python 程式設計

第 五 講

串 列 (二)

星號式子 (一)：用在串列變數

- 在等號左側：合併 (打包) 元素為串列

函式參數設定	運算結果	說明
`( a , *b ) = (1,2,3,4)`	`a = 1 , b = [2,3,4]`	小括號可省略
`*a , b , c = [1,2,3,4]`	`a = [1,2]` `b = 3 , c = 4`	
`*a , b = 1`	錯誤	右式不是常串列，缺逗號
`a , *b = 1 ,`	`a = 1 , b = []`	
`*a , b = 1 ,`	`a = [] , b = 1`	
`*a = 1 , 2 , 3`	錯誤	左式不是常串列，缺逗號
`*a , = 1 , 2 , 3`	`a = [1,2,3]`	
`*a , = ()`	`a = []`	

❖ 在等號左側星號僅能有一個

星號式子 (二)

- 在等號右側：拆解串列成元素

函式參考設定	運算結果	說明
`x = [4,5]` `( a , b , c ) = ( 3 , *x )`	`a = 3` `b = 4 , c = 5`	小括號可省略
`x = [1,2]` `a , b = *x ,`	`a = 1` `b = 2`	末尾逗號不可省略
`x = [1,2]` `a , b , c = *x ,`	錯誤	等號左側多一個變數
`x = [1,2,3] , y = [4]` `a , b , c , d = *x , *y`	`a = 1 , b = 2` `c = 3 , d = 4`	等號右側可有多個星號用來拆解串列

sum 求和函式

- 計算串列的數字和

```
>>> sum( [1,3,7] )
11

>>> sum( [1,3,7] + [4,2] )
17
```

max，min 求最大值與最小值

- 找出串列的極值

```
>>> max( [2,9,7] )
9

>>> min( [4,2,8,9] )
2
```

enumerate 列舉函式 (一)

- **enumerate 列舉函式：將串列的下標與資料打包**
  - 使用 list 將 enumerate 函式的輸出轉型為串列

```
>>> list( enumerate( ["rat","ox","tiger","rabbit"] ) )
[(0, 'rat'), (1, 'ox'), (2, 'tiger'), (3, 'rabbit')]
```

enumerate 列舉函式 (二)

- 印出串列的次序與其值

```
# 型式一:
for n , val in enumerate( [”rat”,”ox”,”tiger”,”rabbit”] ) :
   print( n+1 , ’:’ , val )
# 型式二: p 為兩個元素的常串列
for p in enumerate( ["rat","ox","tiger","rabbit"] ) :
   print( p[0]+1 , ’:’ , p[1] )
```

輸出:

```
1 : rat
2 : ox
3 : tiger
4 : rabbit
```

若使用傳統方式，須另設整數:

```
n = 0
for val in [”rat”,”ox”,”tiger”,”rabbit”] :
   print( n+1 , ’:’ , val )
   n += 1
```

zip 拉鏈函式 (一)

- **zip** 拉鏈函式：在多個一維串列與一個多維串列之間互換
- 將多個一維串列合成一個多維串列
  - 使用 **list** 將 **zip** 函式的輸出轉型為串列

```
>>> a = list( zip( ["cat","dog","bird"] , [20,55,38] ) )
>>> a
[('cat', 20), ('dog', 55), ('bird', 38)]
```

zip 拉鏈函式 (二)

- 英文組句

```
subjects = [ "John" , "Tom" , "Mary" ]
verbs    = [ "likes" , "has" , "plays with" ]
objects  = [ "cat" , "dog" , "parrot" ]
for s , v , o in zip( subjects , verbs , objects ) :
    print( s , v , o )
```

輸出:

```
John likes cat
Tom has dog
Mary plays with parrot
```

- 由分量組成座標點

```
>>> xs , ys , zs = [1,2,3] , [4,5,6] , [7,8,9]
>>> pts = list( zip( xs , ys , zs ) )
>>> pts
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

zip 拉鏈函式 (三)

- 將一個多維串列拆解為多個一維串列
  - 拆解多維串列時需使用 * 星號於串列之前
  - 分解動物與數量

```
>>> pets , num = zip( *[ ("birds",35),("dogs", 20),("cats",15) ] )
>>> pets
('birds', 'dogs', 'cats')
>>> num
(35, 20, 15)
```

傳統方式撰寫:

```
pets , num = [] , []
for a , b in [ ("birds",35),("dogs", 20),("cats",15) ] :
    pets.append(a)
    num.append(b)
```

zip 拉鏈函式 (四)

- 分解座標點成各分量

```
>>> pts = [(1, 4, 7), (2, 5, 8), (3, 6, 9)]
>>> xs , ys , zs = zip( *pts )
>>> print( xs , ys , zs )
(1, 2, 3) (4, 5, 6) (7, 8, 9)
```

lambda 函式:快速定義一行無名函式

- 使用 **lambda** 定義函式

```
import math
fn = lambda x : int(math.sqrt(x)*10)  # 定義 fn lambda 函式

for s in range(1,100) : print( fn(s) )
```

以上 **lambda** 函式設定等同

```
def  fn( x ) :
    return int(math.sqrt(x)*10)
```

- 多參數的 **lambda** 函式

```
>>> g = lambda s , t : ( s , t , s+t )
>>> g(1,2)
(1, 2, 3)
```

sort 與 sorted 排序函式 (一)

- `foo.sort()` ：將 `foo` 串列由小排到大，沒有回傳
- `sorted(foo)` ：回傳 `foo` 串列由小排到大的結果，但 `foo` 保持不變

```
>>> a = [3,2,4,1]
>>> a.sort()                    # 沒有回傳
>>> a                           # a 由小排到大
[1, 2, 3, 4]

>>> b = [3,2,4,1]
>>> sorted(b)                   # 回傳由小排到大的結果
[1, 2, 3, 4]
>>> b                           # b 保持不變
[3, 2, 4, 1]
```

❖ 如果串列為 tuple，僅能使用 sorted 來排序

sort 與 sorted 排序函式 (二)

- 逆向排序
  - `foo.sort(reverse=True)` ：將 `foo` 串列由大排到小，沒有回傳
  - `sorted(foo,reverse=True)`：回傳 `foo` 串列由大排到小的結果，但 `foo` 保持不變

```
>>> a = [3,2,4,1]
>>> a.sort(reverse=True)          # 沒有回傳
>>> a                             # a 由小排到大
[4, 3, 2, 1]
```

sort 與 sorted 排序函式 (三)

- 自訂排序方式
  - **foo.sort(key=fn)** : **foo** 串列依 **fn** 函式設定排列，沒有回傳
  - **sorted(foo,key=fn)** : 回傳 **foo** 串列依 **fn** 函式設定排列，但 **foo** 保持不變

```
# 比較個位數，由小排到大
>>> sorted( [12,76,3,25] , key=lambda x : x%10 )
[12, 3, 25, 76]

# 比較字串長度，由大排到小
>>> sorted( ["cat","ox","tiger"] , key=len , reverse=True )
['tiger', 'cat', 'ox']

# 比較字串長度，由大排到小
>>> sorted( ["cat","ox","tiger"] , key=lambda x : -len(x) )
['tiger', 'cat', 'ox']
```

sort 與 sorted 排序函式 (四)

- 雙重串列排序：串列元素為串列

```
>>> animal_no = [ ["pig",18], ["fish",20],
                  ["dog",20], ["cat",11] ]

# 根據動物名稱排序
>>> sorted( animal_no , key = lambda p : p[0] )
[['cat', 11], ['dog', 20], ['fish', 20], ['pig', 18]]

# 先根據動物數量(由大到小)，再依據名稱
>>> sorted( animal_no , key = lambda p : ( -p[1] , p[0] ) )
[['dog', 20], ['fish', 20], ['pig', 18], ['cat', 11]]
```

❖ 更複雜的比較方式須設計函式，待之後說明

map 映射函數 (一)

- **map(fn,foo) ：一一取出 foo 串列的元素送到 fn 函式處理**
  - 使用 **list** 將 **map** 函式的輸出轉型為串列

```
>>> a = list( map( len , [ "cat", "tiger" , "lion" ] ) )
>>> a
[3, 5, 4]
```

❖ **此種方式等同使用以下的 list comprehension**

```
>>> a = [ len(x) for x in [ "cat", "tiger" , "lion" ] ]
```

  - 逐一印出字串前後字母，中間字元以星號取代

```
>>> b = map( lambda x : x[0]+"*"*(len(x)-2)+x[-1] , [ "cat", "duck" ] )
>>> list(b)
['c*t', 'd**k']
```

map 映射函數 (二)

- 將整數字串轉為整數數字

```
>>> foo = list( map( int , [ ”100” , ”200” , ”300” ] ) )
>>> foo
[100, 200, 300]
```

- 計算整數字串的平方和

```
>>> sum( map( lambda x : int(x)**2 , [ ”1” , ”2” , ”3” ] ) )
14
```

map 映射函數 (三)

- 三角塔

```
>>> for x in map( lambda x : " "*(5-x)+"*"*(2*x+1),range(5))
                    : print(x)
        *
       ***
      *****
     *******
    *********
```

- 印出字串與數字合併: a5 b7 c2

```
for z in map( lambda x,y : x+str(y) , ["a","b","c"] , [5,7,2] ) :
    print( z )
```

- 計算兩整數串列內積

```
>>> sum( map( lambda x , y : x*y , [1,2,3] , [4,5,2] ) )
20
```

filter 過濾函式

- **filter(fn,foo) ：過濾出 foo 串列滿足 fn 函式條件的元素**
  - 使用 **list** 將 **filter** 函式的輸出轉型為串列

```
>>> list( filter( lambda x : x > 0 , [1,3,-2,4,9] ) )
[1, 3, 4, 9]
```

❖ **此種方式等同使用以下的 list comprehension**

```
>>> [ x for x in [1,3,-2,4,9] if x > 0 ]
```

  - 計算正數和

```
>>> sum( filter( lambda x : x > 0 , [1,3,-2,4,9] ) )
17
```

  - 找出 [0,50] 間數字的個位數與十位數之和大於 10 的數

```
>>> list( filter( lambda x : x//10+x%10 > 10 , range(51) ) )
[29, 38, 39, 47, 48, 49]
```

pylab 畫圖套件

- 極為簡單方便的畫圖套件
- **pylab** 是 **matplotlib** 物件導向畫圖函式庫的程序介面
- 大部份的繪圖指令用法近似 **MATLAB** 相關功能函式語法
- 若要對所產生的圖形有更多的控制權限，建議使用 **matplotlib**

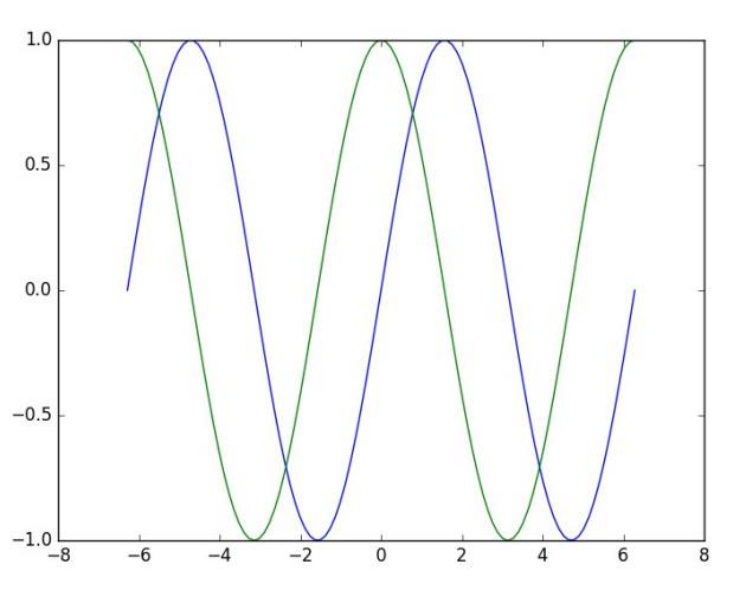
pylab 折線圖 (一)

- 折線圖步驟
  - 對每一條折線:
    - 將折線各點座標分量分別存為 **xs, ys** 串列
    - **plot(xs,ys)** : 畫此連接線
  - 顯示或儲存圖形:
    - **show()** :
      顯示圖形於螢幕
    - **savefig('foo.jpg')** :
      圖形存入 **foo.jpg** 檔案，也可儲存為 **foo.png**、**foo.pdf**、**foo.eps** 等圖形格式

pylab 折線圖 (二)

- 操作範例:

```
import pylab

a , b = -10*pylab.pi , 10*pylab.pi
n = 101
dx = (b-a)/(n-1)

# 分別計算 sin(x) 與 cos(x) 座標
xs = [ a + i * dx for i in range(n) ]
ys1 = [ pylab.sin(x) for x in xs ]
ys2 = [ pylab.cos(x) for x in xs ]

# 分別畫 sin(x) 與 cos(x) 折線圖
pylab.plot(xs,ys1)
pylab.plot(xs,ys2)

# 顯示圖形
pylab.show()
```



顏色設定：多種方式

- X11/CSS4 顏色名稱：

  white、red、green、blue、cyan、magenta、yellow、black

- '#rrggbb' 字串：

  rr、gg、bb ∈ [0, 255] 整數，以 16 進位表示。

  '#ff0000'：紅色、'#e0ffe0'：淺綠

- (r,g,b) 常串列：

  r：紅、g：綠、b：藍，三數皆 ∈ [0, 1] 浮點數

- (r,g,b,a) 常串列：

  r、g、b、a ∈ [0, 1] 浮點數，a 代表不透明度，0 為完全透明，1 為完全不透明

- 設定折線線條顏色

```
# 線條為綠色
pylab.plot(xs,ys1,color='#00ff00')
# 線條為淡藍色
pylab.plot(xs,ys2,color=(0.9,0.9,1))
```

❖ 許多函式都可透過設定 color 變換顏色，這也包含文字

pylab 圖形設定 (一)

- **pylab 圖形設定: 使用 plot 後, 可用以下函式改變圖形設定**
  - **title("tstr") :** 設定圖形標頭字串
  - **xlabel("xstr"), ylabel("ystr") :**
    分別設定 X 軸與 Y 軸的標示字串
  - **xlim(x_1,x_2), ylim(y_1,y_2) :**
    分別設定圖形顯示區域為 $[x_1,x_2]\times[y_1,y_2]$
  - **axis(v=(x_1,x_2,y_1,y_2)) :**
    一次性設定圖形顯示區域為 $[x_1,x_2]\times[y_1,y_2]$。此函式也可使用 axis('off') 來隱藏兩軸線。若要同時隱藏軸線且設定顯示區域, 則使用 axis('off',v=(x_1,x_2,y_1,y_2))
  - **xticks(locs,labels), yticks(locs,labels) :**
    設定 X 軸或 Y 軸某位置的對應標籤, 例如:
    locs=[1,2,3], labels=["A","B","C"], 代表在 1、2、3 等位置要使用 "A"、"B"、"C" 替代, 此多用於直條圖 。若要將隱藏刻度線, 則可用 xticks([],[]) 或 yticks([],[])。

pylab 圖形設定 (二)

- **grid()** : 產生背景格網
- **text(x,y,'tstr',fontsize=16)** :
  在圖形的 **(x,y)** 座標標示 **tstr** 字串，字元大小為 **16** 點
- **arrow(x,y,dx,dy)**:
  由 **(x,y)** 到 **(x+dx,y+dy)** 兩座標點間畫一箭頭線
- **legend()** :
  配合 **plot(... , label='sstr')** 在圖形內產生 **sstr** 圖例
- **figure(figsize=(10,12),facecolor='white',edgecolor='k',lw=10)**:
  設定圖形大小 **10×12** 吋，背景顏色為白色，邊框顏色為黑色，
  邊框寬度(或用 **linewidth**) 為 **10pt**

❖ **圖形內文字字元大小可透過 fontsize=xx 設定為 xx 點大小**

使用 LATEX 語法文字於 pylab 圖形

- 圖形標頭，X 軸與 Y 軸或其它字串可以使用 LATEX 語法
- pylab.rc('text',usetex=True) 讓字串可使用 LATEX 語法
- LATEX 字串要在字串前加上 r，代表原生字串(參考第 ?? 頁)

例如：以下將圖形標頭設為 $f(x) = \frac{sin(x)}{x}$，

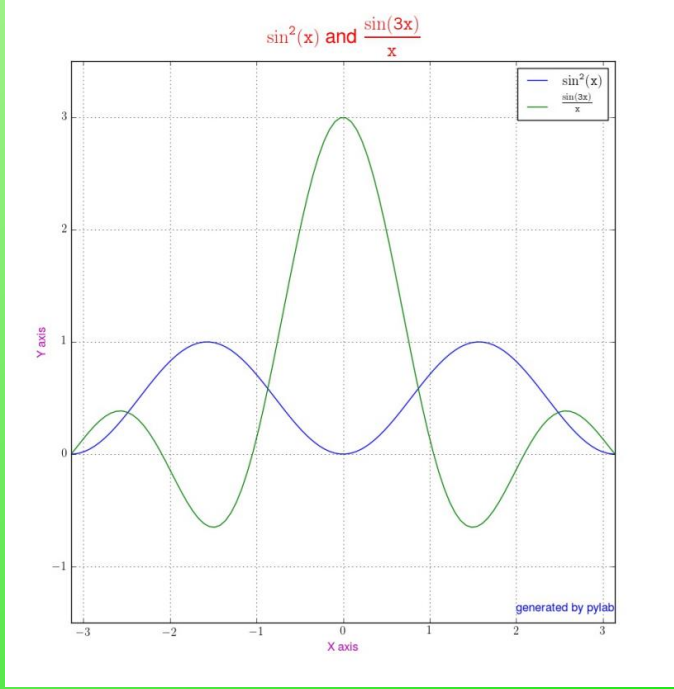
```
pylab.title( r'$$\mathtt{f(x) = \frac{sin(x)}{x}}$$' )
```

- 因中文字型預設於繪圖系統中，圖形字串仍無法顯示中文字

畫圖 (一)

- 畫圖: $\sin^2(x)$ 與 $\frac{\sin(3x)}{x}$

```
import pylab

# 讓圖形背景為白色
pylab.figure(facecolor='white')

# 讓圖形可使用 LaTeX 語法文字
pylab.rc('text',usetex=True)

# 函式 x 在 [-2 pi,2 pi] 共 200 個點
a , b , n = -2*pylab.pi , 2*pylab.pi , 200
dx = (b-a)/(n-1)

# 第一條折線圖 sin(x)^2 : xs 與 ys1 座標
# 第二條折線圖 sin(3x)/x: xs 與 ys2 座標
xs = [ a + i * dx for i in range(n) ]
ys1 = [ pylab.sin(x)**2 for x in xs ]
ys2 = [ pylab.sin(3*x)/x for x in xs ]
```



畫圖 (二)

```
# 畫第一條折線圖，並設定線條圖例
pylab.plot(xs,ys1,label=r"$\mathtt{\sin^2(x)}$")

# 畫第二條折線圖，並設定線條圖例
pylab.plot(xs,ys2,label=r"$\mathtt{\frac{\sin(3x)}{x}}$")

# 設定圖形標頭文字
pylab.title(r"$$\mathtt{\sin^2(x) \mbox{and} \frac{\sin(3x)}{x}}$$",
            fontsize=20,color='r')

# 設定 x 軸 y 軸的文字，顏色為 magenta(洋紅色)
pylab.xlabel("X axis",color='m')
pylab.ylabel("Y axis",color='m')

# 設定圖形 x 與 y 顯示範圍 [-pi,pi]x[-1.5,3.5]
pylab.axis((-pylab.pi,pylab.pi,-1.5,3.5))

# 產生背景線
pylab.grid()
```

畫圖 (三)

```
# 根據各個 plot 的 label 來產生線條圖例
pylab.legend()

# 在 (2,-1.4) 座標以藍色 12 pt 文字標示: generated by pylab
pylab.text(2,-1.4,'generated by pylab',color='blue',fontsize=12)

# 顯示圖形
pylab.show()
```

plot 畫線函式的線條設定(一)

- **plot(xs , ys , style , ...) :**
  在相鄰座標點間畫連接線段， **n** 個座標點間會有 **n-1** 條線段
  xs , ys 為代表座標點的 **x** 與 **y** 座標串列，**style** 為代表圖示性質的字串。

- **style** 指定線的幾個性質:
  - 顏色(**color 或 c**):

代碼	顏色
r	紅色
m	洋紅色

代碼	顏色
g	綠色
y	黃色

代碼	顏色
b	藍色
k	黑色

代碼	顏色
c	青色
w	白色

plot 畫線函式的線條設定 (二)

- 點的顯示形狀(**marker**):

代碼	符號	代碼	符號	代碼	符號	代碼	符號	代碼	符號
.	.	o	o	+	+	x	×	*	*
D	◊	v	▽	^	△	s	■	|	|

❖ **若為空字串，代表不畫點**

- 連接線的樣示(**linestyle or ls**):

代碼	符號	代碼	符號	代碼	符號	代碼	符號
-	實線	--	虛線	:	點線	-.	點虛線

❖ **若為空字串，代表不畫線**

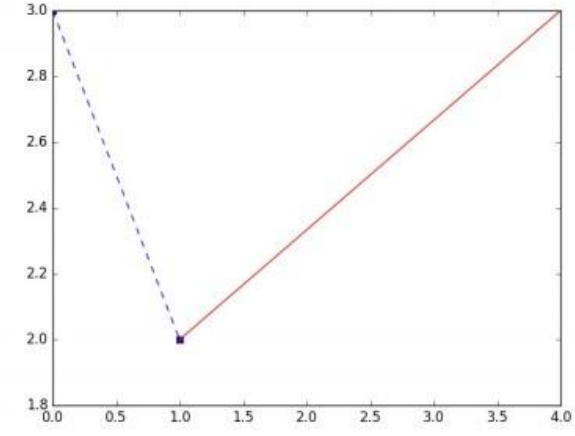
plot 畫線函式的線條設定(三)

- 範例
  - 畫線段連接 [0,3], [1,2] 兩點，使用虛線、藍色、方塊，線寬 5pt

```
pylab.plot( [0,1] , [3,2] , "--bs" , lw=5 )
# 同上
pylab.plot( [0,1] , [3,2] , ls='--' , c='b' , marker='s' , lw=5 )
```

  - 先畫線連接 [0,3], [1,2] 兩點，使用虛線、藍色、方塊，再畫線連接 [1,2],[4,3] 兩點，使用實線、紅色、加號

```
pylab.plot( [0,1] , [3,2] , "--bs" , [1,4] , [2,3] , "-r+" )
```



linspace (一)：快速取得平分點

- **linspace(a,b,n) :**
  在 **[a,b]** 間產生包含兩端點的 **n** 個等距點，回傳 **array** 物件
- **linspace(a,b,n,endpoint=False) :**
  同上，但不包含 **b** 點，，**endpoint** 預設為 **True**
- **linspace(a,b,n,retstep=True) :**
  也回傳間距，**retstep** 預設為 **False**
  retstep 可與 **endpoint** 一起使用

linspace (二)

```
# 在 [0,5] 間產生包含頭尾兩點合起來共 6 個點存於 xs
>>> xs = pylab.linspace(0,5,6)
>>> xs
array([0., 1., 2., 3., 4., 5.])

