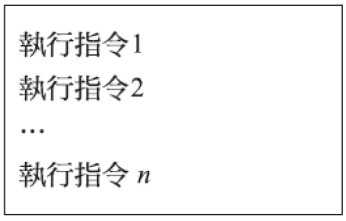
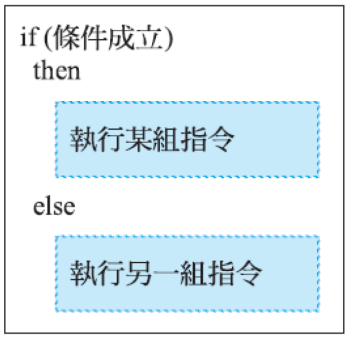
重複結構

演算法通常是由下列三種結構所組成：

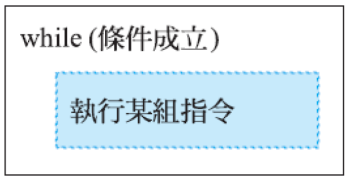
- 序列（sequence）
- 決策（decision）
- 重覆（repetition）



（a）序列結構



（b）決策結構



（c）重覆結構

挑戰題目: 自我介紹

請用for迴圈，輸出六位參與者的自我介紹和最喜歡的甜點。

- 六位參與者，他們的名字分別是 Mario, Peach, Luigi, Daisy, Toad和Yoshi。
- 喜歡的甜點依次為Star Pudding, Peach Pie, Popsicles, Honey Cake, Cookies, Jelly Beans。

預期結果如下：

Hi, my name is Mario. My favorite dessert is Star Pudding.

Hi, my name is Peach. My favorite dessert is Peach Pie.

....

```
## Input Data
people = ["Mario", "Peach", "Luigi", "Daisy", "Toad",
"Yoshi"]
desserts = ["Star Pudding", "Peach Pie", "Popsicles",
"Honey Cake", "Cookies", "Jelly Beans"]

## Process
for index in range(len(people)): # range(0,6,1)
    name = people[index]
    dessert = desserts[index]
    print(f"Hi! My name is {name}. My favorite dessert is
{dessert}.")
```

從這個例子中，同學要瞭解的觀念有：

(1) 串列可以儲存一系列的資料，不只是純量(單一的數字、文字、布林)。

(2) 使用range()函數可以更方便地處理數字串列。

(3) 學會for計數迴圈搭配串列的使用方法。

(4) 學會for計數迴圈搭配range()函數的使用方法。

重複結構 (一)計數迴圈

1. 計數迴圈（一）:for與資料容器

1-1 資料容器 - 串列

1. 把串列想像成一列火車，火車有一系列的車廂，每個車廂都可以裝東西(資料)。

2. 語法

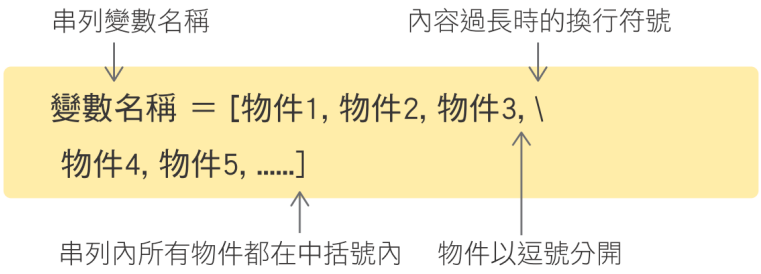


圖 4-2.2　串列的定義

- 串列將資料放在方括號裡面。
- 每個項目都是以逗號分隔。
- 串列項目的索引編號i從0開始。
- 索引編號為i的資料用`變數名稱[i]`取得。

範例1：購物清單

請用串列列出你最喜愛的五樣東西，當成購物清單(shopping_list)！

```
## 列出購物清單
shopping_list = ['牛奶', '蛋', '咖啡豆', '西瓜', '鳳梨']
print(shopping_list)
## 取出購物清單上面第三樣物品，也就是索引編號為2的商品咖啡豆
shopping_list[2]
```

範例2：建立串列資料使用中括號 []

```
## 建立串列資料
## 數字清單
```

```
list1 = [1,2,3,4,5] # 數字資料
print(list1)
## 文字清單
list2 = ["1", "2", "3", "4","5"] # 文字資料
print(list2)
## 混合資料型態
list3 = [1, 2.0, 'Python', True] # 串列可包含各種資料型別元素
print(list3)

## 使用list()函數建立串列資料
list4 = list('python') # 字串會自動轉串列，將每個字元看成一個單位
print(list4)
list5 = list(12345)  # 數值無法轉串列，因為沒有更小的單位
print(list5)
```

本課程的資料以純量為主，Python除串列之外，還有其他資料容器，高年級課程再做介紹。

1-2 for 迴圈與串列

1. for迴圈會逐次取出串列中每一個元素，再對每一個元素進行運算。

2. for迴圈語法

```
for 暫時變數 in 資料容器：
    程式區塊 （暫時變數逐一取出、逐一處理）
```

- 每次從資料容器（串列）裡只拿一個東西
- 每個東西都會依序拿一次。
- 暫時變數 = 這次拿出來的東西。
- 執行程式區塊
  - 第一行結尾必須加上冒號。
  - 迴圈內所有程式碼必須向右縮四個空格並且對齊。

