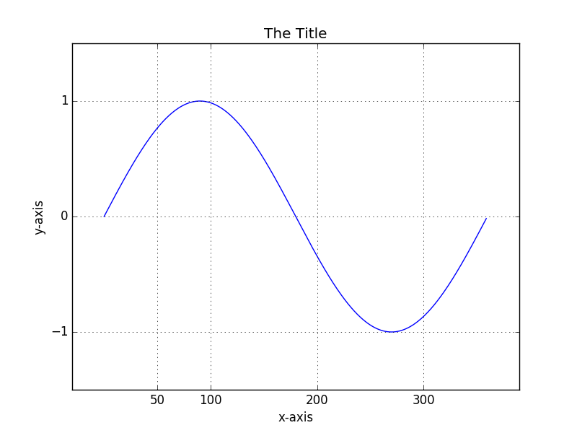
注意：字串決定法或參數決定法只能選一種

除此之外還可以自設座標軸的間隔，如果加上

plt.xticks([50,100,200,300])

plt.yticks([-1,0,1])

可得到下圖



第二步：畫多條線

很簡單...想畫幾行就多加幾行plt.plot(....)

#coding=utf-8

import matplotlib.pyplot as plt

import numpy as np

x1 = np.arange(0,360,5)

y1 = np.sin(x1 \* np.pi / 180.0)

x2 = x1

y2 = np.cos(x2 \* np.pi / 180.0)

y3 = np.tan(x1 \* np.pi / 180.0)

plt.grid(True)

plt.plot(x1,y1,lw=3,label="line1") #label代表這條線的名稱

plt.plot(x2,y2,"ro",label="line2") # k:黑色 m:紫色 c:青藍色 y:黃色 g:綠色

#注意不可以”ro”字串 和 lw參數混在一起用

#所以不可寫成plt.plot(x2,y2,"ro",lw=3,label="line2")

#同理字串意不可和color, linestyle, marker 參數混在一起用

plt.plot(x1,y3,"y--",lw=5,label="line3")

plt.ylabel("y\_label")

plt.xlabel("x\_label")

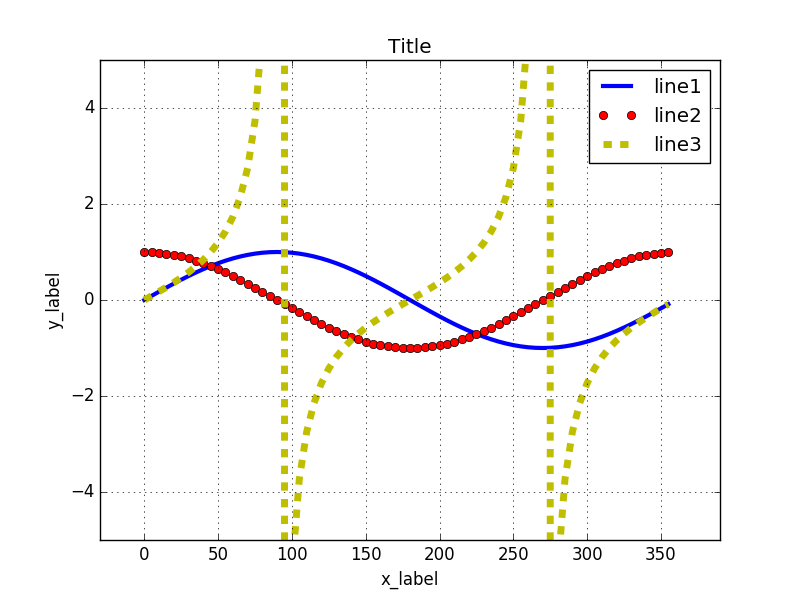
plt.title("Title")

plt.legend()#印出圖例

plt.xlim(-30,390)

plt.ylim(-5,5)

plt.show()



