Python turtle 模組的進階應用

上次程式作業介紹了 Python 的 turtle 模組，我們可以利用簡單的指令來操作烏龜的行走路徑，以下介紹 turtle 模組更進階的應用。

繪製幾何圖形

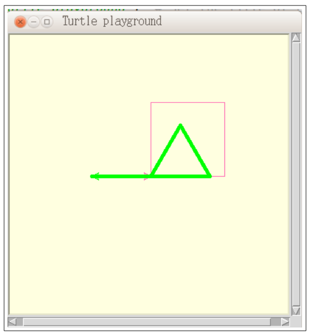
我們可以利用烏龜來繪製幾何圖形，以下為 turtleGeometry.py 範例程式：先產生一個視窗(Screen)，設定視窗大小、標題、及背景顏色，然後誕生一隻名叫 square 的粉紅小烏龜，再操作烏龜前進及轉向便可以繪製一個正方形，最後使用者滑鼠點擊視窗結束程式。

```
# turtleGeometry.py

import turtle                      # Allows us to use turtles

screen = turtle.Screen()           # Creates a playground for turtles
screen.setup(400, 400)             # Set the size of the screen
screen.title('Turtle playground')  # Set the title of the screen
screen.bgcolor('lightyellow')      # Set the background color of the screen

# Draw a square
square = turtle.Turtle()     # Create a turtle, assign to 'square'
square.color('hotpink','')   # Set the color of the turtle
for i in range(4):           # Draw a square
    square.forward(100)
    square.left(90)

screen.exitonclick()         # Wait for user to close the screen
```

註：顏色選項詳見這裡。

可以再加一些程式來繪製三角形：誕生一隻名為 triangle 的綠色小烏龜來畫三角形，再加入以下程式片段。

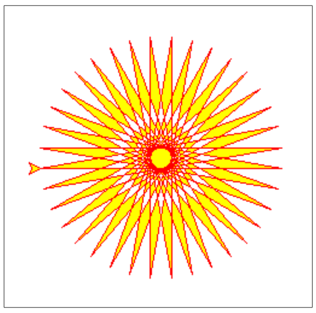
```
# Draw a triangle
triangle = turtle.Turtle()   # Create a turtle, assign to 'triangle'
triangle.color('green','')   # Set the color of the turtle
triangle.pensize(5)          # Set the pen size
for i in range(3):           # Draw an equilateral triangle
    triangle.forward(80)
    triangle.left(120)
triangle.right(180)          # Turn around
triangle.forward(80)         # Move away from the origin
```



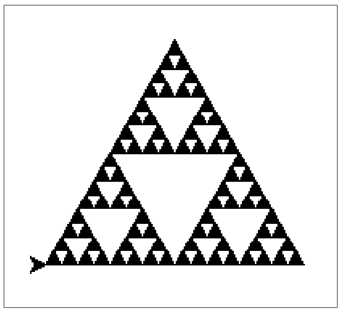
繪製漂亮圖案

turtle 模組可以用來繪製許多漂亮圖案:

```
# turtleStar.py

import turtle

screen = turtle.Screen()
star = turtle.Turtle()
star.color('red', 'yellow')

star.begin_fill()
while True:
    star.forward(200)
    star.left(170)
    if abs(star.pos()) < 1:
        break;
star.end_fill()

screen.exitonclick()
```



```
# turtleSTriangle.py
# (Sierpinsky triangle algorithm)

import turtle

def striangle(myTurtle, depth, base):
    myTurtle.down()
    if depth == 0:
        myTurtle.begin_fill()
        for i in 0,1,2:
            myTurtle.forward(base)
            myTurtle.left(120)
        myTurtle.end_fill()
    else:
        for i in 0,1,2:
            striangle(myTurtle, depth-1, base)
            myTurtle.up()
            myTurtle.forward(base*2**depth)
            myTurtle.left(120)
            myTurtle.down()

screen = turtle.Screen()
```



```
triangle = turtle.Turtle()
triangle.speed(0)
triangle.reset()
striangle(triangle, 4, 10)
screen.exitonclick()
```

繪製路徑

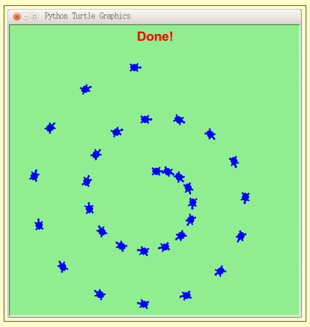
可以利用 penup()或 pendown()來設定是否要畫路徑，例如在 turtleGeometry.py 的第一個 for 迴圈做以下修改：

```
for i in range(4):            # Draw a square
    if i==1:
        square.penup()        # Skip the second side
    else:
        square.pendown()
    square.forward(100)
    square.left(90)
```