# endpoint = False 不包含末端點值
>>> xs = pylab.linspace(0,5,5,endpoint=False)
>>> xs
array([0., 1., 2., 3., 4.])

# retstep = True 可回傳兩點間距
>>> xs , dx = pylab.linspace(0,5,6,retstep=True)
>>> dx
1.0
```

❖ array 物件使用方式可參考第 11 章

向量化運算 (一)

向量化運算：**array** 物件可用向量式運算簡化執行步驟

- **array** 物件的每個元素可做相同運算

```
>>> xs = pylab.linspace(0,3,4)

# ys 為 xs 每個元素加上 10
>>> ys = xs + 10
>>> ys
array([10., 11., 12., 13.])

# zs 儲存 xs 每個元素的平方根
>>> zs = pylab.sqrt(xs)
>>> zs
array([0.        , 1.        , 1.41421356, 1.73205081])
```

向量化運算 (二)

- 串列無法進行向量化運算

```
>>> xs = [1,2,3]
>>> xs = xs / 3                  # 錯誤

>>> ys = [4,5]
>>> ys = ys * 2                  # 串列複製成兩倍，但非向量化運算
>>> ys
[4, 5, 4, 5]
```

- 串列轉為 **array** 物件

```
>>> xs = pylab.array([1,2,3])
>>> xs = xs / 3
>>> xs
array([0.33333333, 0.66666667, 1.          ])

>>> ys = pylab.array([1,2,3])/3
>>> ys
array([0.33333333, 0.66666667, 1.          ])
```

向量化運算 (三)

- 快速畫圖法

```
# 在 [-10,10] 區間畫出 sin(x) 函數圖形
xs = pylab.linspace(-10,10,100)
pylab.plot( xs , pylab.sin(xs) )
pylab.show()
```

❖ 更多 linspace 與向量化運算的範例可參考第 11 章

其它圖形顯示 (一)

- **bar(xs,hs,width=0.8)**：直條圖

  xs 與 **hs** 為串列，在 xs_i **x** 座標畫出高度為 hs_i 的直條圖，**width** 控制直條寬度（預設為 **0.8**），直條圖通常要使用 **xticks** 設定刻度文字。

- **barh(ys,ws,height=0.8)**：橫條圖

  ys 與 **ws** 為串列，在 ys_i **y** 座標畫出長度為 ws_i 的橫條圖，**height** 控制橫條高度（預設為 **0.8**），橫條圖通常要使用 **yticks** 設定刻度文字。

- **scatter(xs,ys,marker='o',s=10)**：散佈圖

  點的散佈圖，相鄰點沒有線段相連。預設 **marker** 符號為 。，點的大小以 **s** 設定。

- **polar(angs,rs)**：極座標圖形

  angs 為弧度串列，**rs** 為長度串列。若混合 **plot**、**scatter**、**fill** 等函式都使用極座標參數。

其它圖形顯示(二)

- **fill(xs,ys,color='k'): 多邊形塗滿**

  xs 與 ys 為多邊形座標，頭尾不需重複，可用 color 參數設定顏色

- **fill_between(xs,ys1,ys2,where=ys1>ys2,color='r'): 在兩個 y 值之間塗色**

  在兩個 y 值之間塗色，若沒有 ys2，代表 ys2 值皆為零，可用 where 塗顏色的條件，例如:

```
fill_between(xs,ys,where=ys>=0,color='b')   # ys 大於等於 0 時用藍色塗滿
fill_between(xs,ys,where=ys<=0,color='r')   # ys 小於等於 0 時用紅色塗滿
```

- **loglog(xs,ys,basex=10,basey=10): log-log 座標**

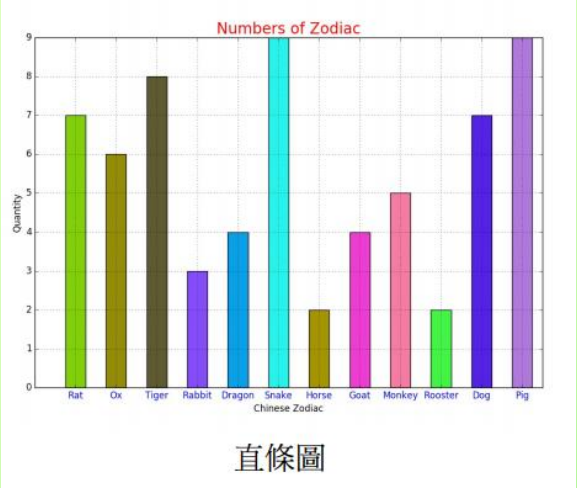
  x 與 y 軸都使用 log scale 座標，x 軸與 y 軸的 log 底數預設皆為 10

- **semilogy(xs,ys,basey=10): y 軸為 log 座標**

  僅有 y 軸為 log scale 座標，預設 log 底數為 10

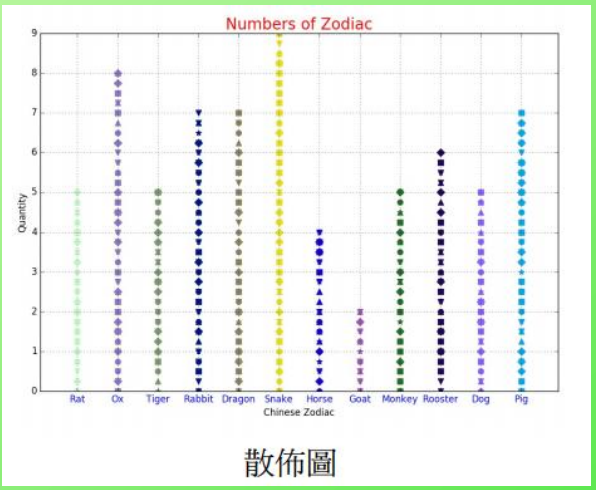
❖ **更多不同形式圖形顯示方式，參考 matplotlib 網站**

其它圖形顯示(三)



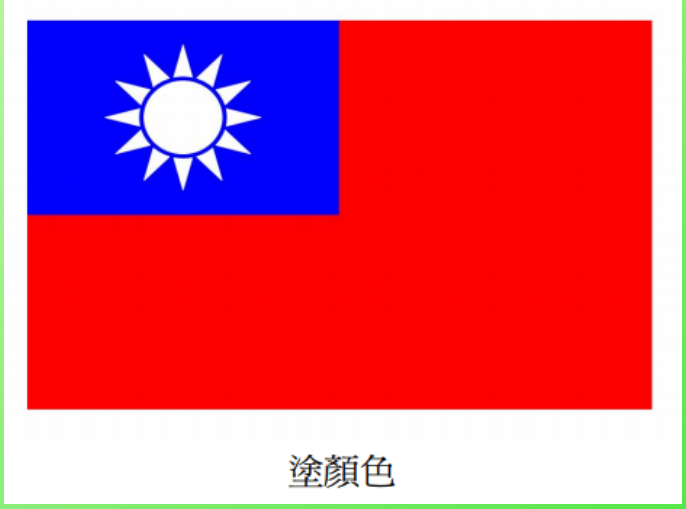
直條圖

橫條圖

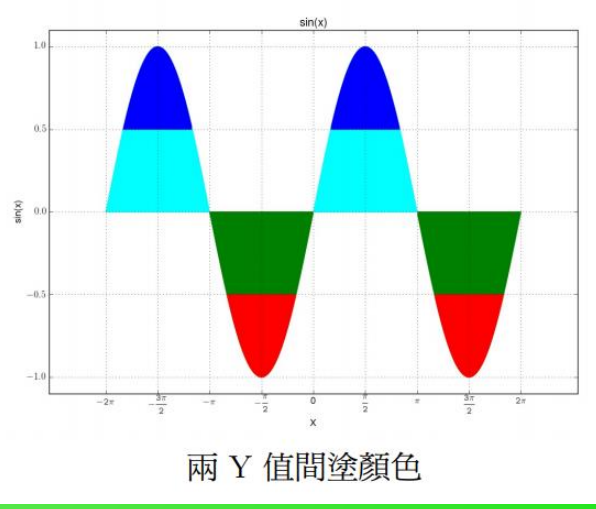


散佈圖

極座標圖



塗顏色



兩 Y 值間塗顏色

數值積分法 (一)

若積分區域 [a,b] 在 x 軸上被切割為 n 等份，每等份長為 h，由左到右 x 的座標點為 $x_0, x_1, x_2, \cdots, x_n$ ， $x_0 = a$, $x_n = b$ ，則有以下積分公式：

- 矩形積分公式：

$$\int_a^b f(x)\,dx \approx h \sum_{i=0}^{n-1} f(x_i)$$

- 梯形積分公式：

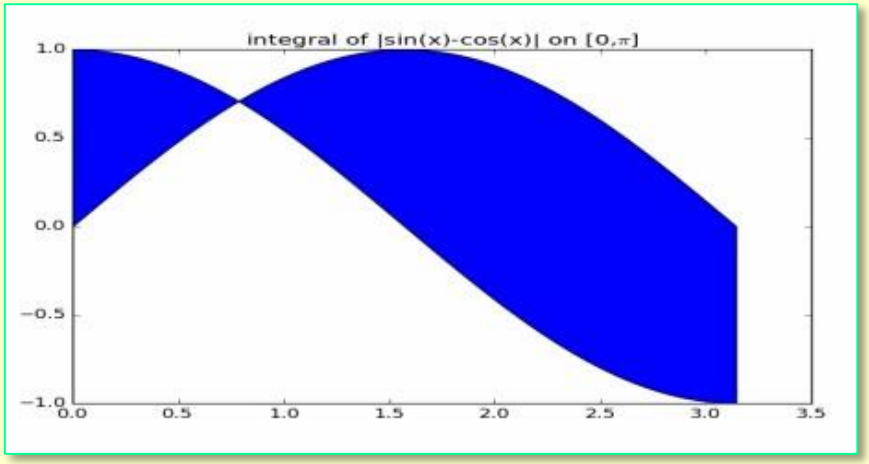
$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(x_0) + 2\sum_{i=0}^{n-1} f(x_i) + f(x_n)\right)$$

- Simpson 積分公式(n 為偶數)：

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f(x_0) + 4(f(x_1) + f(x_3) + \cdots + f(x_{n-1})) + 2(f(x_2) + f(x_4) + \cdots + f(x_{n-2})) + f(x_n)\right)$$

數值積分法 (二)

```
from math import *

n , a , b = 100 , 0 , pi
h = (b-a)/n

# lsum : lower , usum : upper , tsum : trapezoid
lsum , usum , tsum = 0 , 0 , 0

y1 = abs( sin(a) - cos(a) )

# 傳統迴圈
for i in range(1,n+1) :
    x = a + i*h
    y2 = abs( sin(x) - cos(x) )
    if y1 < y2 :
       lsum += y1
       usum += y2
    else :
       lsum += y2
       usum += y1

    tsum += y1 + y2

    y1 = y2

lsum *= h
usum *= h
```

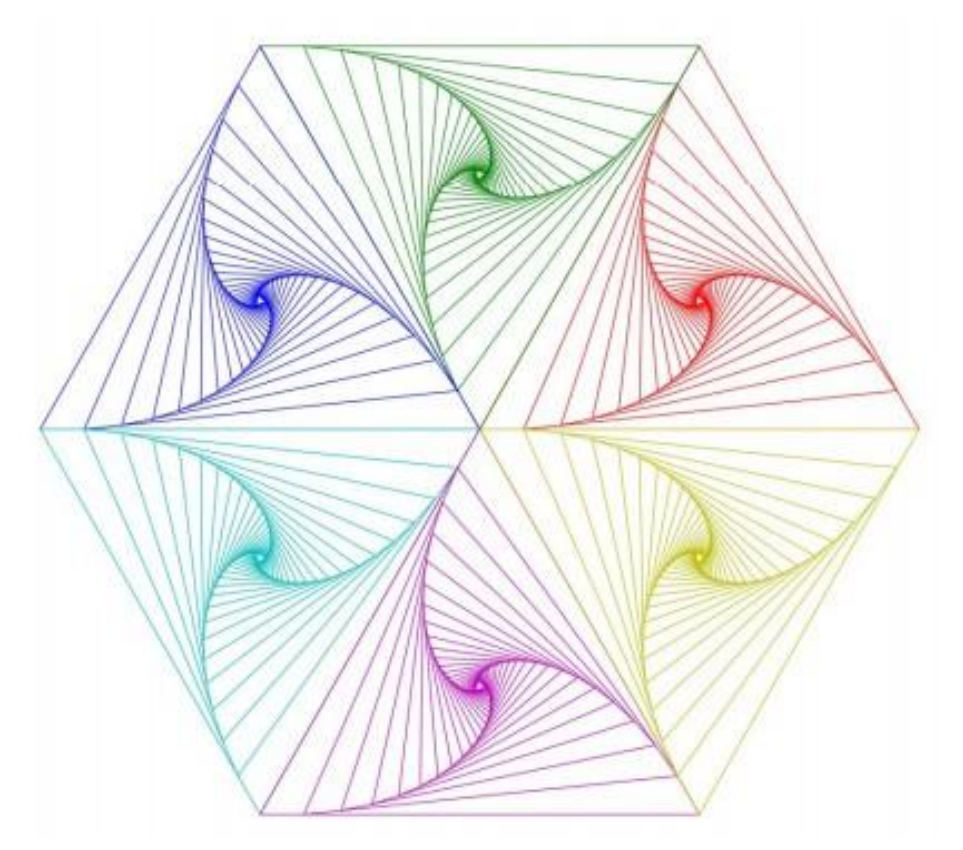


$$\int_0^{\pi} |\sin(x) - \cos(x)| \, dx = 2\sqrt{2}$$

數值積分法 (三)