如果想把sin 圖和cos圖分別畫在上下兩張圖的話，就要用plt.subplot

#coding=utf-8

import matplotlib.pyplot as plt

import numpy as np

x1 = np.arange(0,360,5)

y1 = np.sin(x1 \* np.pi / 180.0)

x2 = x1

y2 = np.cos(x2 \* np.pi / 180.0)

plt.grid(True)

plt.subplot(211) #代表接下來要畫一號圖

#以下的設定只會對一號圖起作用

plt.plot(x1,y1,lw=3, color="g",label="line1")

#這裡一定要加上plt.legend()代表對一號圖顯示圖例，否則一號圖會不顯示圖例

plt.legend(loc="lower right") #loc代表圖例的位置

plt.subplot(212) #代表接下來要畫二號圖

#以下的設定只會對二號圖起作用

plt.plot(x2,y2,"ro",label="line2")

plt.ylabel("y\_label")

plt.xlabel("x\_label")

plt.title("Title")

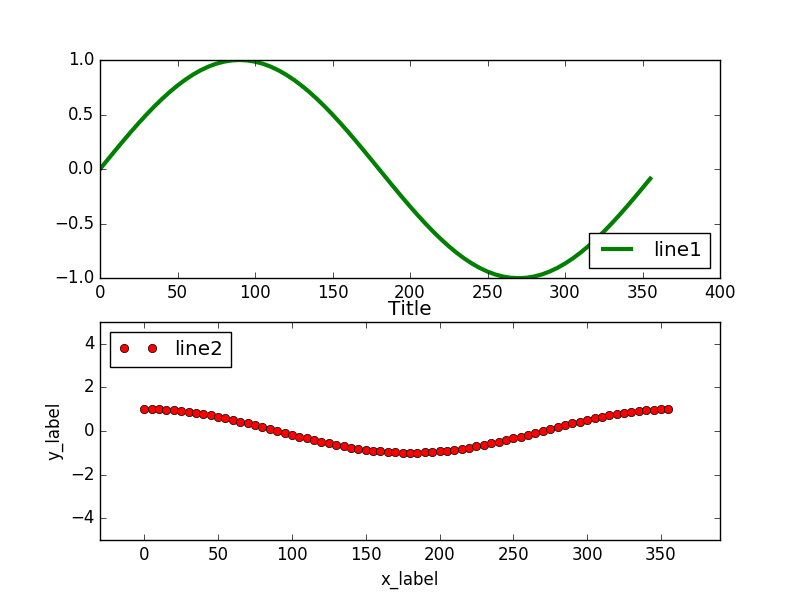
#二號圖顯示圖例

plt.legend(loc="upper left")

plt.xlim(-30,390)

plt.ylim(-5,5)

plt.show()



plt.subplot() 這個指令就是拿來畫多張圖用的指令。

而引入的參數是一個三位數的號碼ex: 212

前兩位號碼是用來分別指定上下、左右 各要畫幾張圖，而最後一位數就是接下來的圖是第幾號圖....(應該有看沒有懂吧...)

拿實例來說明。

假設要畫六張圖，上下兩張圖 左右三張圖 的排列方式: ( 2 x 3 )

圖1 圖2 圖3

圖4 圖5 圖6

可以看到上面的排列，圖的順序是由上而下由左而右來算的。而第三位碼就是拿來指定第幾張圖的。

所以在畫圖1的時候:

plt.subplot(231)

plt.plot(x1,y1)

...