範例3: 使用「for」迴圈讀取串列中的每一個元素

- 暫時變數輪流等於串列中的每個項目值

- 透過索引值的變動
- 由不同的索引值，取得串列對應索引項目的值

```
## 1. 文字串列
names = ["Allen", "James", "Tom", "Jack"]
## 計數迴圈：把names串列中的東西一個一個直接拿出來
for i in names: #每次拿出來的東西暫時叫它i
    print('Welcome to Class!', i)  #清單東西都拿完就結束
```

```
## 2. 數字串列
## 可用[]，也可用range(資料量大時，通常用range)
## 方法(一): for迴圈 + 資料容器(數字串列)
for i in [0,1,2,3,4]:
    print(i)
## 方法(二):for迴圈 + range()函數(下面馬上教到)
for i in range(5): # range(5) = [0,1,2,3,4]
    print(i)
```

範例4: 文字串列的索引應用 - 購物清單

```
## 位置索引與項目(一)
sheet = ['牛奶', '蛋', '咖啡豆']
index = [0,1,2] # index是索引值, sheet[index]是購物項目
for i in index:
    print (i)

for i in sheet: # 購物清單迴圈
    print (i)

## 位置索引與項目(二)
for i in index:
    print (i, sheet[i]) #i是索引值，sheet[i]是項目

## 位置索引與項目(三) (下面馬上教到)
for i in range(len(sheet)): # length()函數計算串列長度(項目數)
    print (i, sheet[i])
```

【隨堂練習1】好朋友名單

請寫一個for迴圈列印自己的好朋友名單，如下:

```
我有幾位好朋友:
Michael 是我的好朋友
Tom 是我的好朋友
Andy 是我的好朋友
June 是我的好朋友
Axel 是我的好朋友
```

```
friends = ["Michael","Tom", "Andy", "June", "Axel"]
print("我有幾位好朋友：")
for index in friends:
    print(index.title()+"是我的好朋友")
```

2. 計數迴圈（二）

2-1 range()函數

1. range()函數可以產生數字序列，給我們某一個範圍的數字清單。

2. 用法

```
# range函數的表示方法:
range（start, stop, step）
```

- **range函數的參數預設值**
  - **初值start的預設值是0**
    - 若省略初值(start)，則變數從第0個位置開始。
  - **遞增值step的預設值是1**
    - 若省略遞增值(step)，則變數每次自動+1
    - 若step = –1時， 則表示變數每次遞減1。
  - **終值stop的預設結尾值，不包含終值、是到終值的前一個數字。**
    - 終值沒有預設值，不能省略。
  - 變數等於終值時離開，所以程式執行的最後一個值是終值的前一個數字。
- **變數會依據初值到終值的變化，依序指定給左邊的變數名稱。**
- **數字串列中的元素，可以依照索引(index)或切片(slice)取得。**

    - 串列中項目的索引編號i是從0開始。
    - 變數索引可用`變數名稱[i]`的方式取得。
        - `d[0]` 代表變數 d 的第 1 筆資料內容。
        - 變數切片可用`變數名稱[i:j]`的方式取得。
            - `d[1:4]`代表變數 d 第 2 筆資料內容到第4筆資料內容。
- **range()函數的三種寫法**
    - range(終值) =>range(0, stop, 1)
    - range(初值, 終值) => range(start, stop, 1)
    - range(初值, 終值, 遞增值)

範例5：range()函數的使用

```
## list1 = [0,1,2,3,4]
a = range(0,5,1) # a = range(0,5), a = range(5)
print(a)

## 要用變數索引才知道用range()函數輸入的元素是什麼
print(a[0], a[1], a[2], a[3], a[4])

## list2 = [2,3,4,5]
b = range(2, 6, 1) # b = range(2,6)
print(b[0], b[1], b[2], b[3]) # 四筆資料，index從0到3

## list3 = [2,5,8]
c = range(2, 10, 3) # step為正，表示座標往右走、一次跳3個數
print(c[0], c[1], c[2]) # 三筆資料，index從0到2

## list4 = [10,8,6,4]
d = range(10, 3, -2)  # step為負，表示座標往左走、一次跳2個數
print(d[0], d[1], d[2], d[3]) # 四筆資料，index從0到3
```

【隨堂練習2】：請用range()函數建立下面的數字串列，並用索引將裡面的項目一一取出。

(1) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

(2) [0, 2, 4, 6, 8]

(3) [5, 7, 9]

(4) [0, -1, -2, -3, -4, -5, -6, -7, -8, -9]

```
a = range(0, 10, 1)
print(a[0], a[1], a[2], a[3], a[4], a[5]. a[6], a[7],
a[8], a[9])

b = range(0, 9, 2)
print(b[0], b[1], b[2], b[3], b[4])

c = range(5, 10, 2)
print(c[0], c[1], c[2])

d = range(0, -10, -1)
print(d[0], d[1], d[2], d[3], d[4], d[5], d[6], d[7],
d[8], d[9])
```