```
tsum *= h/2
isum = 2*sqrt(2)

print( "數學積分   :" , round(isum,8) )
print( "上矩形積分 :" , round(usum,8) , " 誤差:" , round(abs(isum-usum),8) )
print( "下矩形積分 :" , round(lsum,8) , " 誤差:" , round(abs(isum-lsum),8) )
print( "梯形積分法 :" , round(tsum,8) , " 誤差:" , round(abs(isum-tsum),10) )
print()

# 使用串列
xs = [ a + i * h for i in range(n+1) ]
fns = [ abs(sin(x)-cos(x)) for x in xs ]
#fns = [ sin(x) for x in xs ]

# 矩形法: 1,1,1,1,...,1,1
isum1 = h * sum( fns[:-1] )

# 梯形法: 1,2,2,2,...,2,2,1
isum2 = h * sum( [ fns[0] , 2*sum(fns[1:-1]) , fns[-1] ] ) / 2

# simpson 1/3 rule: 1,4,2,4,2,...,2,4,1
isum3 = h * sum( [ fns[0] , 4*sum(fns[1:-1:2]) , 2*sum(fns[2:-1:2]) ,
                   fns[-1] ] ) / 3

print( "矩形積分法  :" , round(isum1,8) , " 誤差:" , round(abs(isum-isum1),10) )
print( "梯形積分法  :" , round(isum2,8) , " 誤差:" , round(abs(isum-isum2),10) )
print( "Simpson積分:" , round(isum3,8) , " 誤差:" , round(abs(isum-isum3),12) )
```

旋轉的三角形 (一)



旋轉的三角形 (二)

畫幾何圖形通常要應用到一些數學公式，以上圖案是由六個內旋三角形所組成的。任兩相鄰的內旋三角形是利用簡單向量公式由大三角形座標計算小三角形座標，假設 a1、 b1、c1 為大三角形的三個頂點座標，那麼小三角形頂點座標 a2、 b2、c2 可由以下公式求得：

$$
\begin{aligned}
a_2 &= r\,a_1 + (1-r)\,b_1 \\
b_2 &= r\,b_1 + (1-r)\,c_1 \qquad r \in [0,1] \\
c_2 &= r\,c_1 + (1-r)\,a_1
\end{aligned}
$$

在此圖案，相鄰大三角形的 r 是在 0.1 與 0.9 之間變換藉以達到順逆時鐘旋轉效果。程式中先將預設的大三角形以 60 度的倍數順時鐘旋轉到適當位置，設定 r 值，再由之算出所有內旋的小三角形。

以下平面點旋轉公式用來計算 (x_1, y_1) 逆時鐘旋轉 θ 角度到 (x_2, y_2)：

$$
\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}
$$

旋轉的三角形 (三)

```
import pylab

xs = [ 0 , 1 , 0.5 ]
ys = [ 0 , 0 , pylab.sqrt(3)/2 ]
colors = "rgbcmyk"

pylab.figure(facecolor='w')

pi = pylab.pi

# 六個旋轉三角形
for k in range(6) :

    cosk = pylab.cos(k*pi/3)
    sink = pylab.sin(k*pi/3)

    # 大三角形
    px = [ *xs[:] , xs[0] ]
    py = [ *ys[:] , ys[0] ]

    # 先旋轉大三角形到相關位置
    for i in range(len(px)) :
        px[i],py[i]  = cosk*px[i] - sink*py[i] ,
                       sink*px[i] + cosk*py[i]

    qx , qy = px[:] , py[:]

    r = 0.9 if k%2==0 else 0.1

    # 再對每個大三角形內部旋轉 25 次
    for i in range(25) :

        pylab.plot(px,py,colors[k])

        for j in range(3) :
            qx[j] = r * px[j] + (1-r) * px[j+1]
            qy[j] = r * py[j] + (1-r) * py[j+1]

        qx[-1] = qx[0]
        qy[-1] = qy[0]

        px , py = qx[:] , qy[:]

pylab.axis('off')
pylab.show()
```

畫出來的數字 (一)

上一章第 ?? 頁利用程式來顯示輸入數字的點陣圖形，呈現的效果如同跑馬燈一樣，所看到的「數字」是由一個個的點(字元)拼湊起來的。在此範例中，程式利用了 **pylab** 的畫線函式將「點」畫成「線」來顯示數字圖案。

為了讓畫出來的數字有連續不斷線的效果，程式中先檢查了每個點上下左右四方是否有相鄰點，若有則將相鄰資料儲存起來。之後在畫個別「點」時，就可根據此點的相鄰資料，將點看成線條，計算線條的兩個端點位置加以畫線，如此點與點之間就連在一起達到了不斷線的效果。例如：下圖的左側點矩圖形是在方框內畫點，點與點之間有空隙，右側則在方框內畫線，同時調整線條寬度，造成相連的效果。

畫出來的數字 (二)

```
import pylab

bmap = ( (15,9,9,9,15), (2,2,2,2,2), (15,1,15,8,15), (15,1,15,1,15),
         (9,9,15,1,1), (15,8,15,1,15), (15,8,15,9,15), (15,9,1,1,1),
         (15,9,15,9,15), (15,9,15,1,15) )

# 每個矩矩陣的橫列數與直行數
R , C = len(bmap[0]) , 4

# 設定每個點矩陣位置的四方鄰居是否存在
nmap = []
for n in range(10) :

    foo = [ [None]*C for i in range(R) ]

    for r in range(R) :
        for c in range(C) :

            s = []
            if bmap[n][r] & ( 1 << c ) :
               if c > 0 :
                   # 檢查右側是否有點
                   if bmap[n][r] & ( 1 << (c-1) ) : s += [1]

               if r > 0 :
                   # 檢查上方是否有點
                   if bmap[n][r-1] & ( 1 << c ) : s += [2]

               if c < C-1 :
                   # 檢查左側是否有點
                   if bmap[n][r] & ( 1 << (c+1) ) : s += [3]
```

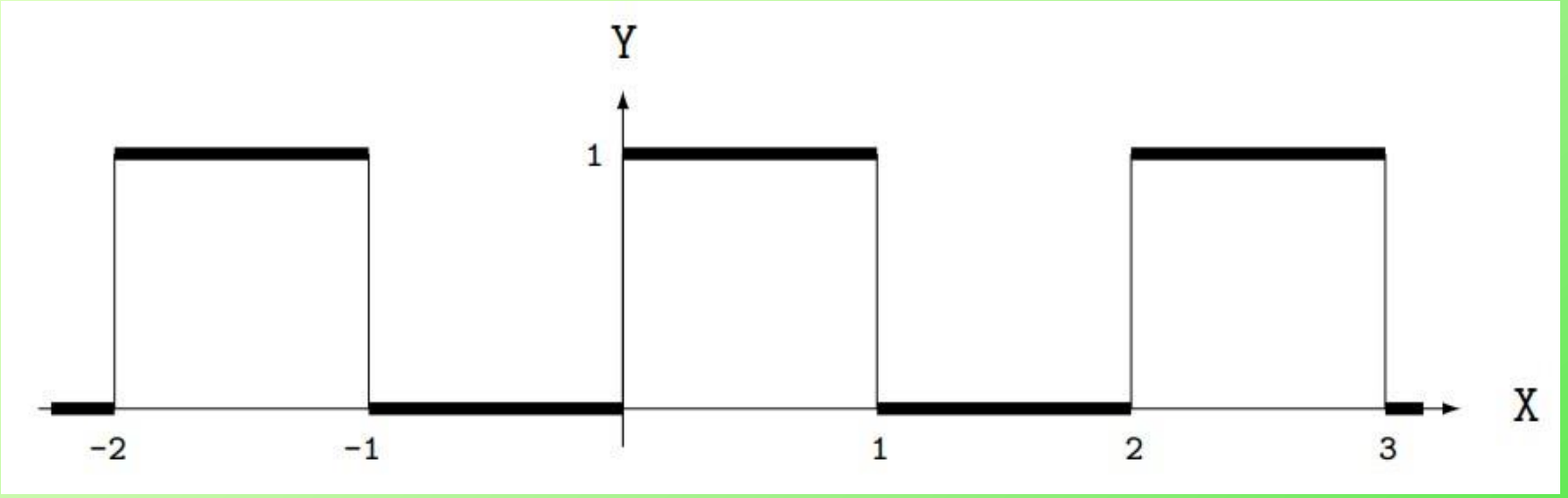
畫出來的數字 (三)

```
                if r < R-1 :
                    # 檢查下方是否有點
                    if bmap[n][r+1] & ( 1 << c ) : s += [4]

            foo[r][c] = sorted(s)

    nmap += [ foo ]

# 設定每個數字的 ds 高, dt 寬
ds , dt = 1 , 0.8

while True :

    # 輸入數字
    num = input("> ")

    pylab.figure(facecolor='w')

    # 每列
    for r in range(R) :

        # 每個數字
        for k in range(len(num)) :

            # 每個位數
            n = int(num[k])

            # 每行
            for c in range(C) :
```

畫出來的數字 (四)

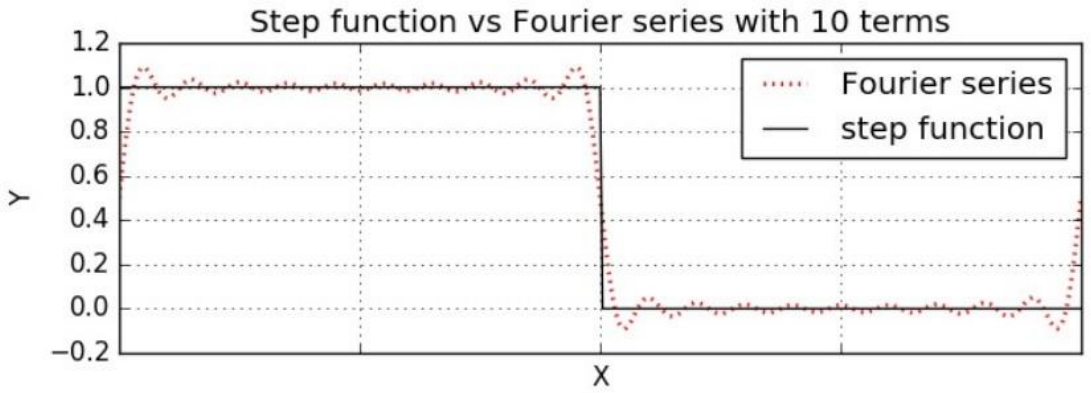
```
            # 中心點位置
            x = ( k*C + k + (C-1-c) )*dt
            y = (R-r)*ds
            # 由每個點的中心位置畫到與鄰居點交界位置
            for w in nmap[n][r][c] :
                if w == 1 :
                    x2 , y2 = x+dt/2 , y
                elif w == 2 :
                    x2 , y2 = x , y+ds/2
                elif w == 3 :
                    x2 , y2 = x-dt/2 , y
                elif w == 4 :
                    x2 , y2 = x , y-ds/2

                pylab.plot( [x,x2] , [y,y2] , 'r' , lw=20)