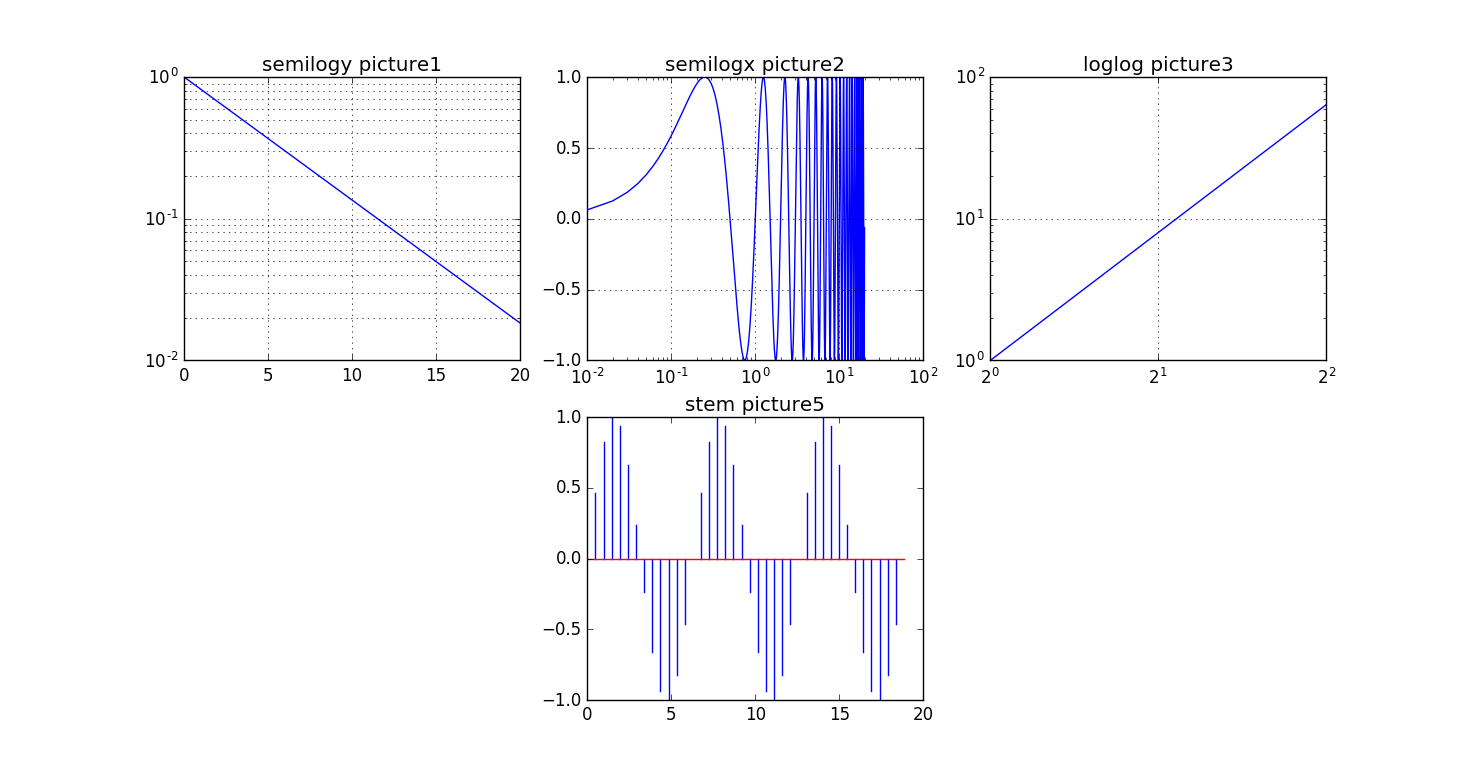
畫圖2的時候:

plt.subplot(232)

plot.plot(x2,y2)

下面給出一個範例

這是該範例的圖形



#coding=utf-8

import numpy as np

import matplotlib.pyplot as plt

t=np.arange(0.01,20,0.01)

# log y axis

plt.subplot(231)

plt.semilogy(t, np.exp(-t/5.0))

plt.title('semilogy picture1')

plt.grid(True,which="both")#新增主要格線和次要格線

# log x axis

plt.subplot(232)

plt.semilogx(t, np.sin(2\*np.pi\*t))

plt.title('semilogx picture2')

plt.grid(True,which=”both”)

# log x and y axis

plt.subplot(233)

x = np.linspace(1, 4, 1000)

plt.loglog(x, x\*\*3, basex=2,basey=10)

plt.grid(True)

plt.title('loglog picture3')

#stem picture

plt.subplot(235)

x = np.linspace(0, 6\*np.pi, 40)

y = np.sin(x)

plt.stem(x, y,markerfmt=" ")

#若不加markerfmt=" "，每根stem的頂端都會有一個圓點

#markerfmt=" "的功用是讓每根stem都為一條直線

#using "" instead of " " will connect the ends of the stem with a line

plt.title("stem picture5")

plt.show()

最後來利用隨機變數的統計來做統計圖與實際pdf來進行比較

#coding=utf-8

import time

import numpy as np

import matplotlib.pyplot as plt

mu=0

std=0.1#標準差

#產生一個10000個點的高斯分佈

n=np.random.normal(0,0.1,100000)

#plt.hist會將一維陣列n畫成長條圖，其中100代表要分成幾條長條圖(100>=1)建議不要設太大，否則正規化圖形容易失真，分割範圍是[min(n[i]) , max(n[j])]

#normed=True代表長條圖的積分值正規化為1

n,bins,c=plt.hist(n,100,normed=True)

#回傳一個tuple 其中bins較重要

#ex. n=[1,2,1,2,1,3,3,3,3,4] 若plt.hist()第二個參數是4代表分成4塊

#所以bins是np.array([1, 1.75, 2.5, 3.25 , 4])

y=np.array(bins)

t1=time.time()

#numpy不建議使用傳統的for迴圈進行陣列運算，效率較差

for i in range(len(y)):

y[i]=1/(np.sqrt(2\*np.pi)\*std)\*np.exp(-((bins[i]-mu)\*\*2)/(2\*(std\*\*2)))

t2=time.time()

#取而代之的是用內建的陣列運算，雖然較不直觀，但運行速度較快

y=1/(np.sqrt(2\*np.pi)\*std)\*np.exp(-(bins-mu)\*\*2/(2\*(std\*\*2)))

t3=time.time()

#印出比較的時間(如果bin很大時

#就會出現別)

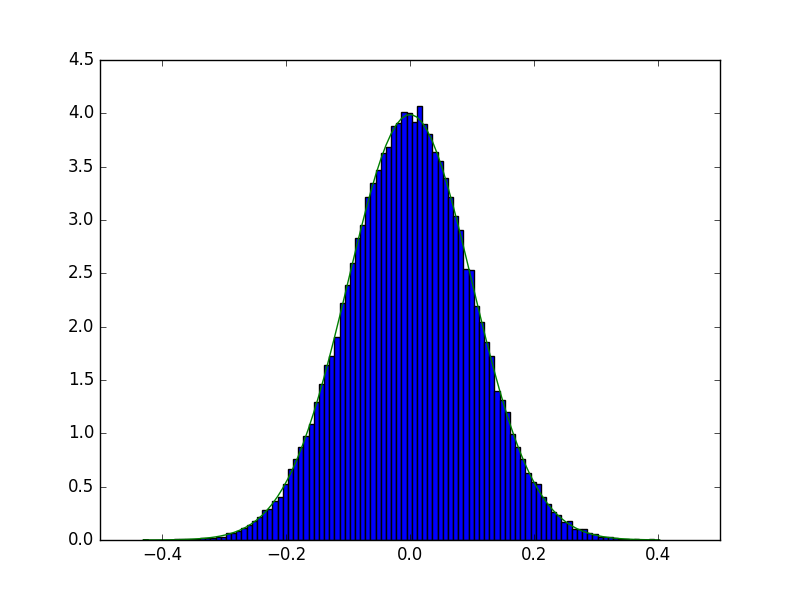
print(t2-t1)