2-2 for迴圈與range函數

1. for迴圈的語法：

通常for迴圈和range函數一起搭配使用。

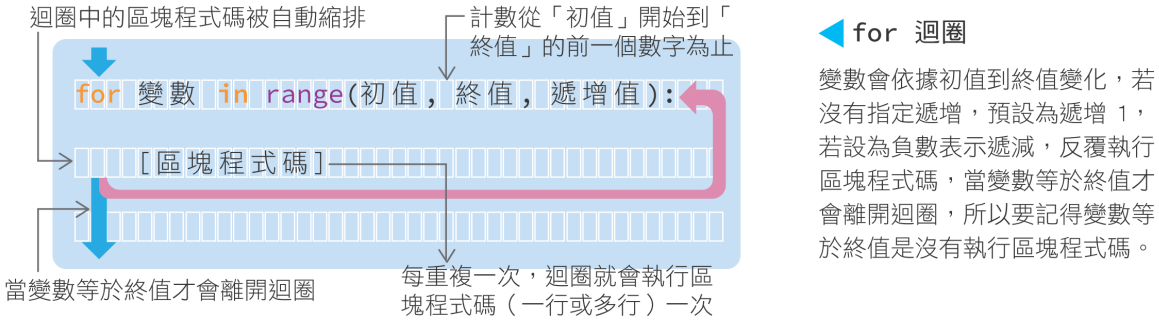


圖 4-3.4　for 迴圈語法說明

- for迴圈，透過range函數，指定迴圈變數的初值、終值與遞增(減) 值。
  - 迴圈變數由初值到終值的前一個數字為止，不包括終值。
  - 每重複執行一次，迴圈變數就依照遞增(減) 值遞增或遞減，並執行迴圈內程式。
- for迴圈指令的**最後面要加「冒號：」**，代表底下區塊程式碼的敘述要開始。

- **區塊程式碼需要縮排**，每行一律空四個空格。
  - Python用「縮排上下對齊」的方式，判斷是否在同一程式區塊。
  - 其他程式語言大多是利用大括號 { } 代表執行區塊。

題型（一）：列印

範例6：文字列印，印出5個Hello。

~ 虛擬碼

for迴圈用i來數數，若i在數字範圍1~5內(range(0,5,1))，就列印Hello，超過就停止。

~ 流程圖

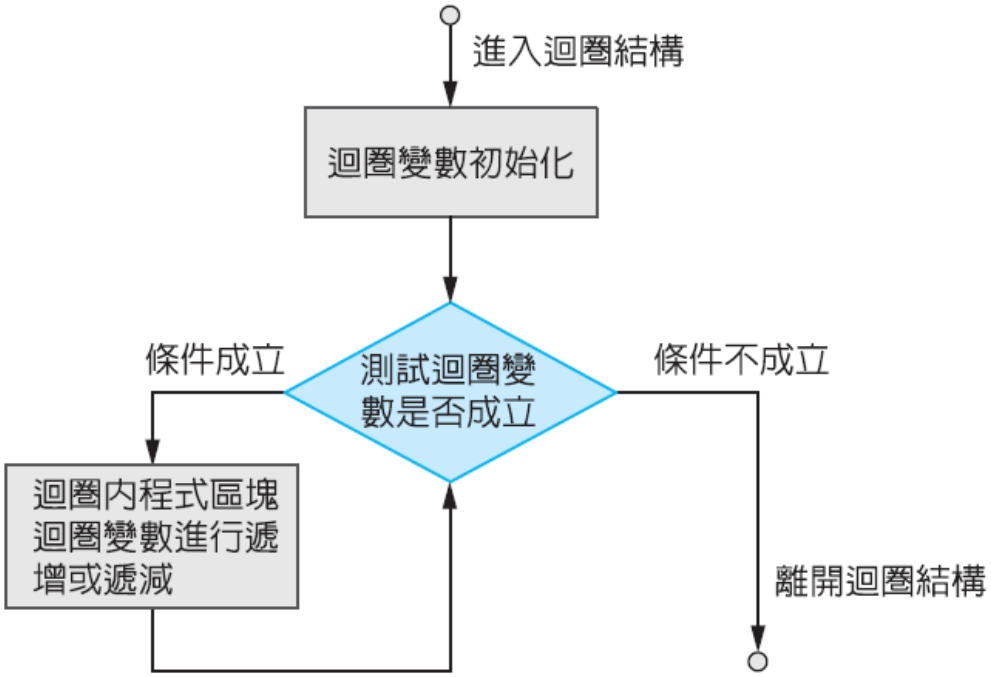


圖 A-11　流程圖

```
## 印出5個Hello
for i in range(0,5,1): #[0,1,2,3,4]
    print("Hello")
```

範例 7: 數字列印，印出range()函數值的範圍

```
## [1,2,3,4,5]
for i in range(1, 6, 1):
```

```
    print(i) # 預設為下一行 \n

## [0,1,2,3,4]
for i in range(5): #  range(0,5,1)
    print(i) # 預設為下一行 \n

## [2,3,4]
for i in range(2,5):
    print(i, end = " ") # 橫著列印，中間空一格

## [2,4,6,8]
for i in range(2,10,2):
    print(i, end = " ")


## [100,97,94,91]
for i in range(100, 90, -3):
    print(i, end = " ")
```