# 不要顯示兩個軸線
pylab.axis('off')

pylab.show()
```

階梯函式 (一)：使用 Fourier Series 逼近



一些上下振盪的階梯函式經常需要使用 **sin θ** 或 **cos θ** 的線性組合來逼近，其中最常使用的組合方式稱為傅立葉級數（**Fourier Series**）。例如上方的階梯函數可透過下方的傅立葉級數來逼近。本範例即是計算傅立葉級數與階梯函數的差距並以圖形顯示出來。

階梯函式 (二):



$$f_n(x) = \frac{1}{2} + \frac{2}{\pi}\left(\sin(\pi x) + \frac{\sin(3\pi x)}{3} + \frac{\sin(5\pi x)}{5} + \cdots\right) = \frac{1}{2} + \frac{2}{\pi}\sum_{j=1}^{\infty}\frac{\sin((2j-1)\pi x)}{2j-1}$$

階梯函式 (三):

為了便於對比函式，特別使用 **subplot()** 函數將數個圖形同時顯示於圖紙上，各個圖形可 自行設定標頭、軸線標籤、格線、圖例等等，以下為簡單的介紹。

- **subplot(nrows=1,ncols=1,sharex=False,sharey=False,...):**
  同時顯示 **nrows×ncols** 個圖形於圖紙上，**nrows** 與 **ncols** 預設值都為 **1** 。
  若 **sharex(sharey)** 為 **True**，則所有圖形使用同樣的 **X(Y)** 刻度。函式回傳兩個物件，第一個為 **Figure** 物件，第二個為圖形物件。後者根據圖形個數設定可為一個元素（少用）、或一維串列或二維串列。

階梯函式 (四):

- 上下兩個圖形，共用 **x** 軸，背景為白色:

```
# gs 為一維串列, gs[0] 上圖 , gs[1] 下圖
fig , gs = pylab.subplot(2,sharex=True,facecolor='w')
```

- 顯示 **2×3** 個圖形於圖紙上，共用 **Y** 軸:

```
# gs 為二維串列, gs[0][0](左上圖) , gs[0][1] , ... ,gs[1][2](右下圖)
fig , gs = pylab.subplot(2,3,sharey=True)
```

➢ 設定各個顯示圖形資料:

透過 **subplot** 回傳的圖形物件可設定各個圖形的標頭、軸線標籤、格線、刻度文字、圖例等等。

階梯函式 (五):

```
# 顯示上下兩圖, gs 為圖形串列: gs[0] 為上圖, gs[1] 為下圖
fig , gs = pylab.subplot(2,sharex=True)

# 上圖
gs[0].bar(xs,ys)                              # 畫直條圖
gs[0].grid()                                  # 顯示格線
gs[0].set_title("title")                      # 設定圖形標頭

# 設定上圖的 Y 刻度文字 (因共用 X 軸, X 刻度文字由下圖控制)
# Y 軸在 [10,20,30] 等刻度分別顯示 A B C
pylab.setp( gs[0] , yticks=[10,20,30] , yticklabels=["A","B","C"] )

# 下圖
gs[1].plot(xs,ys1,label="fn1")                # 畫折線圖並設定圖例文字
gs[1].plot(xs,ys2,label="fn2")                # 畫折線圖並設定圖例文字
gs[1].set_xlabel("X")                         # 設定 x 軸文字
gs[1].set_ylabel("Y")                         # 設定 y 軸文字
gs[1].legend()                                # 顯示圖例

# 設定下圖的 X 與 Y 軸刻度文字,
# X 軸在 [0,1,2,3] 等刻度分別顯示 Cow Pig Chicken Fish
# Y 軸在 [10,20,30] 等刻度分別顯示 A B C
pylab.setp( gs[1] ,
            xticks=[0,1,2,3] , xticklabels=["Cow","Pig","Chicken","Fish"] ,
            yticks=[10,20,30] , yticklabels=["A","B","C"] )

pylab.show()
```

❖ 有關各函式更詳細的用法, 請參考 matplotlib 網站

階梯函式 (六):

```
import pylab
from math import *

f , gs = pylab.subplots(2, sharex=True)

m = int( input("no of terms : ") )

a , b , np = 0 , 2 , 400
dx = (b-a)/np

xs = [ a + i * dx for i in range(np+1) ]
ys = [ None ] * (np+1)
ss = [ 1 if x<=1 else 0 for x in xs ]

# 計算 Fourier Series 的估算值
for i , x in enumerate(xs) :
    y , pix = 0 , pi * x
    for j in range(m) : y += sin((2*j+1)*pix)/(2*j+1)
    ys[i] = 0.5 + 2*y/pi
```

階梯函式 (七):

```
# 上圖
gs[0].plot(xs,ys)
gs[0].plot([a,b],[1,1],':r',[a,b],[0,0],':k')
gs[0].set_title("Step function : First " + str(m) +
                " terms of Fourier series")

# 下圖
gs[1].fill_between(xs,ys,ss)
gs[1].plot([a,b],[1,1],':r',[a,b],[0,0],':k')
gs[1].set_title("Error of Fourier Series")

# 顯示圖案
pylab.show()
```

練習題 (一)

1. 撰寫程式，模擬橋牌規則發牌，每人的牌面由大到小排列，以下為四人的牌面輸出樣式：

```
SJ S10 S5 S4   S3 S2 H9 H5   H3 DA D5 D2   CQ
S8 S7   HJ H10 H7 H6 D8 D4   CJ C9 C5 C4   C2
SQ S9   HA HQ  H2 DK DQ D10  D7 CA CK C10  C6
SA SK   S6 HK  H8 H4 DJ D9   D6 D3 C8 C7   C3
```

以上的 S、H、D、C 分別代表 Spade、Heart、Diamond、Club。

練習題 (二)

2. 四個不同多面柱體骰子分別可顯示 1 到 3 點，1 到 4 點，1 到 5 點，與 1 到 6 點等不同點數，撰寫程式模擬隨意甩這四個骰子，驗算四個骰子的點數呈現連續數字的機率為 0.075 $(=\frac{27}{360})$ 。
(提示：可使用排序，驗證是否連續數字)

練習題 (三)

3. 撰寫程式驗證隨意取五張撲克牌，得到順子的機率為 0.0039246。請留意，要排除同花順的情況。

練習題 (四)

4. 請根據以下男童、女童圖形，自行設計相關的點矩陣串列。在程式中加入相關條件印出四肢與身體等符號，同時使用以下串列定義男女童資料，以亂數控制兒童數量在 [3,10] 之間與各別的男童、女童。執行時，每按 rtn 鍵後即重新設定 kids 串列，然後隨之印出兒童圖案。

```
kids = [ randint(0,1) for i in range(randint(3,10)) ]
```

輸出為:

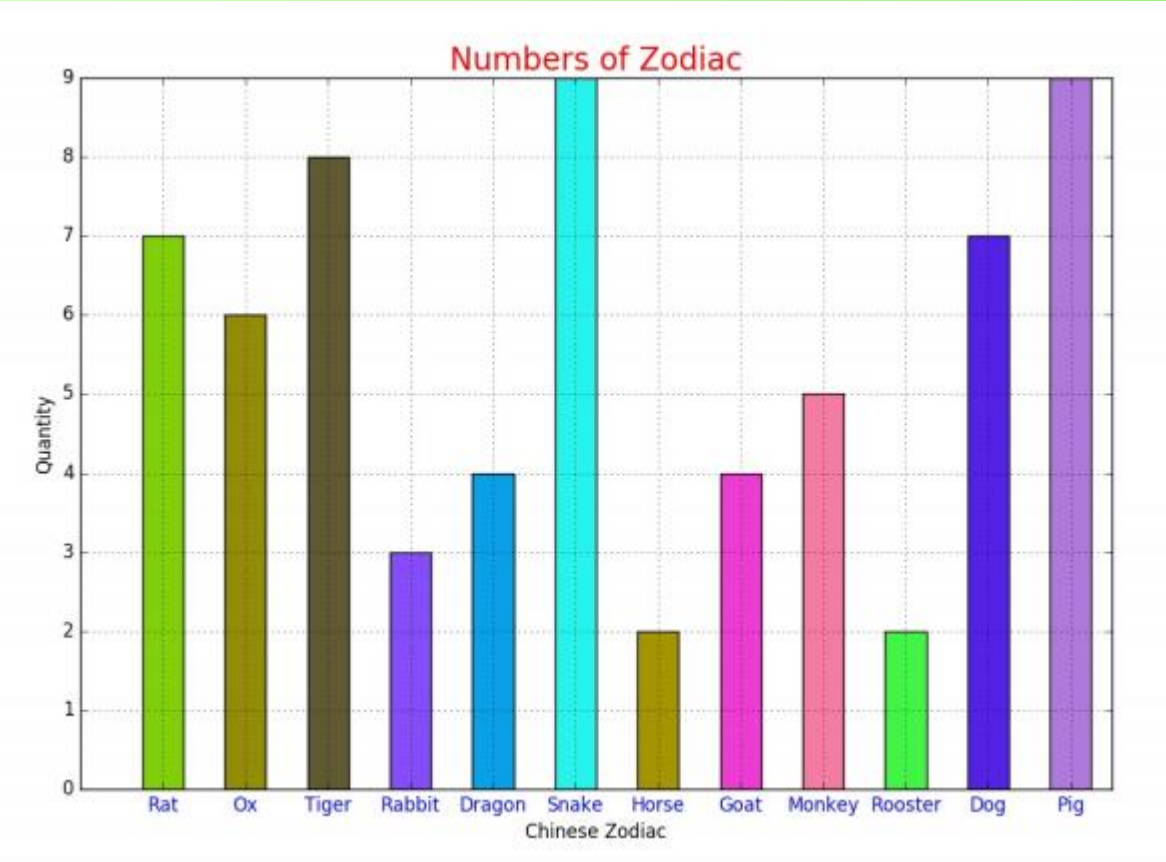
```
>
   ***     ***     ***     ***
  *****   *****   *****   *****
/ *** \   ***   / *** \   ***
    I       I       I       I
   111     222     333     444
  /111\   /222\   /333\   /444\
/ 111 \/ 222 \/ 333 \/ 444 \
   ***     222     ***     444
  *****    | |    *****    | |
   | |     | |     | |     | |
============================
```

練習題 (五)

```
>
  ***     ***     ***     ***     ***     ***     ***     ***
 *****   *****   *****   *****   *****   *****   *****   *****
  ***   / *** \ / *** \   ***   / *** \   ***   / *** \   ***
   I       I       I       I       I       I       I       I
  111     222     333     444     555     666     777     888
 /111\   /222\   /333\   /444\   /555\   /666\   /777\   /888\
/ 111 \ / 222 \ / 333 \ / 444 \ / 555 \ / 666 \ / 777 \ / 888 \
  111     ***     ***     444     ***     666     ***     888
  | |    *****   *****    | |    *****    | |    *****    | |
  | |     | |     | |     | |     | |     | |     | |     | |
==========================================================
```

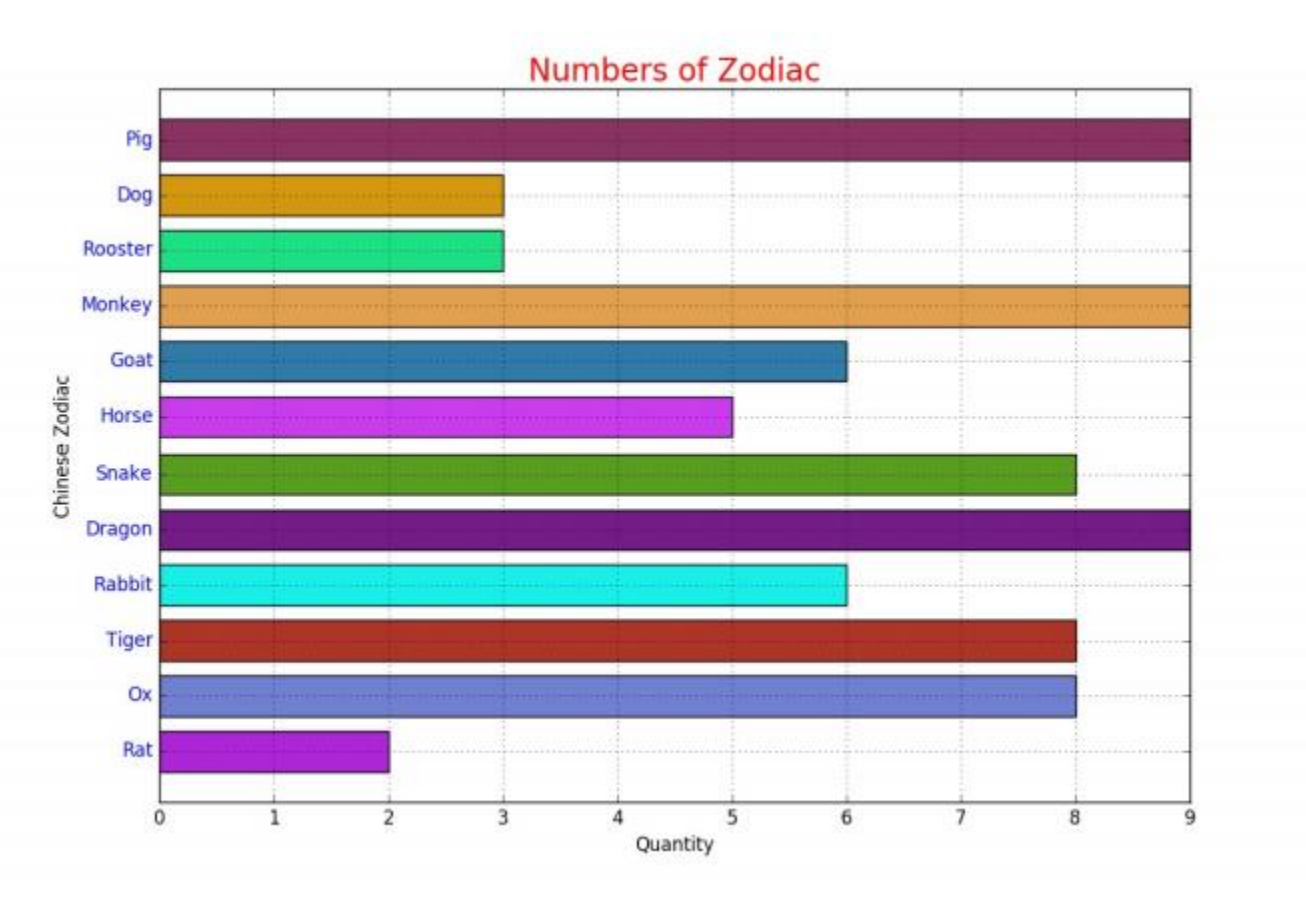
練習題 (六)

5. 利用亂數設定 12 生肖的數量，使用直條圖呈現以下圖形：



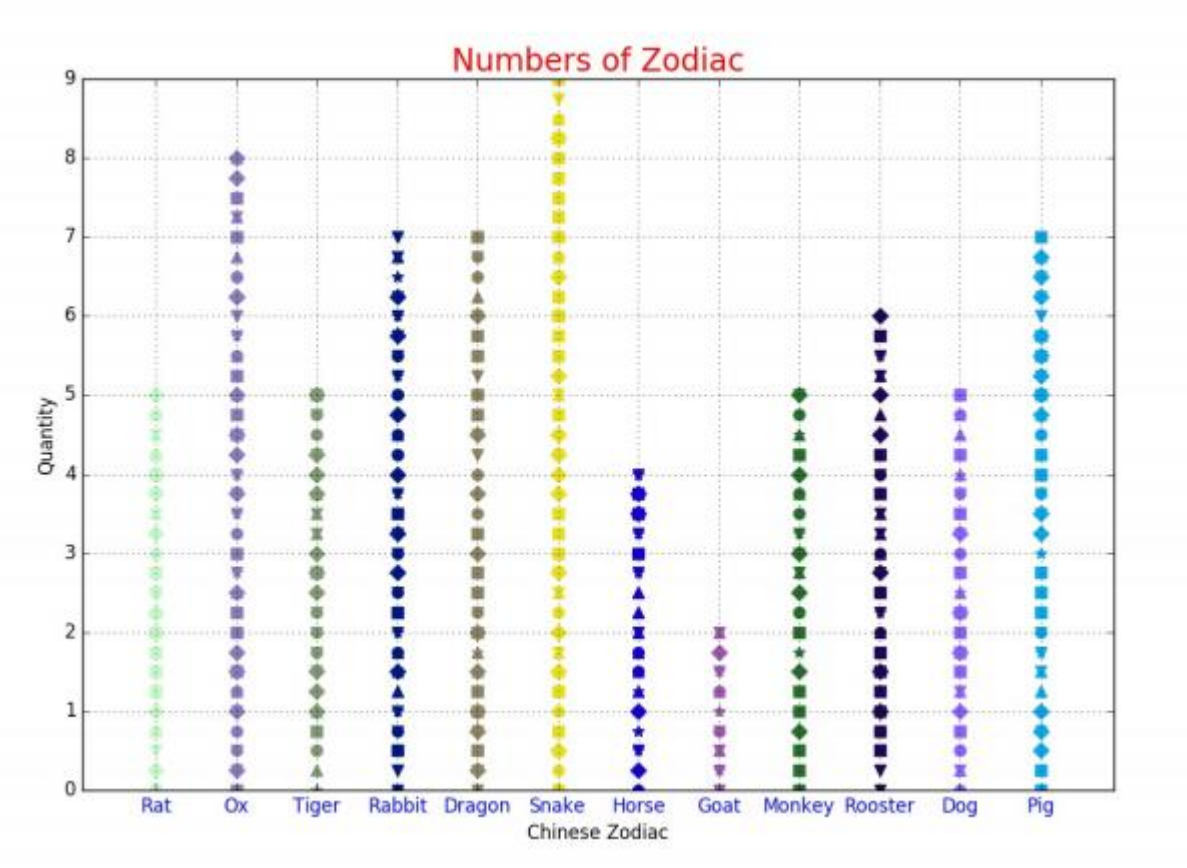
練習題 (七)

6. 利用亂數設定 12 生肖的數量，使用橫條圖呈現以下圖形：



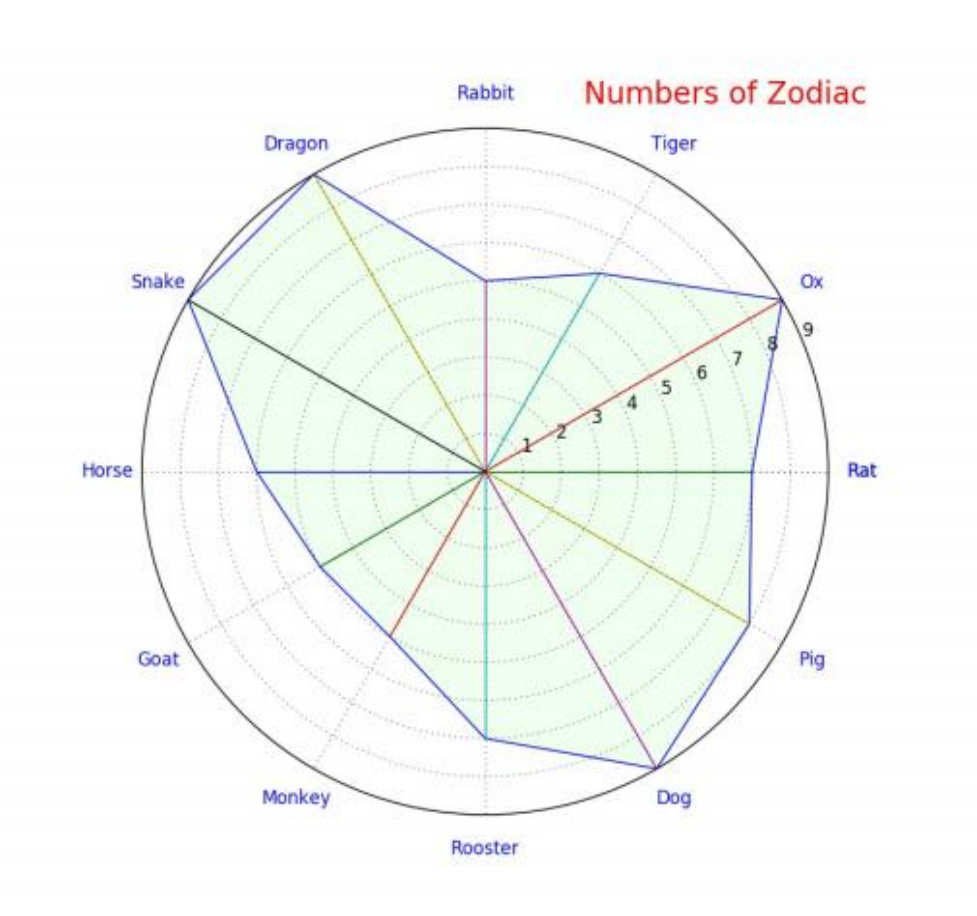
練習題 (八)

7. 利用亂數設定 12 生肖的數量，利用散佈圖呈現以下圖形：



練習題 (九)

8. 利用亂數設定 12 生肖的數量，使用極座標呈現以下圖形：

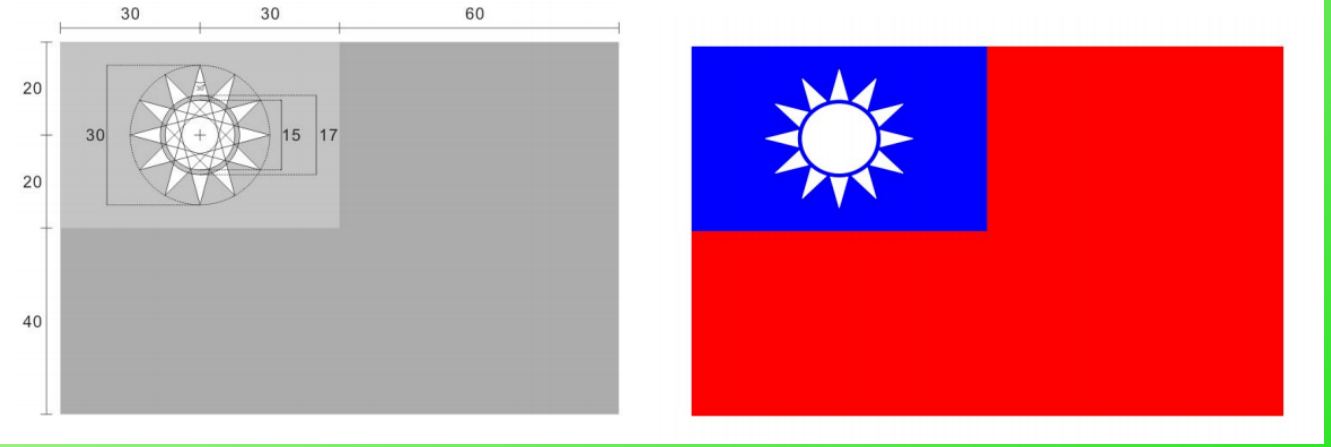


練習題 (十)

9. 以下左圖為國旗圖形尺寸，撰寫程式畫出國旗，程式將會用到點旋轉公式：

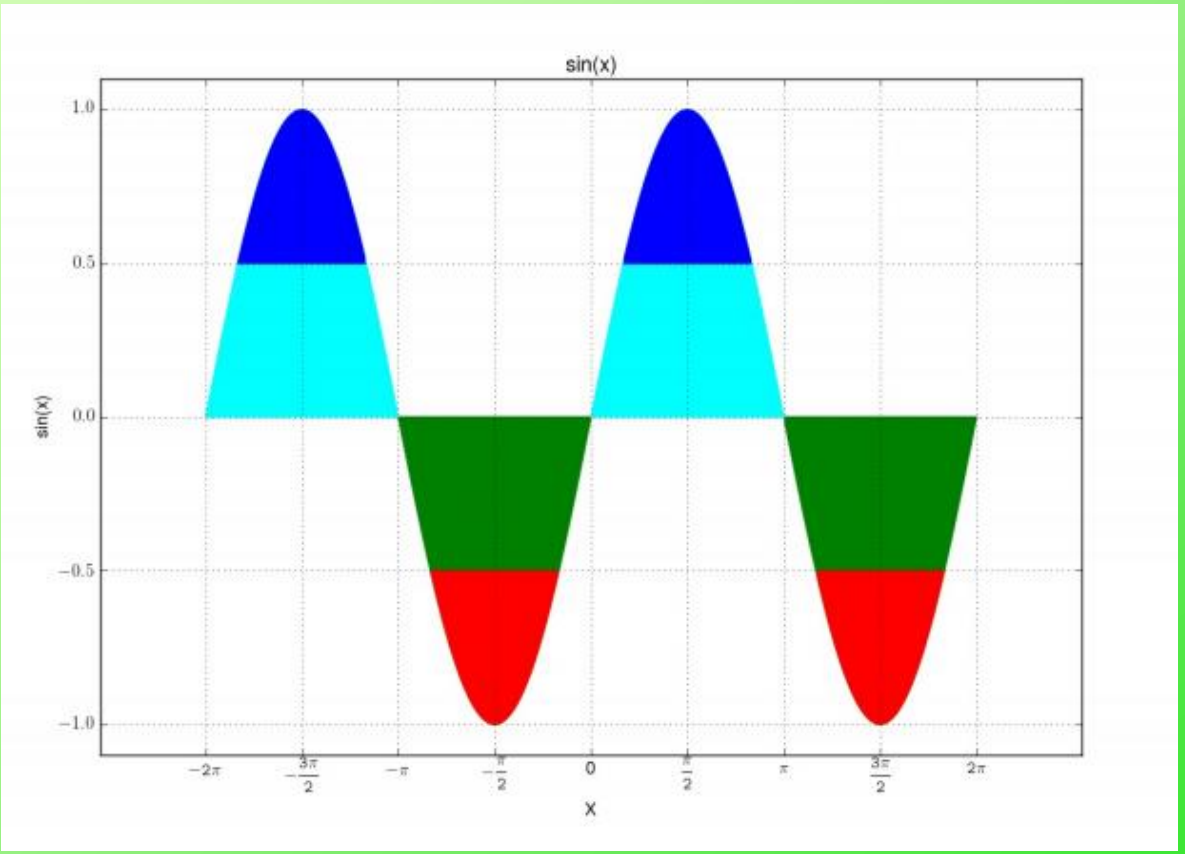
$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

為方便起見，在計算上可使用以下資料：
首先將座標軸的原點定義在國旗左上角的圓心上，國旗的第一道光芒的尖角在正 **x** 軸線上，設定此尖角座標為 **(15,0)**，經過計算後此光芒內側弧形的兩個端點座標分別為**(8.3088,1.7929)**，與 **(8.3088,-1.7929)**，其弧角為 **24.3538** 度。



練習題 (十一)

10. 利用 pylab.fill_between 函式在 x ∈ [-2π,2π] 間畫出 sin(x) 函數，圖形依照 y 值區分為四個區域，用不同顏色表示。

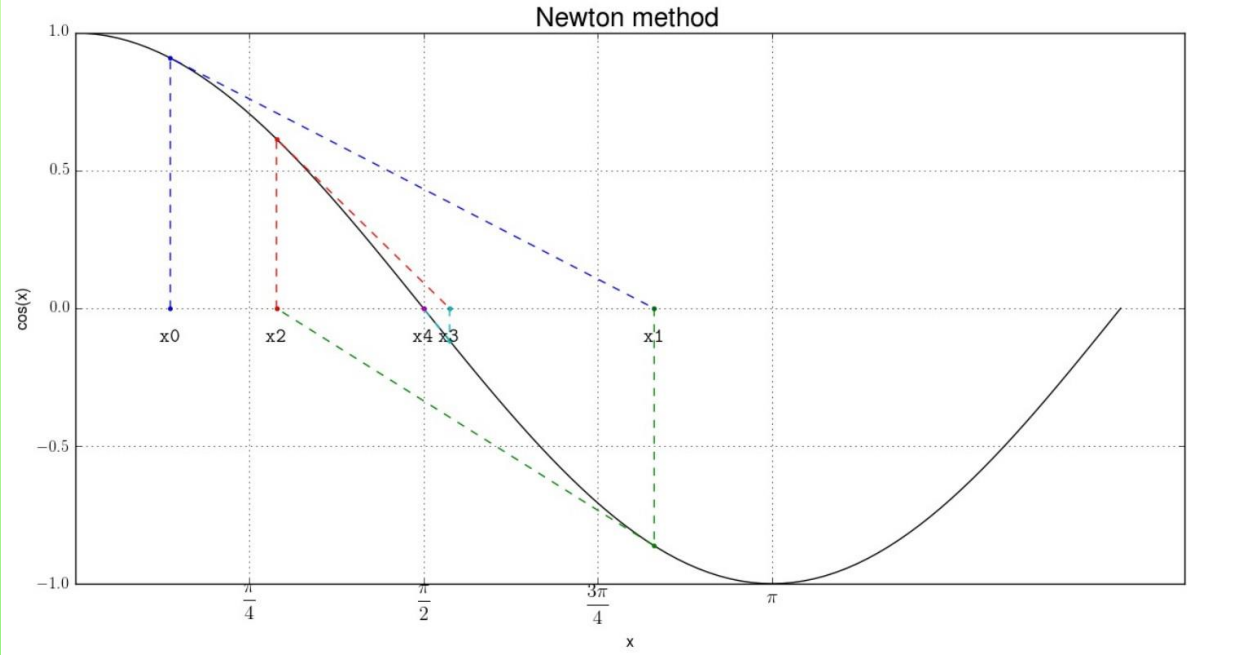


練習題 (十二)

11. 撰寫程式實作牛頓迭代求根法，公式如下：

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad i \geq 0$$

求解 f(x) = cos(x) 的近似根，畫出牛頓迭代法迭代五步的過程，假設 x_0 為 0.43。



練習題 (十三)

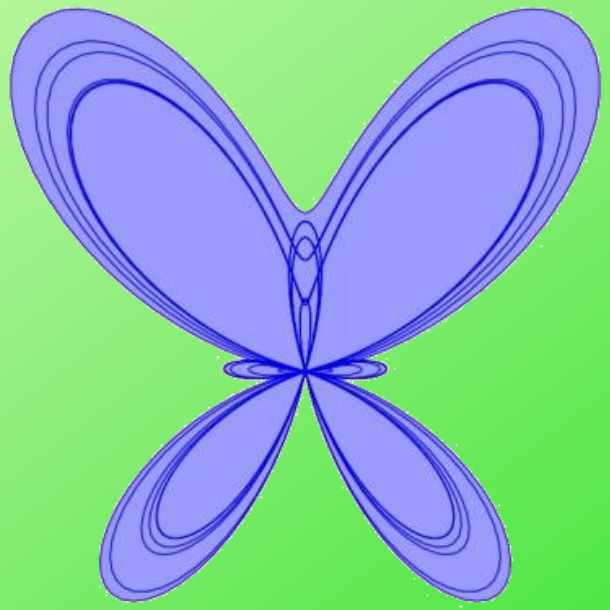
12. 在數學上，蝴蝶曲線被定義為以下參數方程式：

$$x = \sin(t)\left(e^{\cos(t)} - 2\cos(4t) + \sin^5\left(\frac{t}{12}\right)\right)$$

$$y = \cos(t)\left(e^{\cos(t)} - 2\cos(4t) + \sin^5\left(\frac{t}{12}\right)\right)$$

$$t \in [0, 12\pi]$$

撰寫程式，分別使用 **plot** 描線與 **fill** 塗色畫出蝴蝶圖形如下。



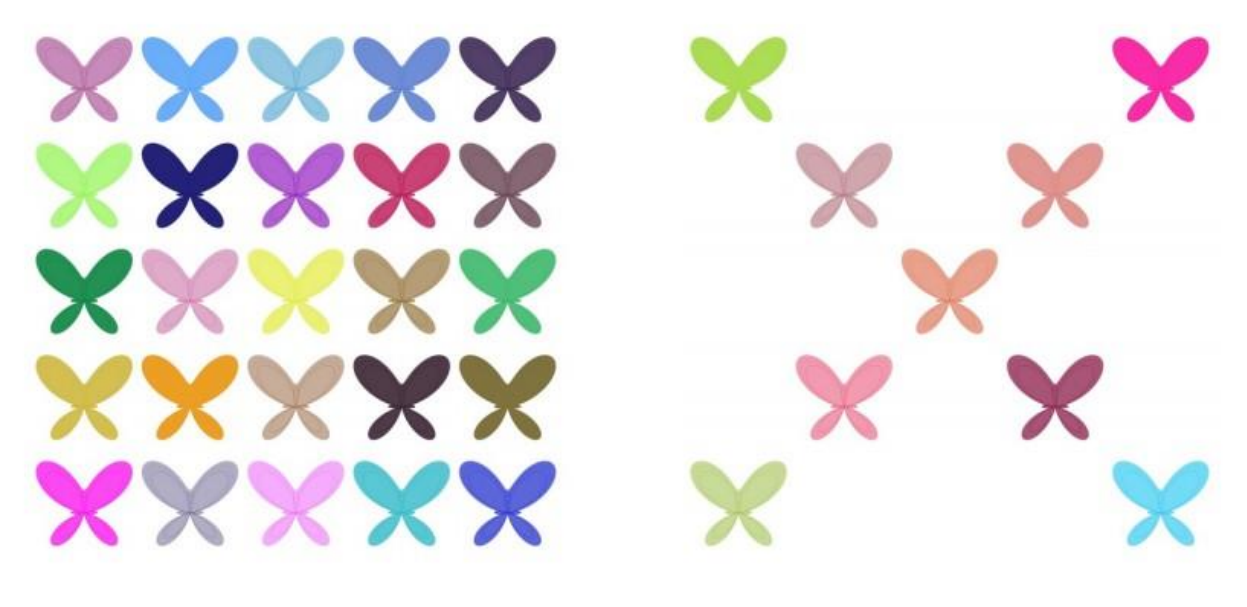
練習題 (十四)

13. 利用以上的蝴蝶曲線參數方程式，並使用 rgba 的顏色代表方式，輸入 n，

(a) 印出 n×n 隻各種不同顏色蝴蝶，

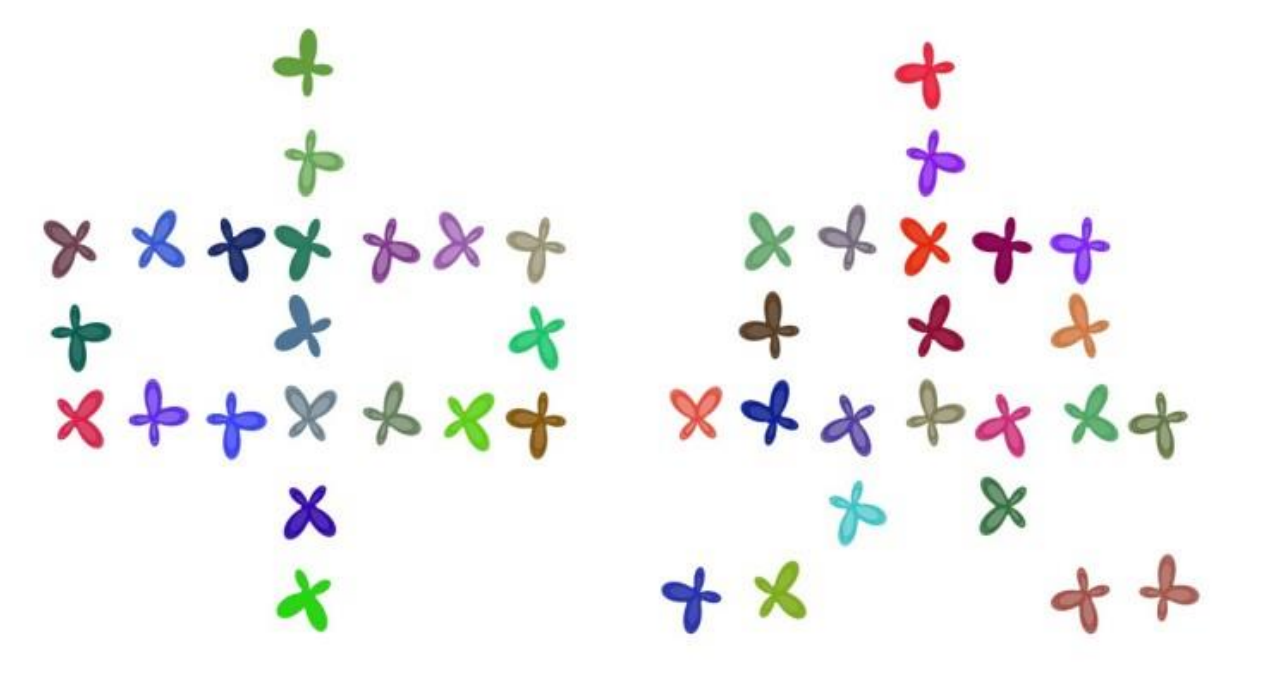
(b) 同上，但只印出在對角線上的蝴蝶。

假設上下相鄰蝴蝶的間距皆設定為 8。



練習題 (十五)

14. 觀察下圖，設定「中央」兩字的點矩陣，撰寫程式利用 **pylab** 畫圖，讓原本應顯示的點都以蝴蝶表示，請自由變更各蝴蝶的旋轉角與縮放比，以下為產生的中央蝴蝶點矩圖形。

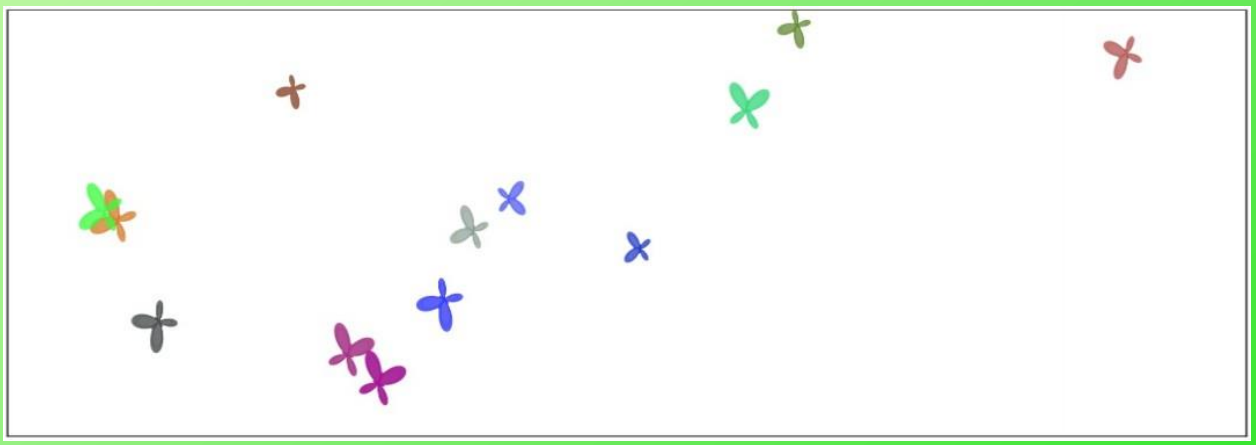


練習題 (十六)

15. 在數學上，要讓 (x_1, y_1) 座標點先經過旋轉 θ 角度、縮放 r 倍、再移動 (dx,dy) 距離，則最後的新座標點 (x_2, y_2) 可由以下公式求得：

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = r \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

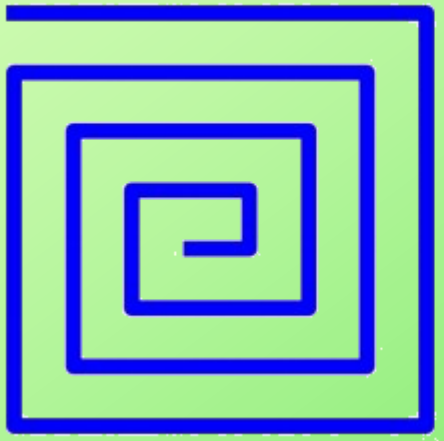
請撰寫程式，使用亂數套件隨意設定 θ、r、[dx,dy] 隨意產生 10 到 15 隻蝴蝶。以下為產生的蝴蝶圖形。



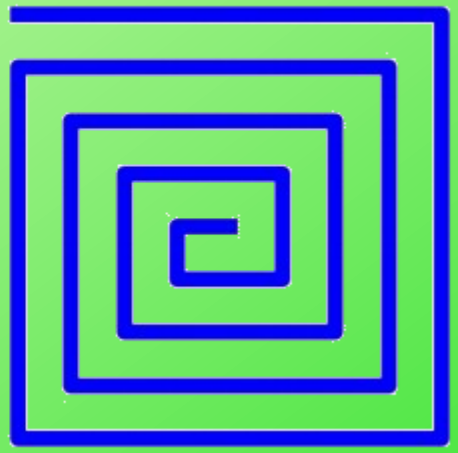
練習題 (十七)

16. 輸入螺線寬度 n ，使用 pylab 畫出以下的方形旋轉螺線。

> 7

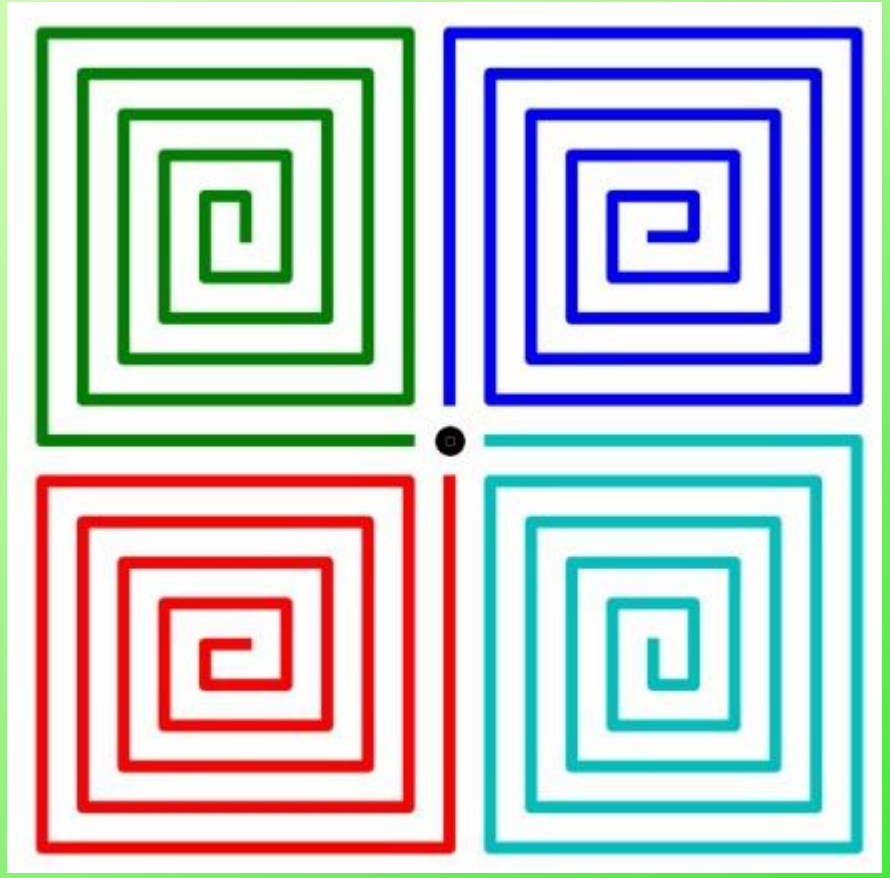


> 8



練習題 (十八)

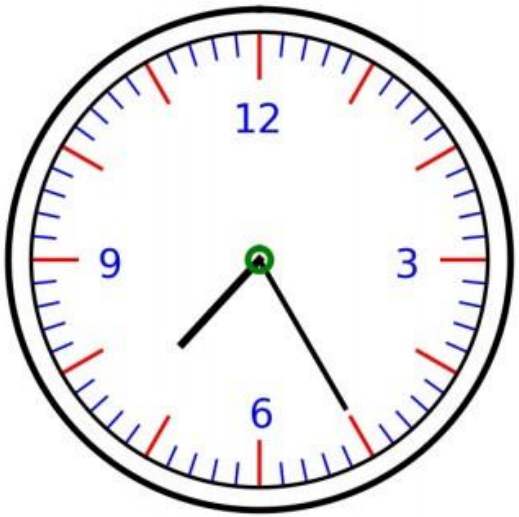
17. 同上題，利用旋轉公式與 pylab 畫圖成以下螺線圖案。



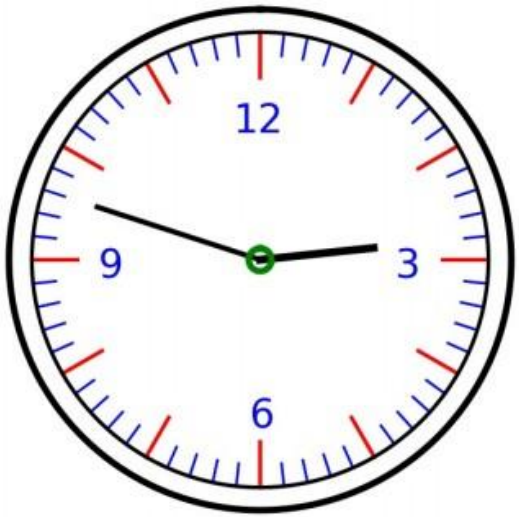
練習題 (十九)

18. 撰寫程式，輸入時分，用 **pylab** 畫出對應的時鐘圖形如下。

> 7 , 25



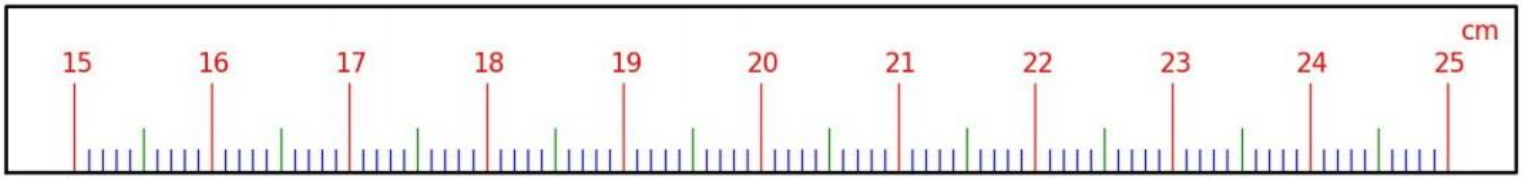
> 2 , 48



練習題 (二十)

19. 撰寫程式，輸入刻度尺的起始刻度，畫出 10 公分長的刻度尺如下。

```
> 15
```



練習題 (二十一)

20. 撰寫程式，輸入三位數，畫出以下的度量儀度刻度表。