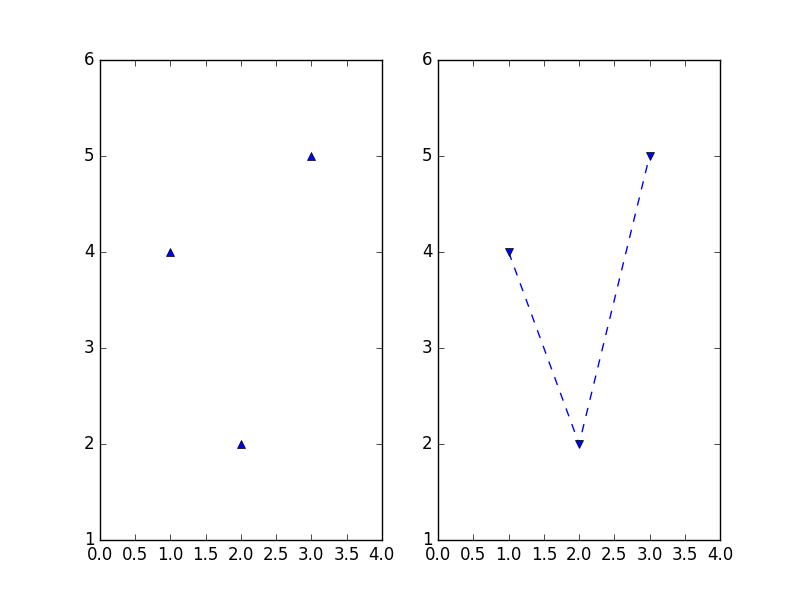
print(t3-t2)

plt.plot(bins,y)

plt.show()

plt.show()



額外補充

#coding=utf-8

import matplotlib.pyplot as plt

import numpy as np

x=[1,2,3]

y=[4,2,5]

plt.subplot(1,2,1)

plt.plot(x,y,linestyle="",marker="^")

plt.xlim(min(x)-1,max(x)+1)

plt.ylim(min(y)-1,max(y)+1)

plt.subplot(1,2,2)

plt.plot(x,y,linestyle="--",marker="v")

plt.xlim(min(x)-1,max(x)+1)

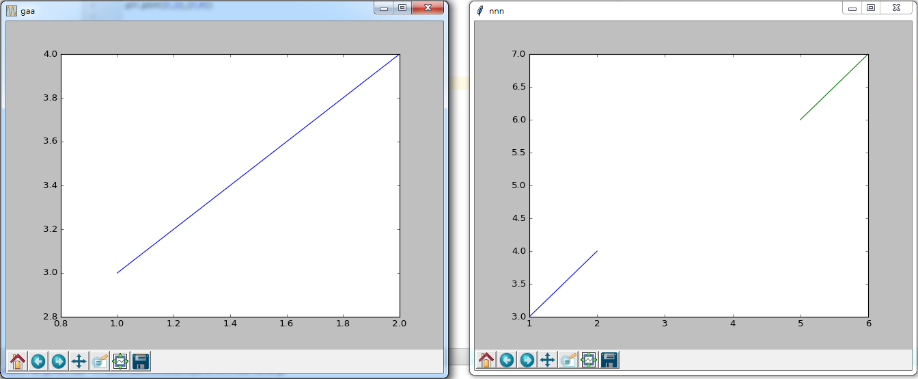
plt.ylim(min(y)-1,max(y)+1)

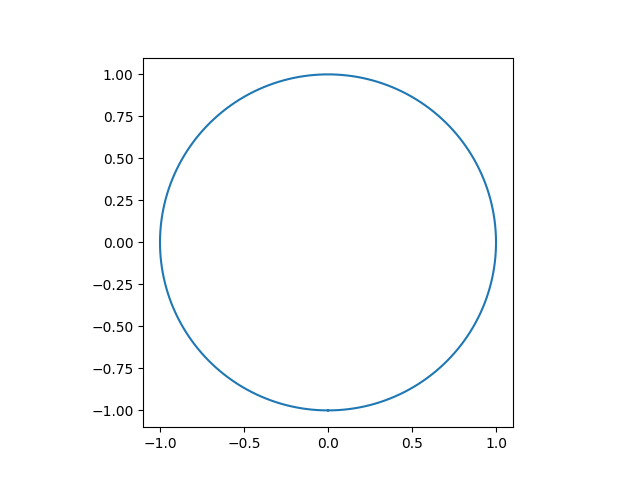
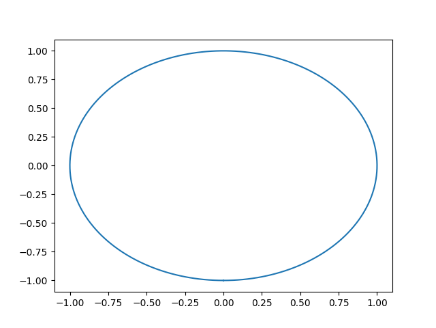
plt.show()