留下印記

烏龜在行進中可以留下印記(Stamp)，例如 turtleStamp.py：

```
# turtleStamp.py

import turtle

screen = turtle.Screen()
screen.setup(600,600)
screen.bgcolor("lightgreen")

myStamp = turtle.Turtle(visible=False)
myStamp.shape("turtle")
myStamp.color("blue")

# myStamp.speed(8)
myStamp.penup()              # Do not draw the path
stepLen = 20
for i in range(31):
    myStamp.stamp()          # Leave an impression on the canvas
    stepLen = stepLen + 3    # Increase the step length on every iteration
    myStamp.forward(stepLen) # Move along
    myStamp.right(24)        # and turn

myStamp.penup()              # Do not draw the path
myStamp.goto(0, 260)         # Move
myStamp.color('red')
myStamp.write('Done!', align='center', font=('Arial', 20, 'bold'))

screen.exitonclick()
```



改變一下 myStamp 烏龜的速度，刪除 `mySṭamp.speed(8)`指令的註解符號試試。註：速度值從 1 到 10 為從慢到快，但 0 為最快(沒有速限)。

計算距離

利用 `distance(x)`方法指令可計算烏龜位置與 x 的距離，x 可為一個位置向量(x, y)、或另一隻烏龜，例如 turtleDistance.py：

```
# turtleDistance.py

import turtle

screen = turtle.Screen()
turtleA, turtleB = turtle.Turtle(), turtle.Turtle()
turtleB.goto(10, 20)

print(turtleA.distance((30,40)), ',', turtleA.distance(turtleB))

screen.exitonclick()
```

範例：繪製 1000 個彩色多邊形

turtlePolygon.py 在視窗中繪製許多彩色多邊形：

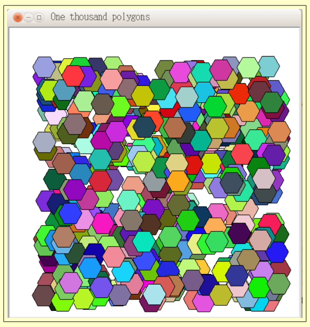
1. 首先撰寫繪製多邊形的函式 `polygon()`，共有 3 個輸入參數：烏龜物件、邊的數量、邊的長度，然後透過操作烏龜的行進，繪製多邊形。
2. 利用 `screen.trace(False)`設定在繪製多邊形的過程中不要更新螢幕，待全部繪製完畢再利用 `screen.tracer(True)`一次整體呈現。
3. For 迴圈執行下列指令：

   3.1 利用 `random` 方法隨機選取一個點(xpos, ypos)準備在該處繪製多邊形。

   3.2 利用 `random` 方法隨機選取一個顏色(red, green, blue)作為烏龜及多邊形的顏色。

   3.3 利用 `fillcolor()`、`begin_fill()`、及 `end_fill()`方法分別填滿烏龜及多邊形顏色。
4. 最後 `exitonclick()`讓使用者在視窗中以滑鼠點擊來結束程式。

```
# turtlePolygon.py

import turtle
import random

# Function to draw the polygon
def polygon(t, sides, length):
    for x in range(sides):
        t.forward(length)
        t.right(360/sides)

print('This program draws colorful polygons.')
numSides,  sideLength  ,  numPolygons=  eval(input('Polygons:  number  of  sides,
side length, total number (e.g.: 4, 40, 1000): '))
screen = turtle.Screen()
myTurtle = turtle.Turtle()

screen.title("One thousand polygons")
screen.setup(500, 500)

myTurtle.hideturtle()
screen.tracer(False)    # Do not update the graphics
for x in range(numPolygons):
    # Choose a random spot
    xpos = random.randint(-200,200)
    ypos = random.randint(-200,200)

    # Goto this spot
    myTurtle.penup()
    myTurtle.goto(xpos, ypos)
    myTurtle.pendown()

    # Generate a random color
    red = random.random() # returns a number between 0 and 1
    green = random.random()
    blue = random.random()

    # fill in our shape
    myTurtle.fillcolor(red, green, blue)

    # Draw the polygon
    myTurtle.begin_fill()
    polygon(myTurtle, numSides, sideLength)
    myTurtle.end_fill()

# Update the screen with our drawing
screen.tracer(True)
screen.exitonclick()
```



練習

繪製以下五角星形以及類似時鐘的圖形: