**第33章**

**動畫與遊戲**

**33-1：繪圖功能**

**33-2：尺度控制畫布背景顏色**

**33-3：動畫設計**

**33-4：反彈球遊戲設計**

**33-5：專題-使用tkinter處理謝爾賓斯基三角形**

這一章我們將介紹Python內建的模組tkinter製作動畫，而動畫也是設計遊戲的基礎。

**33-1：繪圖功能**

**33-1-1：建立畫布**

可以使用Canvas( )方法建立畫布物件。

tk = Tk( ) # 使用tk當視窗Tk物件

canvas = Canvas(tk, width=xx, height=yy) # xx,yy是畫布寬與高

canvas.pack( ) # 可以將畫布包裝好，這是必要的

畫布建立完成後，左上角是座標0,0，向右x軸遞增，向下y軸遞增。

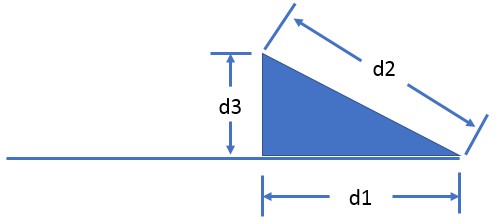
**33-1-2：繪線條create\_line( )**

它的使用方式如下：

create\_line(x1, y1, x2, y2, …, xn, yn, options)

線條將會沿著(x1,y1), (x2,y2), …繪製下去，下列是常用的options用法。

* arrow：預設是沒有箭頭，使用arrow=tk.FIRST在起始線末端有箭頭，arrow=tk.LAST在最後一條線末端有箭頭，使用arrow=tk.BOTH在兩端有箭頭。
* arrowshape：使用元組(d1, d2, d3)代表箭頭，預設是(8,10,3)。



* capstyle：這是線條終點的樣式，預設是BUTT，也可以選PROJECTING、ROUND，程式實例可以參考ch33\_4.py。
* dash：建立虛線，使用元組儲存數字資料，第一個數字是實線、第二個數字是空白、如此循環當所有元組數字用完又重新開始。例如：dash=(5,3)產生5像素實線，3像素空白，如此循環。例如：dash=(8,1,1,1)產生8像素實線和點的線條。dash=(5,)產生5實線5空白。
* dashoffset：與dash一樣產生虛線，但是一開始數字是空白的寬度。
* fill：設定線條顏色。
* joinstyle：線條相交的設定，預設是ROUND，也可以選BEVEL、MITER，程式實例可以參考ch33\_3.py。
* stipple：繪製位元圖樣(Bitmap)線條，下列是在各作業系統平台街可以使用的位元圖。程式實例可以參考ch33\_5.py。

error hourglass info questhead question

warning gray12 gray25 gray50 gray75

下列是上述位元圖由左到右、由上到下依序的圖例。

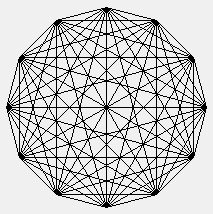
../../../Downloads/p12.jpg

* tags：為線條建立標籤，未來配合使用delete(刪除標籤)，再重繪標籤，可以創造動畫效果，可參考33-3-5節。
* width：線條寬度。

程式實例ch33\_1.py：在半徑為100的圓外圍建立12個點，然後將這些點彼此連線。



執行結果

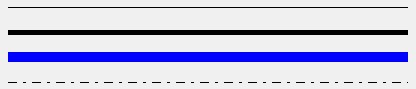


上述程式使用了數學函數sin( )和cos( )以及pi，這些是在math模組。使用create\_line( )時，在options參數欄位可以用fill設定線條顏色，用width設定線條寬度。

程式實例ch33\_2.py：不同線條顏色與寬度。



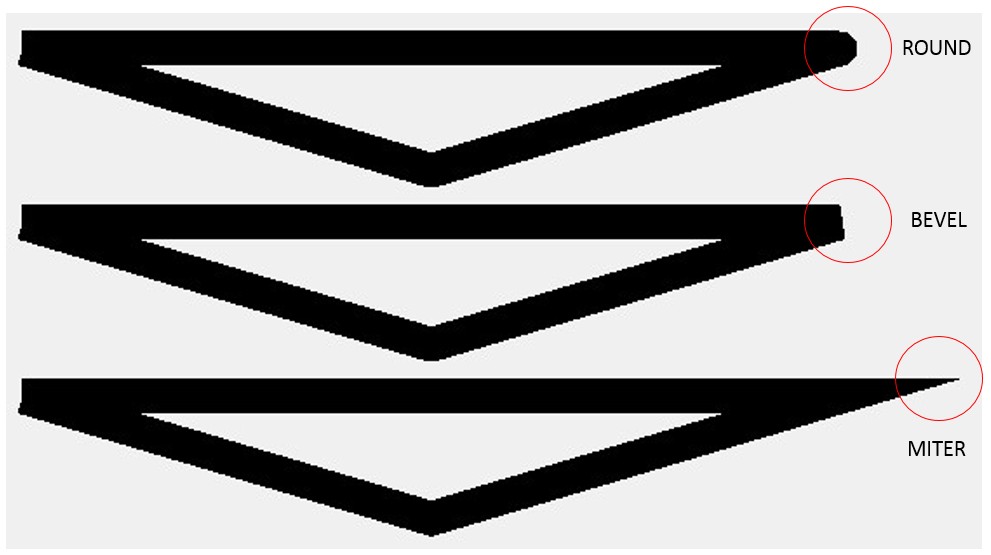
執行結果



程式實例ch33\_3.py：由線條交接了解joinstyle參數的應用。



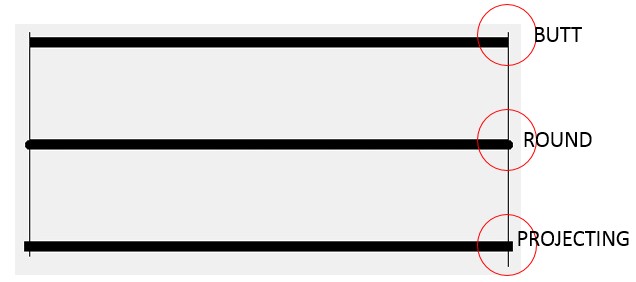
執行結果



程式實例ch33\_4.py：由線條了解capstyle參數的應用。



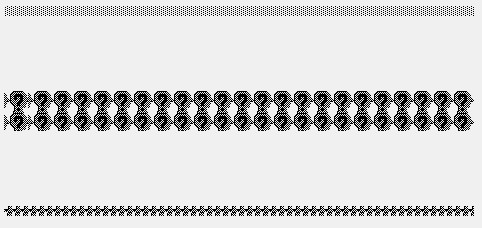
執行結果



程式實例ch33\_5.py：建立位元圖樣線條(stipple line)。



執行結果



**33-1-3：繪矩形create\_rectangle( )**

它的使用方式如下：

create\_rectangle(x1, y1, x2, y2,options)