#linestyle代表線的種類 可以有 ””:沒有線 “-”:實線 “--”:虛線 “-.” “:” (預設是實線)

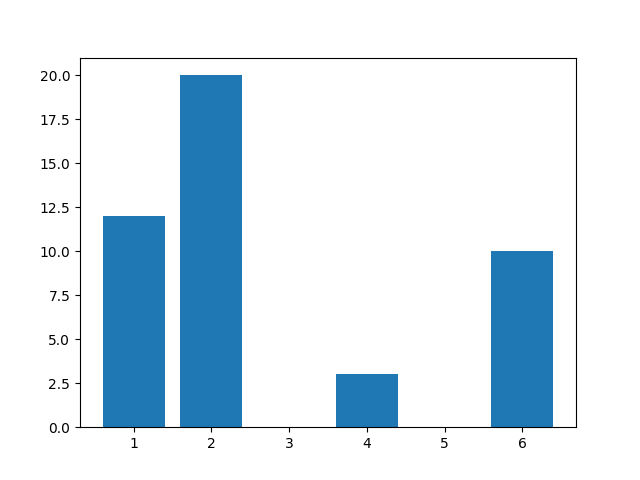
#marker代表點的種類 常見的有 . , o v ^ < > 1 2 3 4 + x

**import** matplotlib.pyplot **as** plt  
  
  
fig = plt.figure(**"gaa"**)*#figure的功用就是建立一個分離的圖案*plt.plot([1,2],[3,4])  
  
fig2 = plt.figure(**"nnn"**)  
plt.plot([1,2],[3,4])  
plt.plot([5,6],[6,7])  
plt.show()



再來介紹畫圓

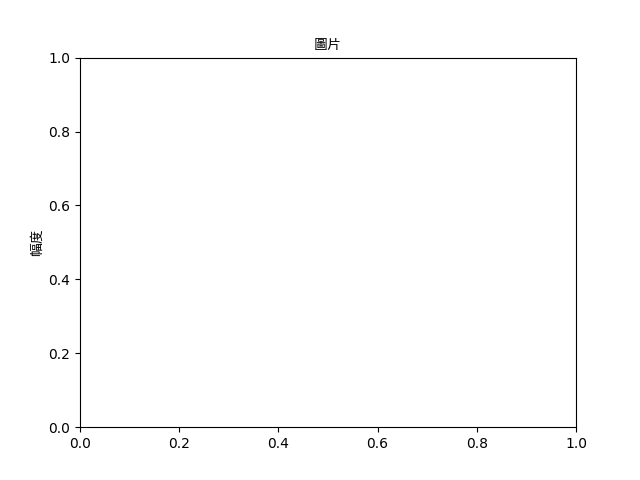
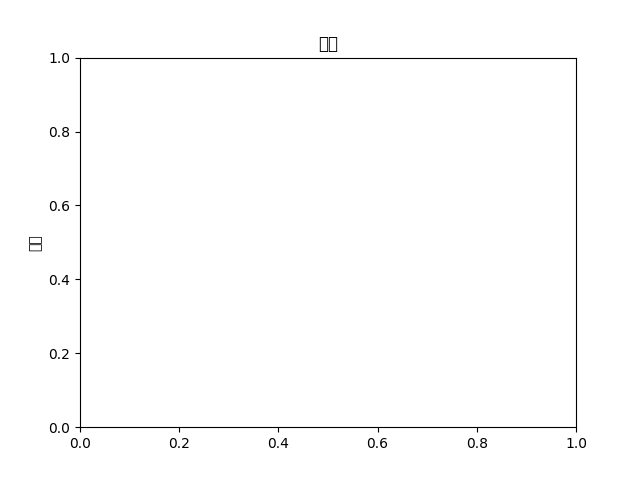
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
  
fig = plt.figure(**"1"**)  
theta = np.linspace(-np.pi, np.pi, 200)  
plt.plot(np.sin(theta), np.cos(theta))  
  
fig = plt.figure(**"2"**)  
plt.axes().set\_aspect(1) *# 使x軸、y軸間隔一致*  
plt.plot(np.sin(theta), np.cos(theta))  
  
plt.show()

再來介紹畫柱狀圖

**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
  
x=[1,2,4,6]  
y=[12,20,3,10]  
plt.bar(x,y)  
plt.show()

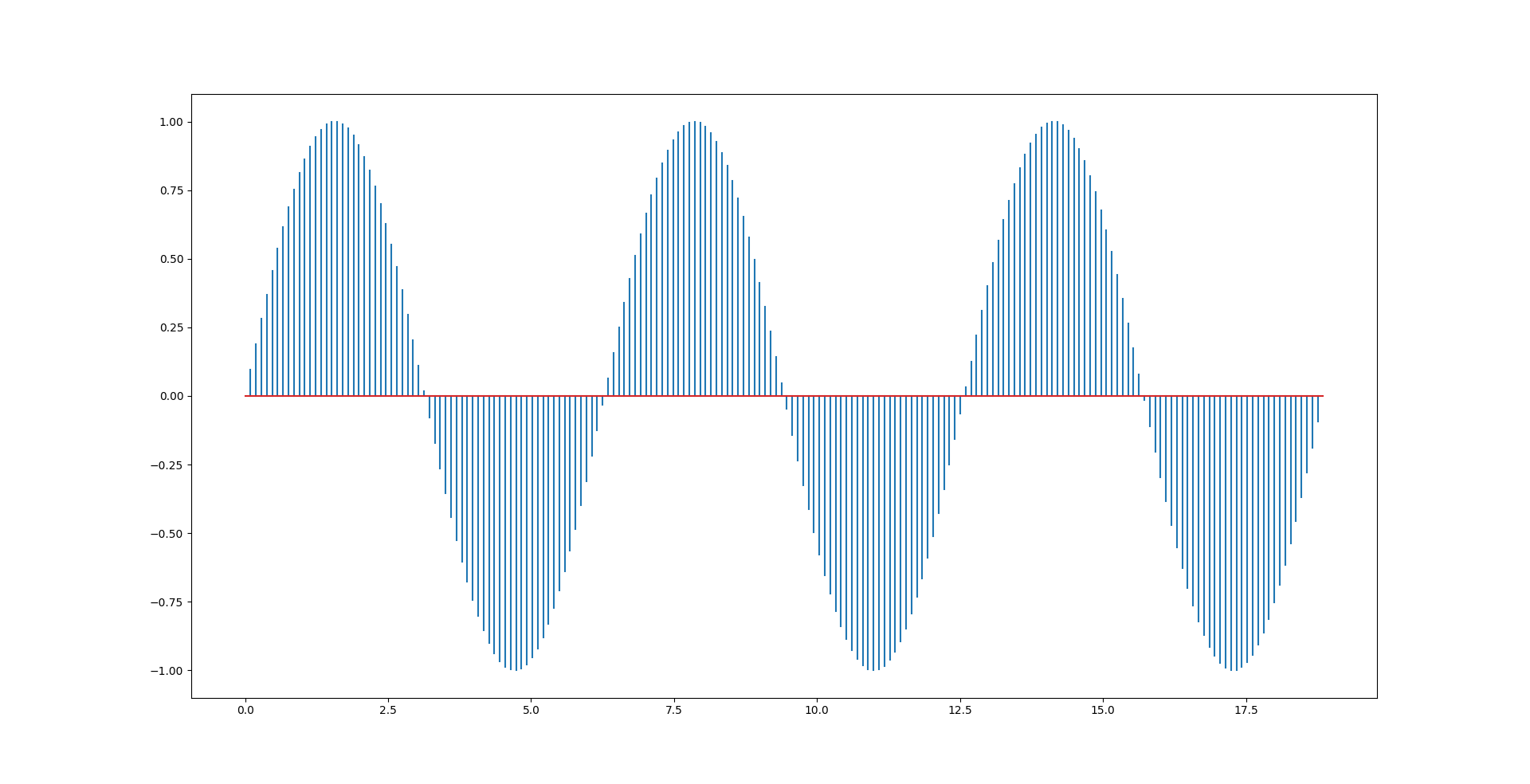
如何顯示中文

**import** matplotlib.pyplot **as** plt  
  
plt.figure()  
plt.title(**'圖片'**)  
plt.ylabel(**'幅度'**)  
plt.figure()  
plt.title(**'圖片'**,fontproperties=**'SimHei'**)  
plt.ylabel(**'幅度'**,fontproperties=**'SimHei'**)  
plt.show()

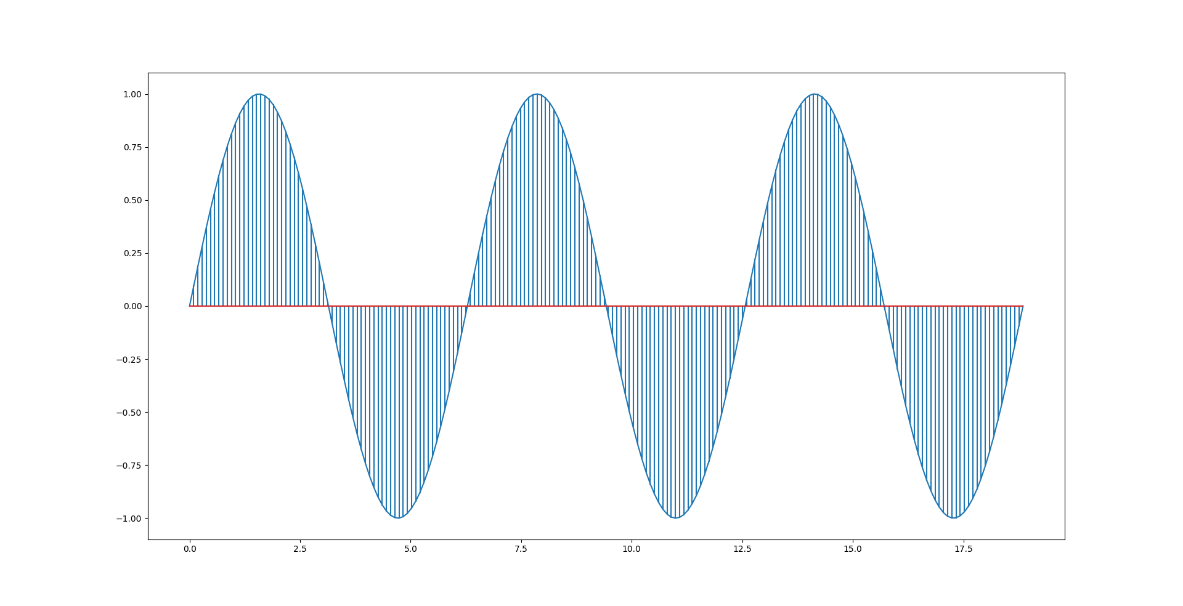


畫stem脈衝圖

**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
  
plt.figure(1)  
x = np.linspace(0, 6\*np.pi, 200)  
y = np.sin(x)  
plt.stem(x, y, markerfmt=**" "**) *# stem圖頂點會用線連起來*plt.figure(2)  
x = np.linspace(0, 6\*np.pi, 200)  
y = np.sin(x)  
plt.stem(x, y, markerfmt=**""**) *# stem圖頂點不會用線連起來*plt.show()



figure(1)



figure(2)

參考

<https://onionys.blogspot.tw/2012/01/python-matplotlib.html?showComment=1481555578029#c1564086401401041632>