=> 一般而言，for迴圈使用的比while迴圈多，而for迴圈與range()函數的搭配用的又比for迴圈與串列多。

【隨堂練習3】：統計摩天輪旋轉次數

小芳在兒童樂園玩摩天輪，她很想知道玩一次摩天輪，到底能轉多少圈。可是她一上摩天輪就緊張，為了不出錯，每次旋轉到最高處，自己拍一張照片。回家後查看照片數，並做好記錄，經過查看照片共統計出摩天輪轉了9次。請問如何用Python語言寫出與下面用手寫做出的相同紀錄。

```
1 @
2 @@
3 @@@
4 @@@@
5 @@@@@
6 @@@@@@
7 @@@@@@@
8 @@@@@@@@
9 @@@@@@@@@
摩天輪總共轉了 9 圈。
```

```
counter = 0
mark = ""
for i in range(1, 10, 1):
    counter = counter + 1
    mark = mark + "@"
    print(i, mark)
print(f"摩天輪總共轉{counter}圈")
```

題型（二）：加總

範例8：寫一個程式計算數字1到10之和。

～解題想法

- 可以用for迴圈。
- 迴圈變數
  - 起始值為加總的起始值
  - 終止值為加總的終止值
  - 每執行一次迴圈，就會依照遞增(減)值變動。
- 加總過程
  - 迴圈內使用「sum = sum + 迴圈變數」進行數值加總。
  - 等號賦值。先計算等號右邊算式「sum + i」，將結果回存到等號左邊變數(sum)。
  - sum要先給初值。

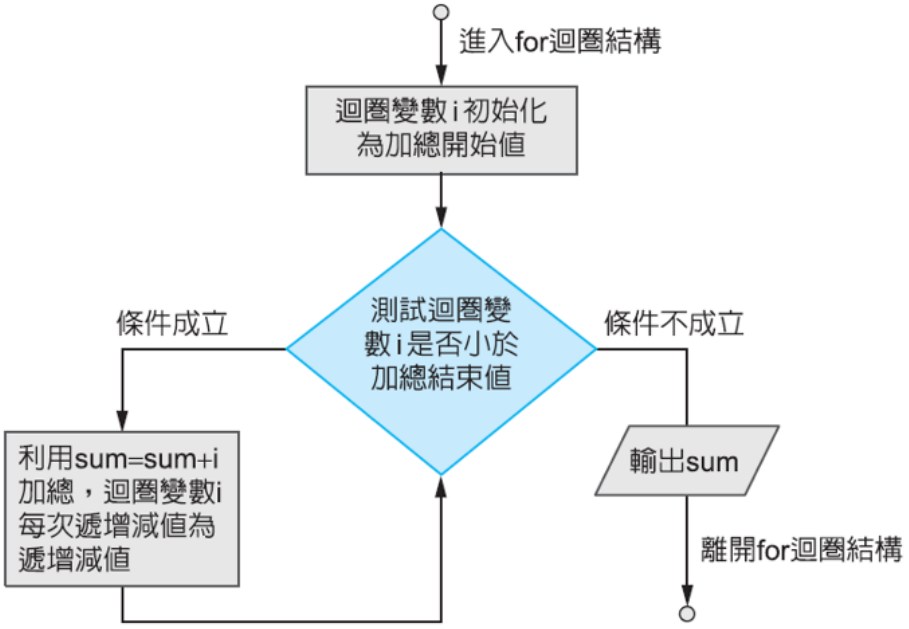


圖 A-13　流程圖

```
# 數字1到10之和。
## 1+2+3+4+5+6+7+8+9+10=?
sum = 0  # 起始條件
for i in range(1, 11, 1): #數字串[1,2,3,4,5,6,7,8,9,10]
    sum = sum + i    # 加總運算,變更條件
print("Total is", sum)
```

2-3 巢狀迴圈

巢狀迴圈是多層迴圈，為迴圈內又有迴圈的程式結構。
從外層迴圈來看，內層迴圈只是外層迴圈內的動作，因此外層迴圈作用一次，內層迴圈全部都需要執行一次。

範例9：畫星星

```
方式一：正常的5列5行
＊＊＊＊＊
＊＊＊＊＊
＊＊＊＊＊
＊＊＊＊＊
＊＊＊＊＊

方式二：左上角的直角三角形
＊＊＊＊＊
＊＊＊＊
＊＊＊
＊＊
＊

方式三：左下角的直角三角形
＊
＊＊
＊＊＊
＊＊＊＊
＊＊＊＊＊
```

```
# for 迴圈嵌套：內循環與外循環相互獨立
print("方式一：正常的5列5行")
for i in range(5):
```

```
    for j in range(5):
        print("*", end="")
    print("")

# for 迴圈嵌套：內迴圈依賴外部循環變數
print("方式二：左上角的直角三角形")
for i in range(5,0,-1):
    for j in range(i):
        print("*", end="")
    print("")

# for 迴圈嵌套：內迴圈依賴外部循環變數
print("方式三：左下角的直角三角形")
for i in range(5):
    for j in range(i + 1):
        print("*", end="")
    print("")
```

隨堂練習4: 列出1到1000不被2也不被3整除的數字

列出所有1到1000中不被2也不被3整除的所有數字，即不是2的倍數也不是3的倍數。

```
for i in range(1, 1001):
    if((i%2 != 0) and (i%3 != 0)):
        print(i, end = "\t")
```