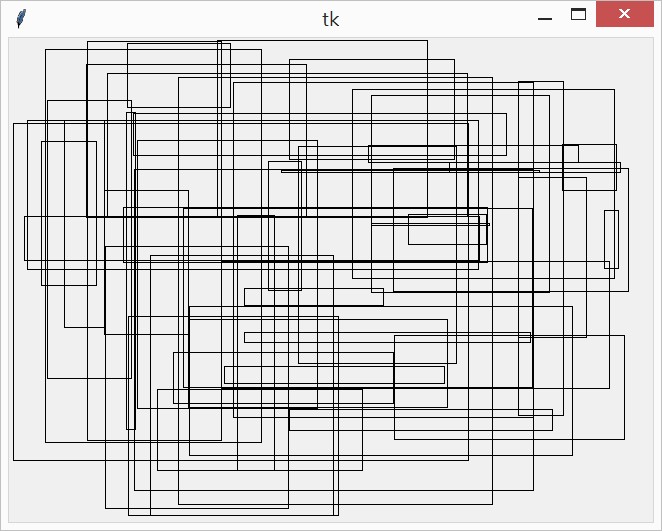
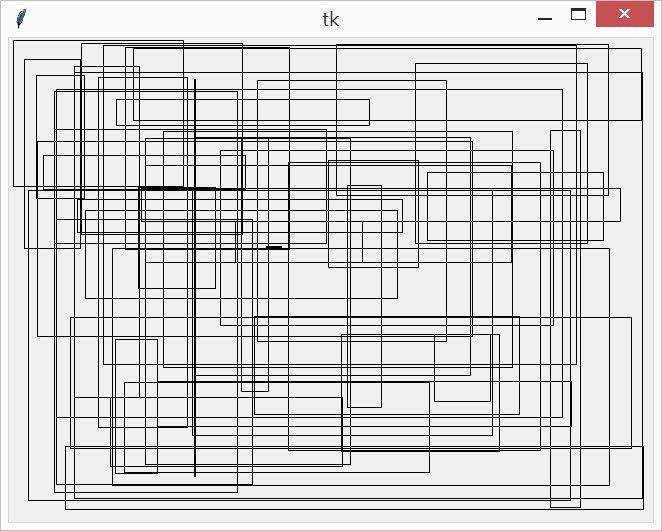
(x1,y1)和(x2,y2)是矩形左上角和右下角座標，下列是常用的options用法。

* dash：建立虛線，觀念與create\_line( )相同。
* dashoffset：與dash一樣產生虛線，但是一開始數字是空白的寬度。
* fill：矩形填充顏色。
* outline：設定矩形線條顏色。
* stipple：繪製位元圖樣(Bitmap)矩形，觀念可以參考33-1-2節，程式實例可以參考ch33\_5.py。
* tags：為矩形建立標籤，未來可以用delete創造動畫效果，可參考33-3-5節。
* width：矩形線條寬度。

程式實例ch33\_6.py：在畫布內隨機產生不同位置與大小的矩形。



執行結果

這個程式每次執行時皆會產生不同的結果，有一點藝術畫的效果。使用create\_rectangle( )時，在options參數欄位可以用fill=’color’設定矩形填充顏色，用outline=’color’設定矩形輪廓顏色。

程式實例ch33\_7.py：繪製3個矩形，第一個使用紅色填充輪廓色是預設，第二個使用黃色填充輪廓是藍色，第三個使用綠色填充輪廓是灰色。



執行結果



由執行結果可以發現由於畫布底色是淺灰色，所以第三個矩形用灰色輪廓，幾乎看不到輪廓線，另外也可以用width設定矩形輪廓的寬度。

**33-1-4：繪圓弧create\_arc( )**

它的使用方式如下：

create\_arc(x1, y1, x2, y2, extent=angle, style=ARC, options)

(x1,y1)和(x2,y2) 分別是包圍圓形矩形左上角和右下角座標，下列是常用的options用法。

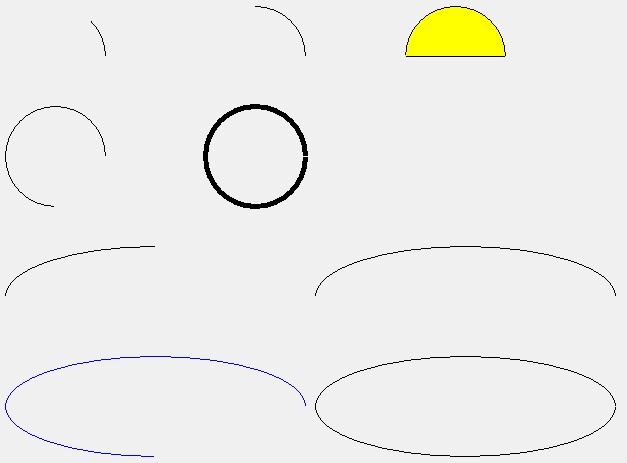
* dash：建立虛線，觀念與create\_line( )相同。
* dashoffset：與dash一樣產生虛線，但是一開始數字是空白的寬度。
* extent：如果要繪圓形extent值是359，如果寫360會視為0度。如果extent是介於1-359，則是繪製這個角度的圓弧。
* fill：填充圓弧顏色。
* outline：設定圓弧線條顏色。
* start：圓弧起點位置。
* stipple：繪製位元圖樣(Bitmap)圓弧。
* style：有3種格式，ARC、CHORD、PIESLICE，可參考ch33\_9.py。
* tags：為圓弧建立標籤，未來可以用delete創造動畫效果，可參考33-3-5節。
* width：圓弧線條寬度。

上述style=ARC表示繪製圓弧，如果是要使用options參數填滿圓弧則需捨去此參數。此外，options參數可以使用width設定輪廓線條寬度(可參考下列ch33\_8.py第12行)，outline設定輪廓線條顏色(可參考下列ch33\_8.py第16行)，fill設定填充顏色(可參考下列ch33\_8.py第10行)。目前預設繪圓弧的起點是右邊，也可以用start=0代表，也可以由設定start的值更改圓弧的起點，方向是逆時針，可參考ch33\_8.py第14行。

程式實例ch33\_8.py：繪製各種不同的圓和橢圓，以及圓弧和橢圓弧。



執行結果



程式實例ch33\_9.py：style參數是ARC、CHORD、PIESLICE參數的應用。



執行結果

../../../Downloads/19-9-2.jpg

**33-1-5：繪製圓或橢圓create\_oval( )**

它的使用方式如下：

create\_oval(x1, y1, x2, y2, options)

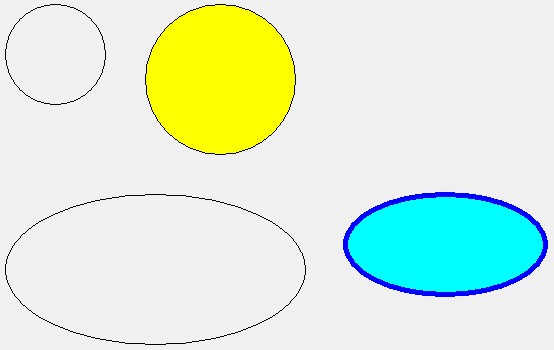
(x1,y1)和(x2,y2) 分別是包圍圓形矩形左上角和右下角座標，下列是常用的options用法。

* dash：建立虛線，觀念與create\_line( )相同。
* dashoffset：與dash一樣產生虛線，但是一開始數字是空白的寬度。
* fill：設定圓或橢圓的填充顏色。
* outline：設定圓或橢圓邊界顏色
* stipple：繪製位元圖樣(Bitmap)邊界的圓或橢圓。
* tags：為圓建立標籤，未來可以用delete創造動畫效果，可參考33-3-5節。
* width：圓或橢圓線條寬度。

程式實例ch33\_10.py：圓和橢圓的繪製。



執行結果



**33-1-6：繪製多邊形create\_polygon( )**

它的使用方式如下：

create\_polygon(x1, y1, x2, y2, x3, y3, … xn, yn, options)

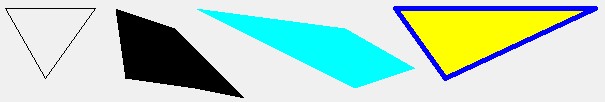
(x1,y1), … (xn,yn)是多邊形各角的x,y座標，下列是常用的options用法。

* dash：建立虛線，觀念與create\_line( )相同。
* dashoffset：與dash一樣產生虛線，但是一開始數字是空白的寬度。
* fill：設定多邊形的填充顏色。
* outline：設定多邊形的邊界顏色
* stipple：繪製位元圖樣(Bitmap)邊界的多邊形。
* tags：為多邊形建立標籤，未來可以用delete創造動畫效果，可參考33-3-5節。
* width：多邊形線條寬度。

程式實例ch33\_11.py：繪製多邊形的應用。



執行結果



**33-1-7：輸出文字create\_text( )**

它的使用方式如下：

create\_text(x,y,text=字串, options)

預設(x,y)是文字串輸出的中心座標，下列是常用的options用法。

* anchor：預設是anchor=CENTER，也可以參考18-5節的位置觀念。
* fill：文字顏色。
* font：字型的使用，觀念可以參考18-2節。
* justify：當輸出多行時，預設是靠左LEFT，更多觀念可以參考18-2節。
* stipple：繪製位元圖樣(Bitmap)線條的文字，預設是””表示實線。
* text：輸出的文字。
* tags：為文字建立標籤，未來可以用delete創造動畫效果，可參考33-3-5節。
* width：多邊形線條寬度。

程式實例ch33\_12.py：輸出文字的應用。



執行結果



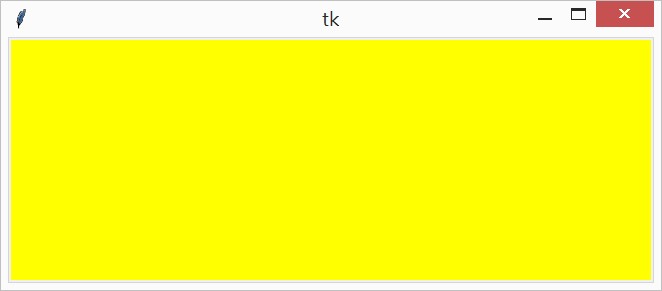
**33-1-8：更改畫布背景顏色**

在使用Canvas( )方法建立畫布時，可以加上bg參數建立畫布背景顏色。

程式實例ch33\_13.py：將畫布背景改成黃色。



執行結果



**33-1-9：插入影像create\_image( )**

在Canvas控件內可以使用create\_image( )在Canvas物件內插入影像檔，它的語法如下：

create\_image(x, y, options)

(x,y)是影像左上角的位置，下列是常用的options用法。

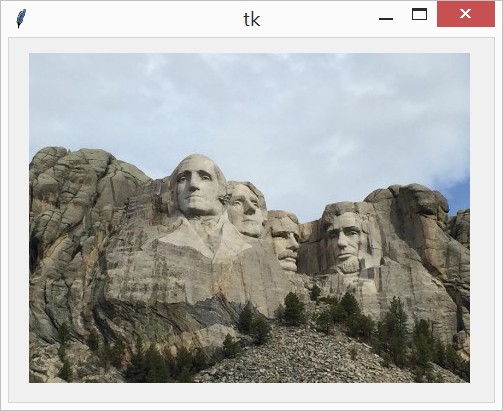
* anchor：預設是anchor=CENTER，也可以參考18-5節的位置觀念。
* image：插入的影像。
* tags：為影像建立標籤，未來可用delete創造動畫效果，可參考33-3-5節 。

下列將以實例解說。

程式實例ch33\_14.py：插入影像檔案rushmore.jpg，這個程式會建立視窗，其中在x軸建立大於影像寬度30像素，y軸則是大於影像寬度20像素。



執行結果



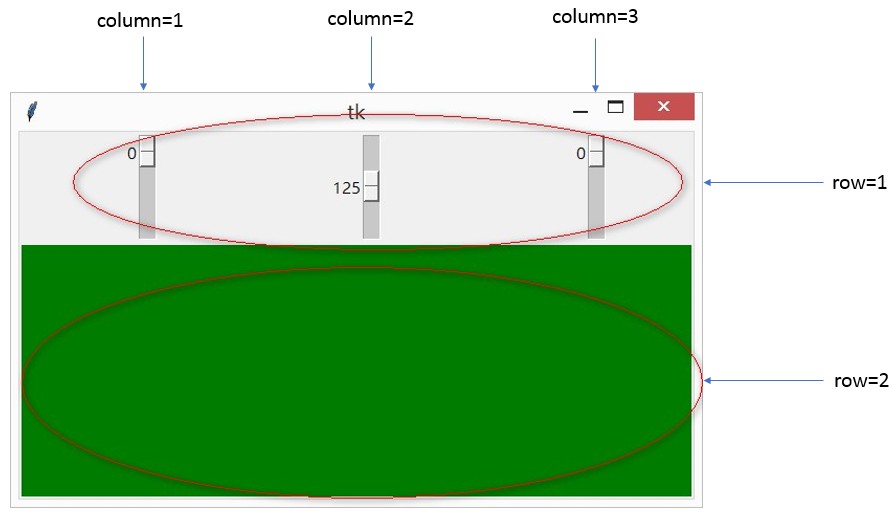
**33-2：尺度控制畫布背景顏色**

前一章筆者有介紹tkinter模組的尺度Scale( )，利用這個方法我們可以獲得尺度的值，下列將會利用3個尺度控制色彩的R、G、B值，然後可以控制畫布背景顏色。

程式實例ch33\_15.py：使用尺度控制畫布背景顏色，其中為了讓讀者了解設定尺度初值的方法，第17行特別設定gSlider的尺度初值為125。這個程式在執行時，若是有捲動尺度將呼叫bfUpdate(source )函數，source在此是語法需要，實質沒有作用。第10行config( )方法是需要使用16進位方式設定背景色，格式是”#007d00”。第18-20行的grid( )方法是定義尺度和畫布的位置，第20行的columnspan=3是設定將3個欄位組成一個欄位。此外，本程式在執行時也同時可以在Python Shell視窗看到R、G、B值的變化。



執行結果



**33-3：動畫設計**

**33-3-1：基本動畫**

動畫設計所使用的方法是move( )，使用格式如下：

canvas.move(ID, xMove, yMove) # ID是物件編號

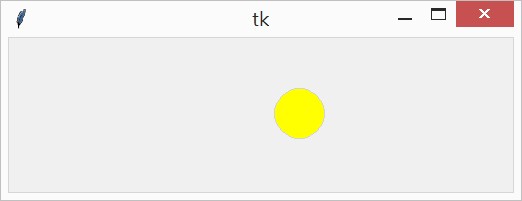
canvas.update( ) # 強制重繪畫布

xMove,yMove是x,y軸移動距離，單位是像素。

程式實例ch33\_16.py：移動球的設計，每次移動5像素。



執行結果



上述第8行筆者執行canvas.create\_oval( )時，會傳回1，所以第10行的canvas.move( )的第一個參數是指第8行所建的物件。上述執行時筆者使用迴圈，第12行相當於定義每隔0.05秒移動一次。其實我們只要設定move( )方法的參數就可以往任意方向移動。

程式實例ch33\_17.py：擴大畫布高度為300，每次移動x軸移動5, y軸移動2。

../../../Downloads/c32-15.jpg

執行結果：讀者可以自行體會球往右下方移動。

上述我們使用time.sleep(s)建立時間的延遲，s是秒。其實我們也可以使用canvas.after(s)建立時間延遲，s是千分之一秒，這時可以省略import time，可以參考ch33\_17\_1.py。

程式實例ch33\_17\_1.py：重新設計ch33\_17.py。



執行結果：與ch33\_17.py相同。

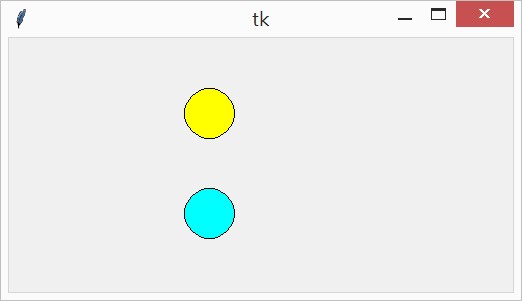
**33-3-2：多個球移動的設計**

在建立球物件時，可以設定id值，未來可以利用這個id值放入move( )方法內，告知是移動這個球。

程式實例ch33\_18.py：一次移動2個球，第8行設定黃色球是id1，第9行設定水藍色球是id2，。



執行結果



**33-3-3：將隨機數應用在多個球體的移動**

在拉斯維加或是澳門賭場，常可以看到機器賽馬的賭具，其實我們若是將球改成賽馬意義是相同的。

* 觀念1：賭場可以作弊方式

假設筆者想讓黃色球跑的速度快一些，他贏的機率是70%，可以利用randint( )產生1-100的隨機數，讓隨機數1-70間移動黃球，71-100間移動水藍色球，這樣筆者就動手腳了。

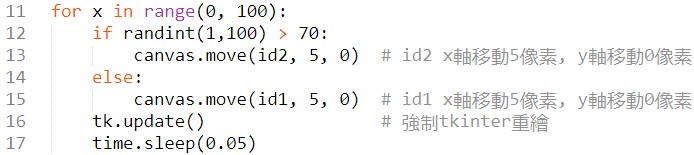
* 觀念2：賭場作弊現形

當我們玩賽馬賭具時必需下注，如果賭場要作弊最佳方式是，讓下注最少的馬匹有較高機率的移動機會，這樣錢潮就滾滾而來了，很久以來筆者已經不碰這類的遊戲了。

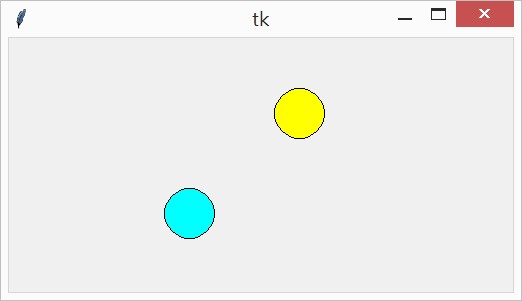
* 觀念3：不作弊

我們可以設計隨機數1-50間移動黃球，51-100間移動水藍色球。

程式實例ch33\_19.py：讓迴圈跑100次看那一個球跑得快，讓黃色球每次有70%取得移動的機會。



執行結果



**33-3-4：訊息綁定**

主要觀念是可以利用系統接收到鍵盤的訊息，做出反應。例如：當發生按下右移鍵時，可以控制球往右邊移動，例如：假設Canvas( )產生的元件的名稱是canvas，我們可以這樣設計函數。

def ballMove(event):

canvas.move(1, 5, 0) # 假設移動5像素

在程式設計函數中對於按下右移鍵移動球可以這樣設計。

def ballMove(event):

if event.keysym == ‘Right’:

canvas.move(1, 5, 0)

對於主程式而言需使用canvas.bind\_all( )函數，執行訊息綁定工作，它的寫法如下：

canvas.bind\_all(‘<KeyPress-Left>’, ballMove) # 左移鍵

canvas.bind\_all(‘<KeyPress-Right>’, ballMove) # 右移鍵

canvas.bind\_all(‘<KeyPress-Up>’, ballMove) # 上移鍵

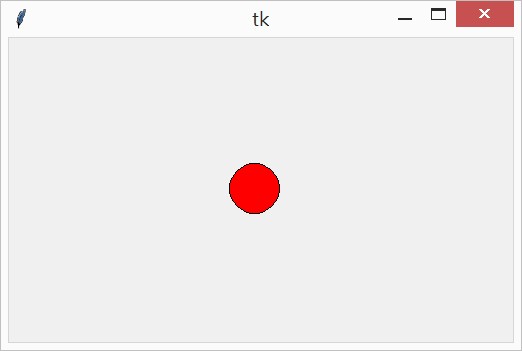
canvas.bind\_all(‘<KeyPress-Down>’, ballMove) # 下移鍵

上述函數主要是告知程式所接收到鍵盤的訊息是什麼，然後呼叫ballMove( )函數執行鍵盤訊息的工作。

程式實例ch33\_20.py：程式開始執行時，在畫布中央有一個紅球，可以按鍵盤的向右、向左、向上、向下鍵，往右、往左、往上、往下移動球，每次移動5個像素。



執行結果



**33-3-5：再談動畫設計**

在33-1節筆者介紹了tkinter的繪圖功能，在該節的繪圖方法的參數中筆者有說明可以使用tags參數將所繪製的物件標上名稱，有了這個tags名稱，未來可以用canvas.delete(“tags名稱”)刪除此物件，然後我們可以在新位置再繪製一次此物件，即可以達到物件移動的目的。

註：，如果要刪除畫布內所有物件可以使用canvas.delete(“all”)。

前一小節筆者介紹了鍵盤的訊息綁定，其實我們也可以使用下面方式執行滑鼠的訊息綁定。

canvas.bind(‘<Button-1>’, callback) # 按一下滑鼠左鍵執行callback方法

canvas.bind(‘<Button-2>’, callback) # 按一下滑鼠中鍵執行callback方法

canvas.bind(‘<Button-3>’, callback) # 按一下滑鼠右鍵執行callback方法

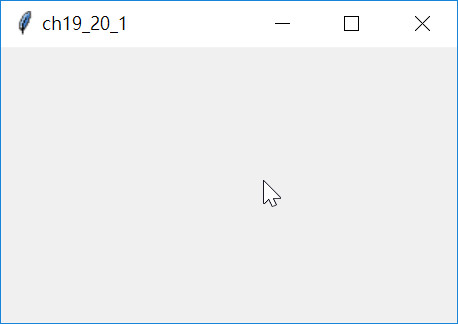
canvas.bind(‘<Motion>’, callback) # 滑鼠移動執行callback方法

上述按一下時，滑鼠相對元件的位置會被存入事件的x和y變數。

程式實例ch33\_20\_1.py：滑鼠事件的基本應用，這個程式在執行時會建立300x180的視窗，當有按一下滑鼠左邊鍵時，在Python Shell視窗會列出按一下事件時的滑鼠座標。



執行結果



下列是Python Shell示範輸出畫面。

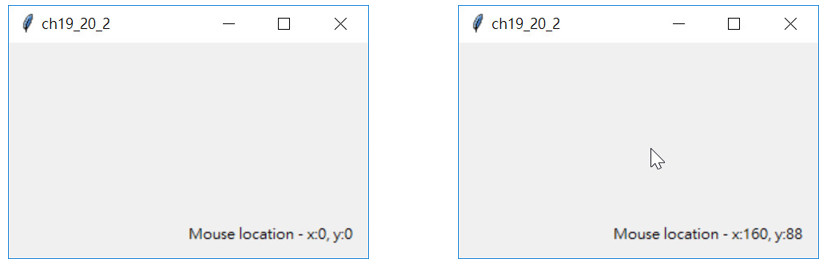


在程式第3行綁定的事件處理程式中必需留意，callback(event)需有參數event，event名稱可以自取，這是因為事件會傳遞事件物件給此事件處理程式。

程式實例ch33\_20\_2.py：移動滑鼠時可以在視窗右下方看到滑鼠目前的座標。



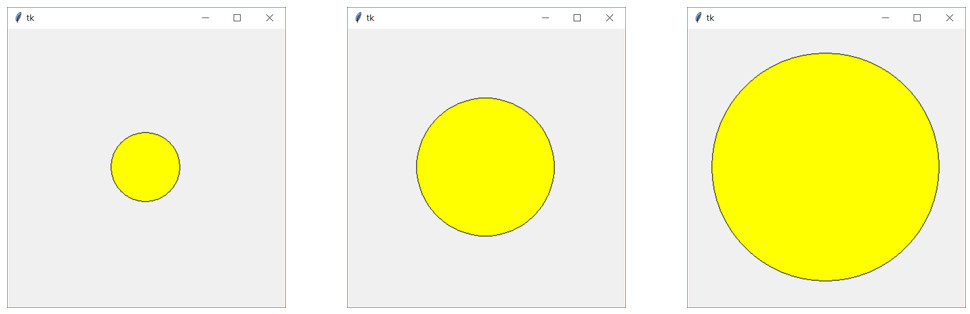
執行結果



程式實例ch33\_20\_3.py：按一下滑鼠左鍵可以放大圓，按一下滑鼠右鍵可以縮小圓。



執行結果



**33-4：反彈球遊戲設計**

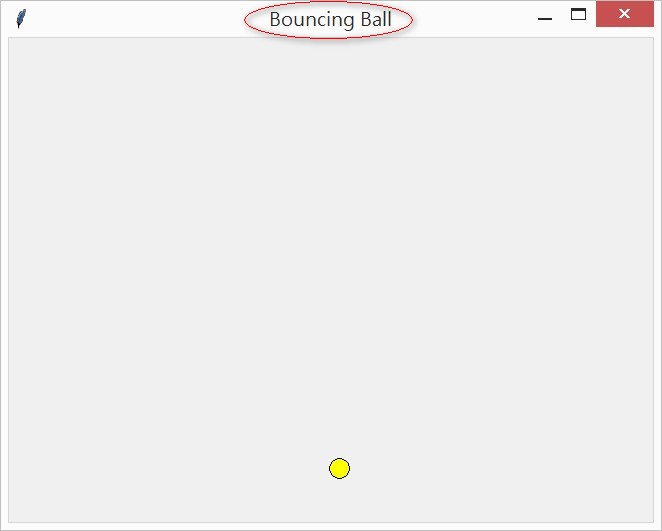
這一節筆者將一步一步引導讀者設計一個反彈球的遊戲。

**33-4-1：設計球往下移動**

程式實例ch33\_21.py：定義畫布視窗名稱為Bouncing Ball，同時定義畫布寬度(14行)與高度(15行)分別為640,480。這個球將往下移動然後消失，移到超出畫布範圍就消失了。



執行結果



這個程式由於是一個無限迴圈(28-31行)所以我們強制關閉畫布視窗時，將在Python Shell視窗看到錯誤訊息，這無所謂，本章最後實例筆者會改良程式此情況。整個程式可以用球每次移動的步伐(16行)和迴圈第31行time.sleep(speed)指令的speed值，控制球的移動速度。

上述程式筆者建立了Ball類別，這個類別在初始化\_\_init\_\_( )方法中，我們在第9行建立了球物件，第10行先設定球是大約在中間位置。另外我們建立了ballMove( )方法，這個方法會依step變數移動，在此例每次往下移動。

**33-4-2：設計讓球上下反彈**

如果想讓所設計的球上下反彈，首先需了解Tkinter模組如何定義物件的位置，其實以這個實例而言，可以使用coords( )方法獲得物件位置，它的傳回值是物件的左上角和右下角座標。

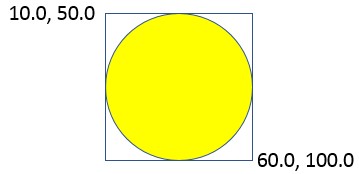
程式實例ch33\_22.py：主要是建立一個球，然後用coords( )方法列出球位置的訊息。



執行結果

../../../Downloads/19-22-1.jpg

若以上述執行結果，可以用下列圖示做解說。



相當於可以用coords( )方法獲得下列結果。

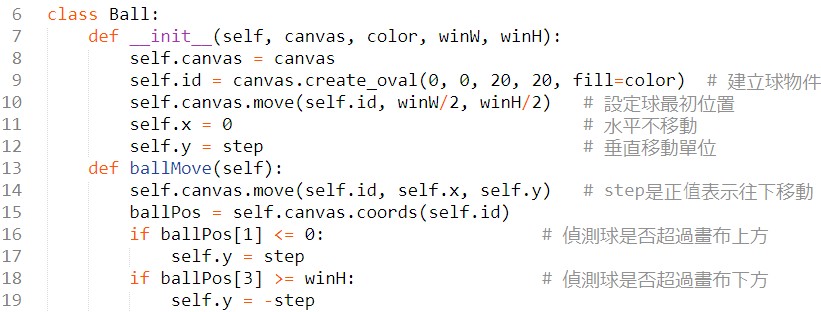
ballPos[0]：球的左邊x軸座標，未來可用於判別是否撞到畫布左方。

ballPos[1]：球的上邊y軸座標，未來可用於判別是否撞到畫布上方。

ballPos[2]：球的右邊x軸座標，未來可用於判別是否撞到畫布右方。

ballPos[3]：球的左邊y軸座標，未來可用於判別是否撞到畫布下方。

程式實例ch33\_23.py：改良ch33\_21.py，設計讓球可以上下方移動，其實這個程式只是更改Ball類別內容。



執行結果：讀者可以觀察螢幕，球上下移動的結果。

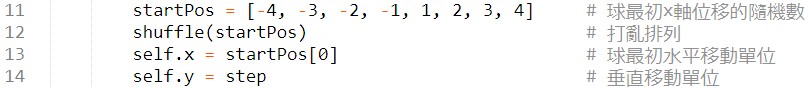
程式第11行定義球x軸不移動，第12行定義y軸移動單位是step。第15行獲得球的位置資訊，第16-17行偵測如果球撞到畫布上方未來球移動是往下移動step單位，第18-19行偵測如果球撞到畫布下方未來球移動是往上移動step單位(因為是負值)。

**33-4-3：設計讓球在畫布四面反彈**

在反彈球遊戲中，我們必需讓球在四面皆可反彈，這時需考慮到球在x軸移動，這時原先Ball類別的\_\_init\_\_( )函數需修改下列2行。

../../../Downloads/p5%20(17).jpg

下列是更改結果。



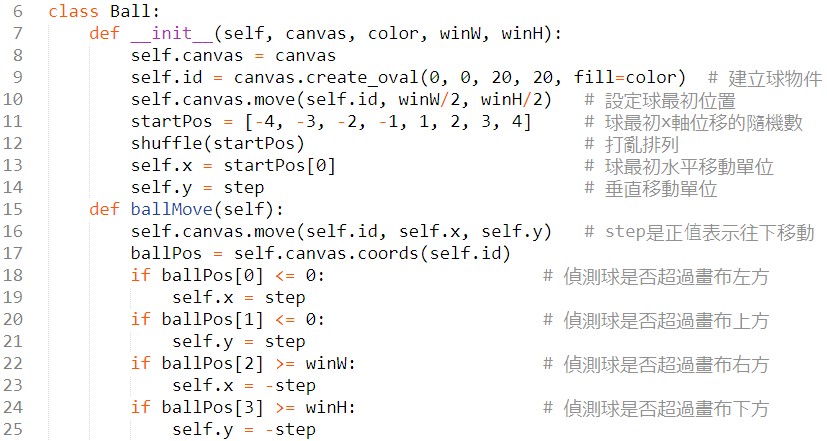
上述修改的觀念是球局開始時，每個迴圈x軸的移動單位是隨機數產生。至於在ballMove( )方法中，我們需考慮到水平軸的移動可能碰撞畫布左邊與右邊的狀況，觀念是如果球撞到畫布左邊，設定球未來x軸移動是正值，也就是往右移動。

../../../Downloads/p7%20(16).jpg

如果球撞到畫布右邊，設定球未來x軸移動是負值，也就是往左移動。

../../../Downloads/p8%20(13).jpg

程式實例ch33\_24.py：改良ch33\_23.py程式，現在球可以在四周移動。

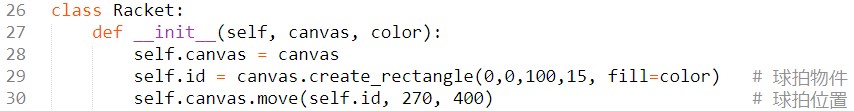


執行結果：讀者可以觀察螢幕，球在畫布四周移動的結果。

**33-4-4：建立球拍**

首先我們先建立一個靜止的球拍，此時可以建立Racket類別，在這個類別中我們設定了它的初始大小與位置。

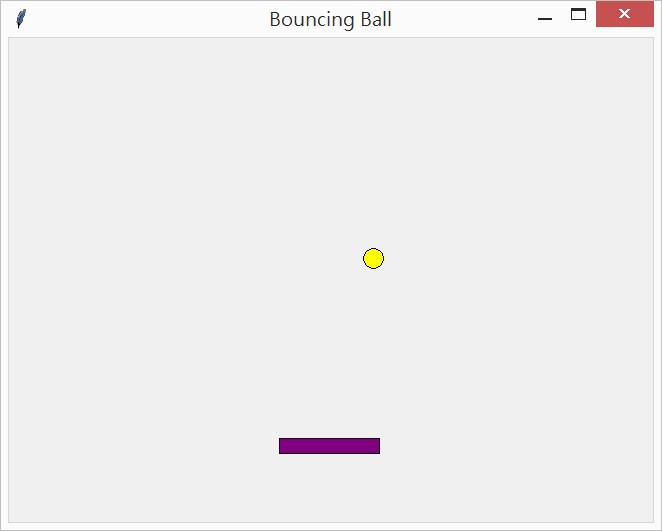
程式實例ch33\_25.py：擴充ch33\_24.py，主要是增加球拍設計，在這裡我們先增加球拍類別。在這個類別中，我們在第29行設計了球拍的大小和顏色，第30行設定了最初球拍的位置。



另外，在主程式增加了建立一個球拍物件。

../../../Downloads/f32-23-1.jpg

執行結果

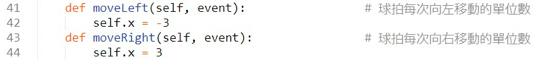


**33-4-5：設計球拍移動**

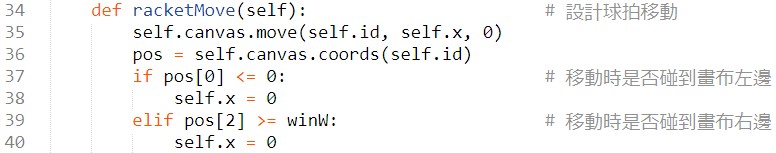
由於是假設使用鍵盤的右移和左移鍵移動球拍，所以可以在Ractet的\_\_init\_\_( )函數內增加，使用bind\_all( )方法綁定鍵盤按鍵發生時的移動方式。

../../../Downloads/p9%20(10).jpg

所以在Ractet類別內增加下列moveRight( )和moveLeft( )的設計。



上述設計相當於每次的位移量是3，如果遊戲有設等級，可以讓新手位移量增加，隨等級增加讓位移量減少。此外這個程式增加了球拍移動主體設計如下：



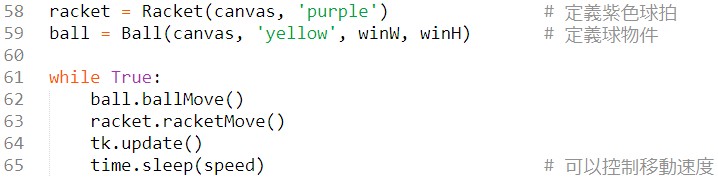
主程式也將新增球拍移動呼叫。



程式實例ch33\_26.py：擴充ch33\_25.py的功能，增加設計讓球拍左右可以移動，下列程式第31行是設定程式開始時，球拍位移是0，下列是球拍類別內容。



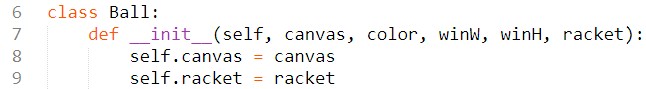
下列是主程式內容。



執行結果：讀者可以觀察螢幕，球拍已經可以左右移動。

**33-4-6：球拍與球碰撞的處理**

在上述程式的執行結果中，球碰到球拍基本上是可以穿透過去，這一節將講解碰撞的處理，首先我們可以增加將Racket類別傳給Ball類別，如下所示：



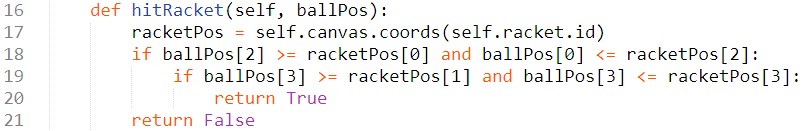
當然在主程式建立Ball類別物件時需修改呼叫如下：

../../../Downloads/p13%20(10).jpg

在Ball類別需增加是否球碰到球拍的方法，如果碰到就讓球路徑往上反彈。

../../../Downloads/p14%20(7).jpg

在Ball類別ballMove( )方法上方需增加下列hitRacket( )方法，檢測是否球碰撞球拍，如果碰撞了會傳回True，否則傳回False。

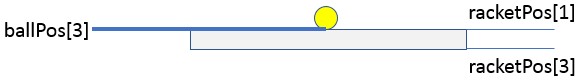


上述偵測是否球撞到球拍的必需符合2個條件：

1：球的右側x軸座標ballPos[2]大於球拍左側x座標racketPos[0]，同時球的左側x座標ballPos[0]小於球拍右側x座標racketPos[2]。



2：球的下方y座標ballPos[3]大於球拍上方的y座標racketPos[1]，同時必需小於球拍下方的y座標reaketPos[3]。讀者可能奇怪為何不是偵測碰到球拍上方即可，主要是球不是一次移動1像素，如果移動3像素，很可能會跳過球拍上方。



下列是球的可能移動方式圖。



程式實例ch33\_27.py：擴充ch33\_26.py，當球碰撞到球拍時會反彈，下列是完整的Ball類別設計。



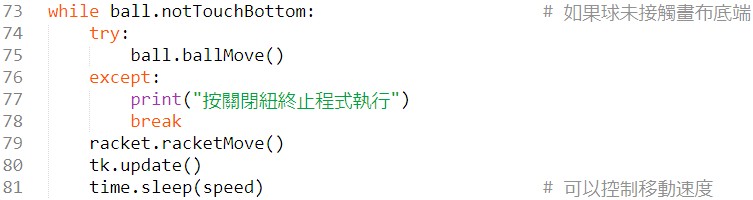
執行結果：讀者可以觀察螢幕，球碰撞到球拍時會反彈。

**33-4-7：完整的遊戲**

在實際的遊戲中，若是球碰觸畫布底端應該讓遊戲結束，此時首先我們在第16行Ball類別的\_\_init\_\_( )函數中先宣告notTouchBottom為True，為了讓玩家可以緩衝，筆者此時也設定球局開始時球是往上移動(第15行)，如下所示：

../../../Downloads/p19%20(3).jpg

我們修改主程式的循環如下：



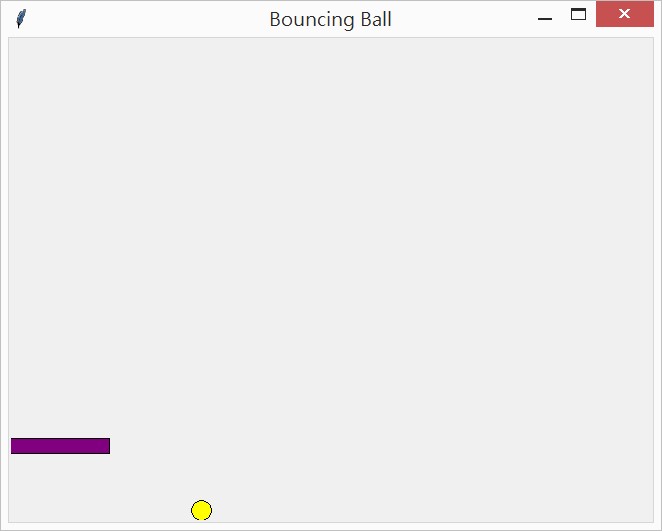
最後我們在Ball類別的ballMove( )方法中偵測球是否接觸畫布底端，如果是則將notTouchBottom設為False，這個False將讓主程式的迴圈中止執行。同時用捕捉異常時處理如果按Bouncing Ball視窗的關閉鈕，這樣就不會再有錯誤訊息產生了。

程式實例ch33\_28.py：完整的反彈球設計。





執行結果



**33-5：專題-使用tkinter處理謝爾賓斯基三角形**

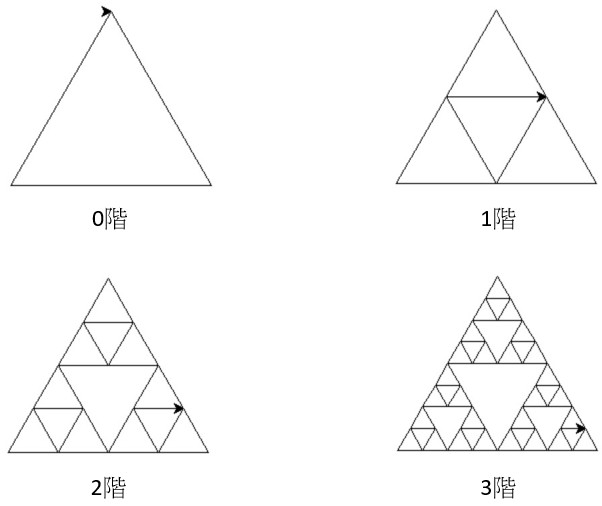
謝爾賓斯基三角形(Sierpinski triangle)是由波蘭數學家謝爾賓斯基在1915年提出的三角形觀念，這個三角形本質上是碎形(Fractal)，所謂碎形是一個幾何圖形，它可以分為許多部分，每個部分皆是整體的縮小版。這個三角形建立觀念如下：

1：建立一個等邊三角形，這個三角形稱0階(order = 0) 謝爾賓斯基三角形。

2：將三角形各邊中點連接，稱1階謝爾賓斯基三角形。

3：中間三角形不變，將其它3個三角形各邊中點連接，稱2階謝爾賓斯基三角形。

4：使用11-6節遞廻式函數觀念，重複上述步驟，即可產生3、4 … 或更高階謝爾賓斯基三角形。



使用tkinter解這個題目最大的優點是我們可以在GUI介面隨時更改階乘數字，然後可以在畫布顯示執行結果。

在這個小節我們計畫介紹另一個元件(widget)框架Frame，也可將此想像成是容器元件，這個框架Frame通常用在碰上複雜的GUI介面設計時，可以將部分其它tkinter元件組織在此框架內(可想成是容器)，如此可以簡化GUI介面。它的建構方法語法如下：

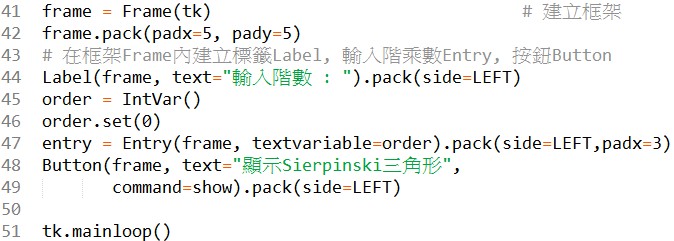
Frame(父物件,options, … )

Frame( )方法的第一個參數是父物件，表示這個框架將建立在那一個父物件內。下列是Frame( )方法內其它常用的options參數：

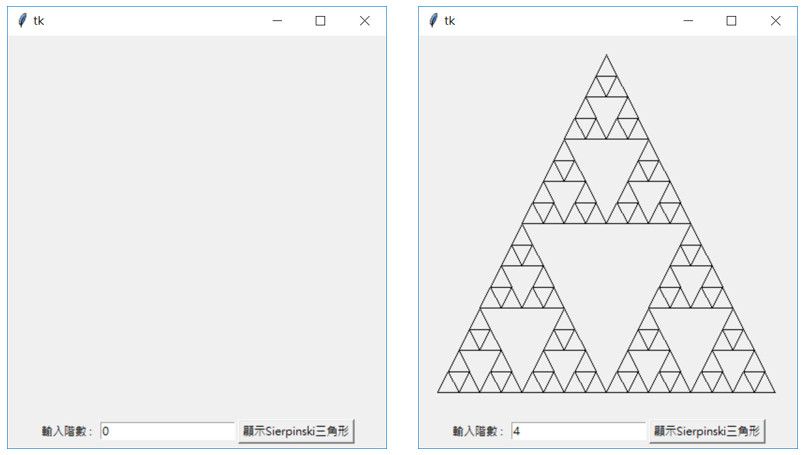
* bg或background：背景色彩。
* borderwidth或bd：標籤邊界寬度，預設是2。
* cursor：當滑鼠游標在框架時的游標外形。
* height：框架的高度單位是像素。
* highlightbackground：當框架沒有取得焦點時的顏色。
* highlightcolor：當框架取得焦點時的顏色。
* highlighthickness：當框架取得焦點時的厚度。
* relief：預設是relief=FLAT，可由此控制框架外框。
* width：框架的高度單位是像素，省略時會自行調整為實際寬度。

程式實例ch33\_29.py：設計謝爾賓斯基三角形(Sierpinski triangle)，這個程式基本觀念是在tk視窗內分別建立Canvas( )物件canvas和Frame( )物件frame，然後在canvas物件內繪製謝爾賓斯基三角形。在frame物件內建立標籤Label、文字方塊Entry和按鈕Button，這是用於建立輸入繪製謝爾賓斯基三角形的階乘數與正式控制執行。

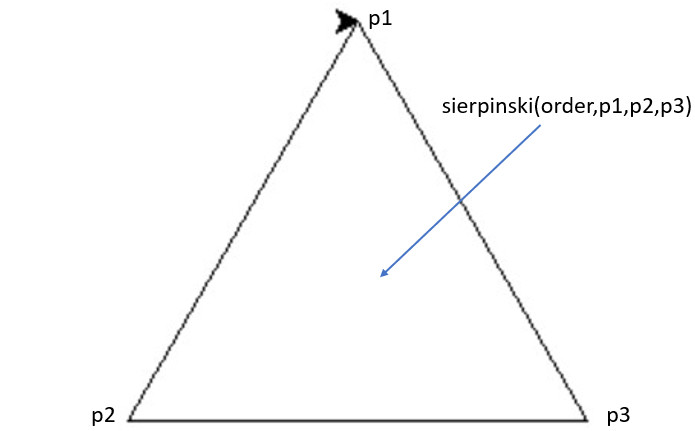




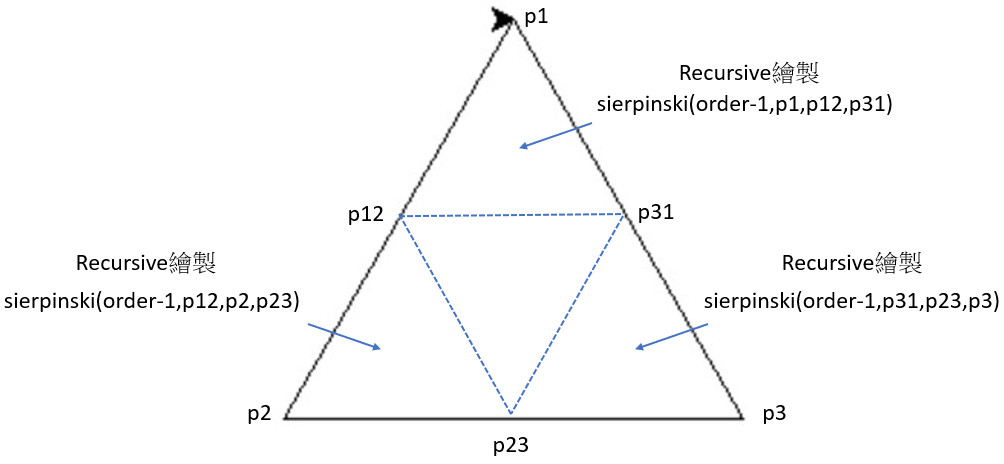
執行結果



上述程式繪製第一個0階的謝爾賓斯基三角形觀念如下：

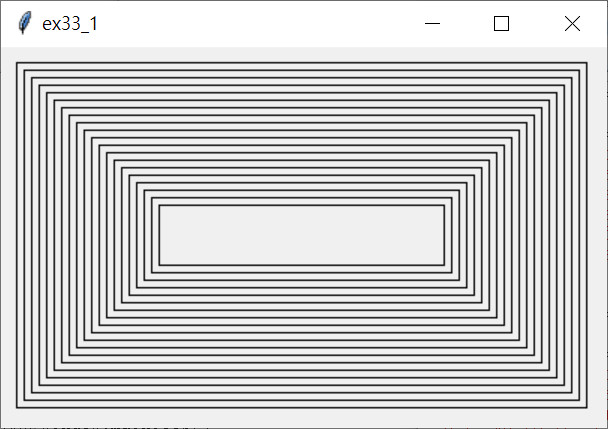


遞廻呼叫繪製謝爾賓斯基三角形觀念如下：



**習題實作題**

1：寫一個程式，畫布寬高分別是400\*250，由外往內繪製，每次寬和高減10，可以顯示20個矩形。(33-1節)



2：寫一個程式，畫布寬高分別是400\*250，由外往內繪製橢圓，每次橢圓寬和高減10，可以顯示20個橢圓形。(33-1節)



3：寫一個程式可以顯示15\*15的網格。(33-1節)



4：寫一個程式可以顯示走馬燈訊息。(33-3節)



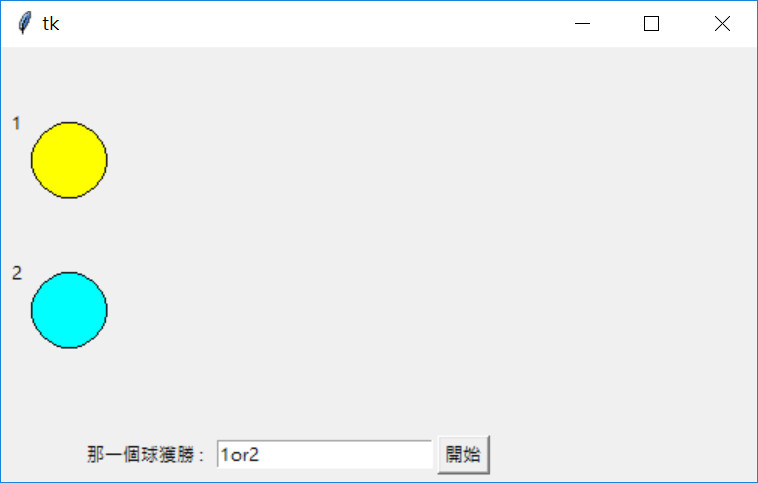
5：寫一個程式，當按up、down、right、left鍵盤時，可以繪製線條。(33-3節)



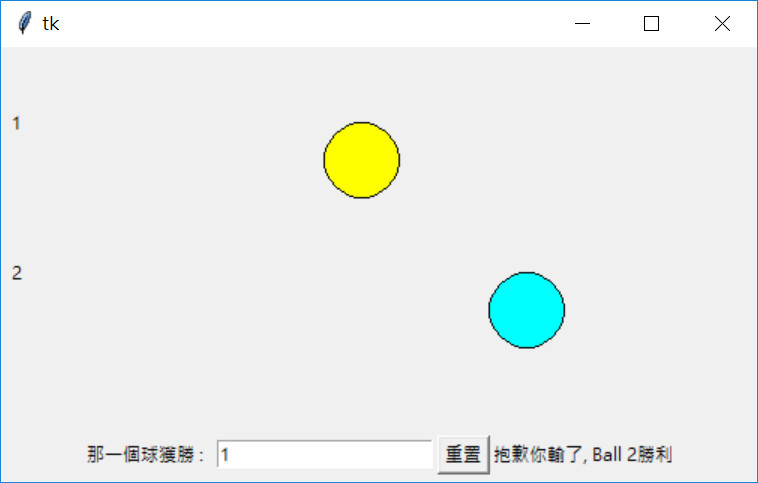
6：繪製含有3片葉子的風扇，視窗的寬度與高度是300\*300，風扇的半徑是120，其它風扇顏色與轉動細節則可以自行發揮。(33-3節)



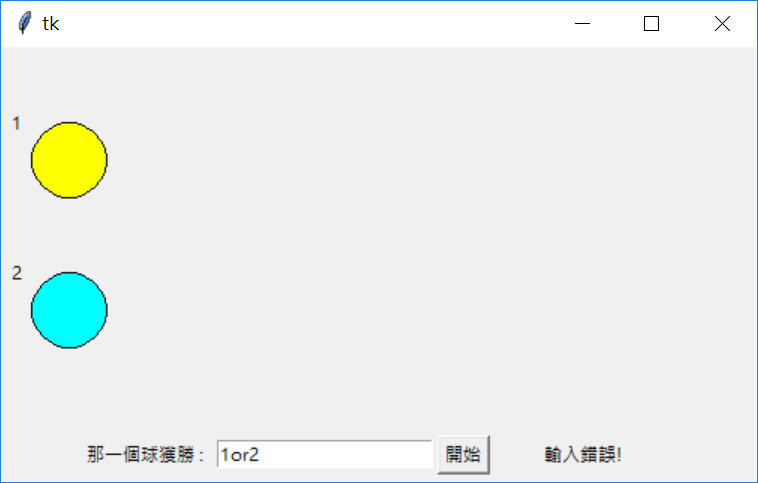
7：重新設計程式實例ch19\_19.py，輸出字串讓玩家由螢幕輸入猜哪一個球跑得快，每次移動時皆讓電腦有60%移動的機率。下列是開始畫面：(33-3節)



下列是選擇1號球勝利，結果是2號球勝利的畫面。



下列是輸入錯誤的畫面。



8：參考ch19\_29.py，繪製一個遞廻樹Recursive Tree，假設樹的分支是直角，下一層的樹枝長度是前一層的0.6倍，下列是不同深度depth時的遞廻樹。(33-5節)