隨堂練習5: 字串拼接成語接龍

用Python語言編制一個成語接龍遊戲。玩家輸入成語後，程序會把所有成語拼皆起來，顯示拼接的長龍。

```
saying = input("請輸入一個四個字的成語： ")
for i in range(3):
    next = input("接上一句成語，再輸入一個成語： ")
    saying += next
print(saying)
```

重複結構 (三)海龜繪圖

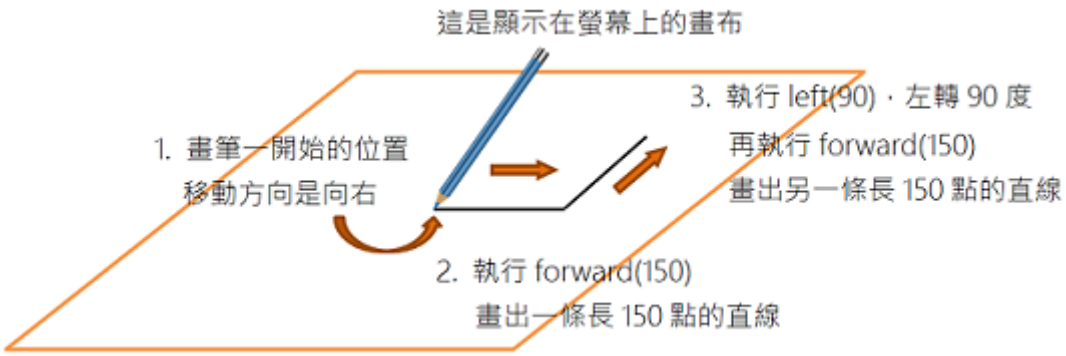
海龜圖是學習Python和使用程式碼創作藝術作品的有趣方式，我們將透過輸入指令用虛擬海龜游標在螢幕上畫圖。

1. 如何在Python畫畫

- 建議用Thonny整合開發環境使用海龜模組 (Colab無法執行)
- 使用海龜模組

要開始使用turtle模組，我們首先需要導入它。方法很簡單，直接輸入import，後面則接著我們想要使用的模組。也就是，開頭輸入`import turtle`程式碼，表示要啟動海龜模組、使用海龜圖。
此時，在螢幕上仍然看不到任何東西，但在幕後電腦已經準備好了，可以開始存取turtle模組不同片段的程式碼。

- 創造一個海龜（畫筆畫畫 => 海龜走路）



建立一個海龜物件，並且指派海龜變數名稱。

- 如shelly = turtle.Turtle()
- 改變海龜形狀，輸入shelly.shape("turtle")
    - 可以用arrow, circle, classic, square, triangle
- 找出海龜的位置
    - shelly.position()

你可以看到一個新視窗跳出，海龜就在哪裡；請控制這個視窗，並用下面的指令讓海龜游標來畫圖。這些指令都是事先寫好存在海龜模組裏的程式碼。

2. 常用指令整理

英文版：https://docs.python.org/3/library/turtle.html

簡中版：https://docs.python.org/zh-cn/3/library/turtle.html

~使用指令的方法：點記法(Dot Notation) - `物件.函數()`

一種顯示某些程式碼區塊彼此相關的方法。

- 例如要告訴電腦，我們想使用專門屬於turtle模組中的Screen物件，就是在它們之間使用點(.)，即`turtle.Screen()`。
- 如果要進一步使用屬於Screen物件的特定函式，例如更改顏色，就在Screen物件和要使用的函式間放一個點（和參數）。
  如:`turtle.Screen().bgcolor("blue")`，就表示：「請電腦找到turtle模組的Screen物件；執行此動作時，請找到屬於它的bgcolor()函式；最後，按照bgcolor()函式所說，使用我們給它的顏色blue去執行程式。」。

(1)運動命令:

- forward(d):向前移動距離d代表距離
- backward(d):向後移動距離d代表距離
- right(degree):向右轉動多少度
- left(degree):向左轉動多少度
- position():位置
- goto(x,y):將畫筆移動到座標為(x,y)的位置
- speed(speed):畫筆繪製的速度範圍[0,10]整數
- stamp():繪製當前圖形

(2)畫筆控制命令:

- penup():畫筆抬起，移動時不繪製圖形
- pendown():畫筆落下，移動時繪製圖形
- pensize(width):畫筆的寬度
- pencolor(colorstring):畫筆的顏色，white,yellow, blue, black, gold, pink, brown, purple, red, gold, seashell. tomato...
- fillcolor(colorstring):繪製圖形的填充顏色。
- color(colorstring):繪製圖形的填充顏色。例：turtle.color(color1, color2)，同時設定pencolor=color1, fillcolor=color2。

- circle(radius, extent):繪製一個圓形，其中radius為半徑，extent為度數，例如若extent為180，則畫一個半圓；畫一個圓形，可不必寫第二個參數
- shape():海龜形狀，turtle, arrow, circle, classic, square, triangle
- setheading(degree):海龜朝向，degree代表角度
- reset():恢復所有設置

3. 做中學：常用指令

```
## 啟動模組
import turtle  #導入海龜模組
shelly = turtle.Turtle()  # 把模組中的螢幕游標叫出來
shelly.shape("turtle") # 將三角形改為烏龜
shelly.position() # 烏龜的位置

## 海龜的外觀
## 海龜的顏色
shelly.color("green", "red") # 內紅邊綠
## 海龜畫線的顏色
shelly.pencolor("black") # 黑色
## 調整海龜尺寸
shelly.turtlesize(10,10,2) # (上下長度,左右寬度,外框粗細)變大
shelly.resizemode("auto") # auto表預設值，還原海龜大小
shelly.turtlesize(5,5,3) # 調整成自己喜歡的大小

## 海龜運動命令
## 畫筆粗細
shelly.width(5)
## 畫一條線長100
shelly.forward(100)
shelly.backward(100)
## 向左轉90度
shelly.left(90)
shelly.right(90)
## 提起筆離開畫布
shelly.penup()
## 放下筆貼近畫布
shelly.pendown()
## 隱藏使用的turtle
```

```
shelly.hideturtle()

## 畫筆控制
## 要填滿繪製的圖案-開始
shelly.begin_fill()
## 用什麼顏色
shelly.fillcolor("orange")
## 繪製圖形
shelly.circle(250) #半徑250的圓
shelly.circle(250, 180, 30) #半徑50，範圍180度，轉動方向30度
shelly.circle(100, 360, 3) # 三角形
shelly.circle(100, 360, 6) # 六邊形
## 要填滿繪製的圖案-停止
shelly.end_fill()

## 蓋章
shelly.stamp()
## 在螢幕寫字
shelly.write("Turtle Rock")
```

龜繪圖套件無法每次都可以執行，會有可執行、被中斷的情形交互出現的情況，除在程式開頭可加turtle.TurtleScreen._RUNNING = True指令外，個人建議重新開機再跑一次最簡單。

3-1 用線條畫出文字

範例10：凸

```
import turtle
turtle.TurtleScreen._RUNNING = True

turtle.forward(100)
turtle.right(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(100)
turtle.right(90)
```

```
turtle.forward(300)
turtle.right(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)

turtle.done()
```

有興趣者也可以練習寫「凹」、「世」、「田」、「王」等這類簡單的字。

3-2 幾何繪圖

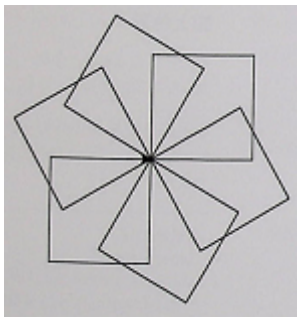
範例11：尺寸為100的正方形 - for迴圈

```
## 虛擬程式碼
- 向前移動100(250)步
- 向左轉90度

- 向前移動100步
- 向左轉90度

- 向前移動100步
- 向左轉90度

- 向前移動100步
- 向左轉90度

=> 用for迴圈寫
```

```
## 方法(一)：for 迴圈
import turtle
shelly = turtle.Turtle()

for i in range(4):       # 畫正方形 - 1 + 2 做四次 [0,1,2,3]
    shelly.forward(250) # 1.畫直線
    shelly.left(90)     # 2.轉向
```

```
    print (i)             # 計數，一共四次

## 方法(二)：for 迴圈 + 色彩(紅色)
import turtle
shelly = turtle.Turtle()
shelly.color("red")  # 顏色 - 線用紅色

for i in range(4):    # 形狀 - 正方形
    shelly.forward(250)
    shelly.left(90)
```

範例12：六個尺寸為100的正方形 - 巢狀迴圈



```
## 虛擬碼
以下重複六次:
    以下重複四次:
        前進250步
        左轉90度
    右轉60度
```

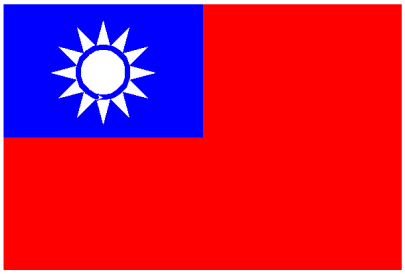
```
##方法(一)：巢狀迴圈
import turtle
shelly = turtle.Turtle()
for n in range(6):              # 外迴圈重複正方形6次
    for i in range(4):          # 內迴圈重複4次來畫正方形
        shelly.forward(250)
        shelly.left(90)
    shelly.right(60)           # 畫下一個正方形前轉彎

## 方法(二)：巢狀迴圈 + 加上色彩
import turtle
shelly = turtle.Turtle()
colors = ["red","green","blue","black","yellow", "purple"]
```

```
for n in range(6):                  # 外迴圈重複正方形6次
    shelly.color(colors[n])
    #
    for i in range(4):              # 內迴圈重複4次來畫正方形
        shelly.forward(250)
        shelly.left(90)
    shelly.right(60)                # 畫下一個正方形前轉彎

## 方法(三) ：巢狀迴圈 +填滿色彩
import turtle
shelly = turtle.Turtle()
colors = ["red","green","blue","black","purple","yellow"]

##外迴圈重複正方形6次
for n in range(6):
    shelly.color(colors[n])
    shelly.begin_fill()
    shelly.fillcolor(colors[n])

    for i in range(4):              #內迴圈重複4次來畫正方形
        shelly.forward(250)
        shelly.left(90)
    shelly.right(60)                #畫下一個正方形前轉彎
    shelly.end_fill()
```

隨堂練習6：畫星星

```
import turtle
shelly = turtle.Turtle()
shelly.color("green")
shelly.pensize(5)

## 將畫筆前移250個單位
for i in range(5):
    shelly.forward(250)
    shelly.right(144)
```

參考繪圖：中華民國國旗



```
import turtle
turtle.TurtleScreen._RUNNING = True
turtle.speed(0)

turtle.penup()
turtle.goto(-300, 200)
turtle.pendown()

turtle.color('red', 'red')
turtle.begin_fill()
turtle.forward(600)
turtle.right(90)
turtle.forward(400)
turtle.right(90)
turtle.forward(600)
turtle.right(90)
turtle.forward(400)
turtle.right(90)
turtle.end_fill()

turtle.color('blue', 'blue')
turtle.begin_fill()
turtle.forward(300)
turtle.right(90)
turtle.forward(200)
turtle.right(90)
turtle.forward(300)
turtle.right(90)
turtle.forward(200)
turtle.right(90)
turtle.end_fill()

turtle.penup()
turtle.goto(-150, 175)
turtle.pendown()
```

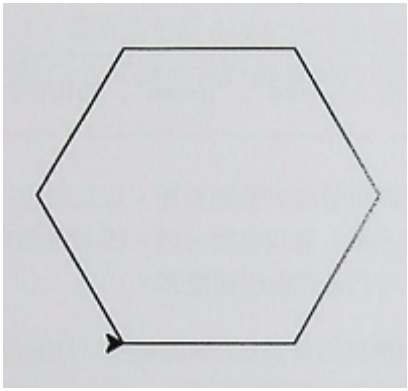
```
turtle.color('white', 'white')
turtle.begin_fill()
turtle.right(75)
for i in range(12):
    turtle.forward(150)
    turtle.right(150)
turtle.end_fill()

turtle.penup()
turtle.goto(-150, 60)
turtle.pendown()

turtle.setheading(0)
turtle.pencolor('blue')
turtle.pensize(8)
turtle.circle(38)

turtle.done()
```

4. 專題：打造你的藝術傑作

Step 1: 畫出六角型

在任何海龜專題的開頭都要先匯入海龜模組，這樣才可以使用裡面的函數。

- 建立一個名為Art.py的檔案，輸入以下程式碼並執行，確認出現海龜。
- 在頂端加入一行註解來提醒自己這個專題是什麼。
- 修改畫正方形的虛擬程式碼，變成六角形。
    - 數量是6
    - 轉彎角度是60



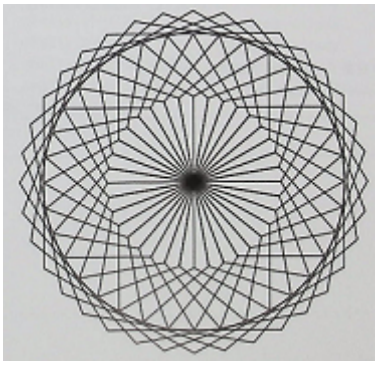
```
# 虛擬碼
以下重複六次
    前進100步
    左轉60度
```

```
## 畫出幾何圖案(一):六角型
import turtle
shelly = turtle.Turtle()
## 重複6次:前進並轉彎
for i in range(6):
    shelly.forward(100)
    shelly.left(60)
```

Step 2: 用巢狀迴圈重複六角形

用迴圈畫出六角形之後,就能把此六角形放在另一個重複的迴圈裡,畫出一個圓的多個六角形,各個稍微重疊。

- 每個六角形相對於前一個六角形只轉10度。
- 畫一個圓,一共需要360/10=36個六角形



```
# 虛擬碼
以下重複36次
    以下重複6次
        前進100步
        左轉60度
    右轉10度
```

```
## 畫出幾何圖案（二）：重複六角型
import turtle
shelly = turtle.Turtle()
for n in range(36):
## 重複6次：前進並轉彎
    for i in range(6):
        shelly.forward(100)
        shelly.left(60)
    shelly.right(10) #加入轉彎
```

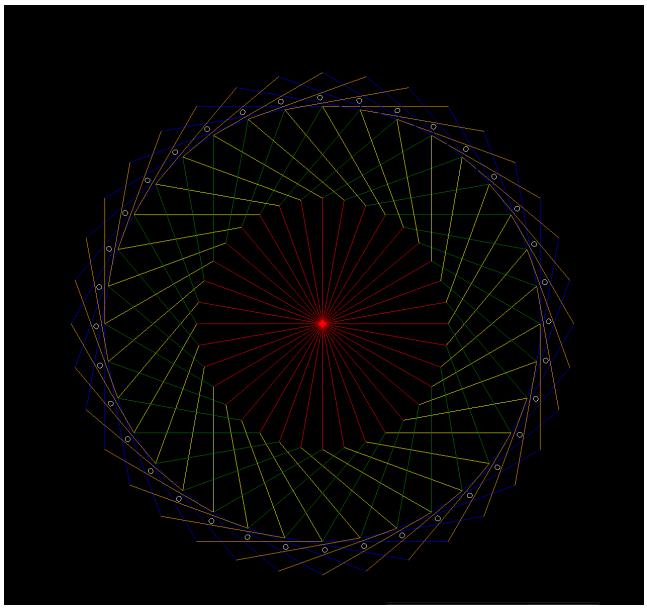
Step 3: 變更背景：加入彩虹顏色

我們可以加入色彩和背景讓這個圖更有趣。

- 背景變黑
  - turtle.bgcolor("black")
- 建立顏色清單，然後用迴圈取顏色。
  - colors = ["red", "yellow", "blue", "orange", "green", "red"]
  - shelly.color(colors[i])

```
## 畫出幾何圖案（三）：彩虹重複六角型
import turtle
shelly = turtle.Turtle()

turtle.bgcolor("black") #把背景變黑

## 畫出36個六角形，各隔10度
for n in range(36):
## 選擇六角形顏色順序
    colors = ["red", "yellow", "blue", "orange", "green",
"purple"]
## 重複6次來畫六角形
    for i in range(6):
```

```
        shelly.color(colors[i]) #選擇i位置的顏色
        shelly.forward(100)
        shelly.left(60)
## 在畫下一個六角形前轉彎
    shelly.right(10)
```



Step 4: 在圖案周圍加上小白圓圈

在海龜圖案的邊緣，畫一個小圓圈。

- 因為有36個六角形，所以會有36個小白圓圈。
- 畫一個小白圓圈，然後退回原點，然後轉10度；再出去畫一個小白圓圈，反覆執行。

```
## 畫出幾何圖案（四）：小圓圈彩虹重複六角型
## 畫36個小圓圈
shelly.penup()
shelly.color("white")
## 重複36次，找到對應的六角形
for i in range(36):
    shelly.forward(220)
    shelly.pendown()
    shelly.circle(5)
    shelly.penup()
    shelly.backward(220)
    shelly.right(10)
## 隱藏海龜
shelly.hideturtle()
```

5. 函數

5-1. 什麼是函式

1. **函式或函數(Functions)是可以重複使用的程式碼區塊，讓我們執行相同的功能並回傳結果。**

當你用電腦解決問題的時候，你會發現有時你會反覆地書寫同樣的程式碼很多次。這個時候，你應該把相同功能的程式定義成一個函數，只要呼叫函數就可以重複使用，不用一再撰寫相同的程式碼。

範例13: 印出多個星號(20, 30, 與50個星號)

每印一次星星，無論幾個，同樣的程式就要重寫一次。

```
for i in range(1,21):
    print("*", end = "")
print()

for i in range(1,31):
    print("*", end = "")
print()

for i in range(1,51):
    print("*", end = "")
print()
```

2. 函數(function)就是將一些語句集合在一起的程式結構，結構化的程式設計有幾個好處：

- 只要寫好函數，做相同動作時，可以利用此函數執行，就不用再寫一次程式了。
- 若要修改某些功能，只要修改這個函數即可。
- 其他程式設計師知道這個函式的功能，什麼輸入有怎樣的輸出，就可以使用這個函數，不需要知道函式實作處理的細節。

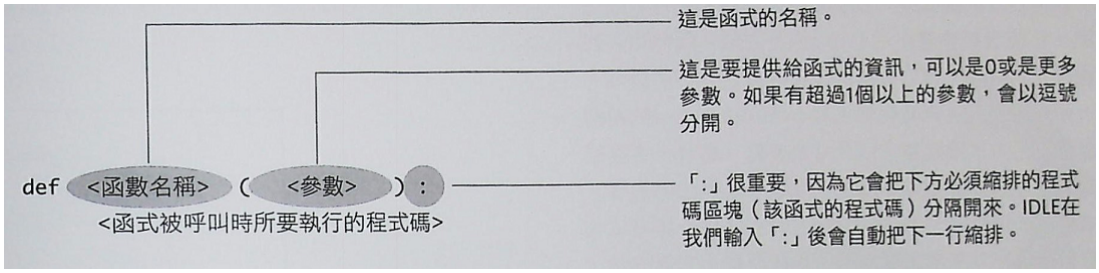
```
## 定義函數 (一)
def printStar(n): # 參數n
    for i in range(1, n + 1):
```

```
        print("*", end = "")
    print()

## 呼叫函數(一):直接呼叫
printStar(20) # 實際參數20
printStar(30) # 實際參數30
printStar(50) # 實際參數40

## 呼叫函數(二):定義函數1(主程式)
def main():
    printStar(20) # 實際參數20
    printStar(30) # 實際參數30
    printStar(50) # 實際參數40

main()

## 呼叫函數(三):定義函數2(數組) (主程式)
def main(times):
    for i in times:
        printStar(i)

number = (20,30,50)
main(number)
```

5- 2. 定義函式與呼叫函數

1. **使用函數**:

- **首先要定義函式**,它是解決某一問題的片段程式。函式的定義先以def開頭,接著函式名稱和小括號及其參數,最後是冒號。
- **再來是呼叫函式**。呼叫函數包括函數名和參數,在函數名後面通常會有一對小括號,這對小括號是用來傳遞參數的,參數可以是具體的值,也可以是變數。傳遞參數給定義的函數後,經過函數體代碼的運算處理,可以得到相應的返回值。



- 要定義一個函式，首先需要描述它的名字及功能。
    - 先從def關鍵字開始，它會向電腦發出信號表示我們正在編寫函式。def是Define（定義）的縮寫，我們正在定義這函式要做什麼。
    - 中間有一個空白字元(space) ，否則直譯器會無法區別
    - 接下來，請為函式命名。記得要選擇一個有意義的名字，清楚地描述函式在做什麼。
    - 接著，將小括弧()添加到函式名稱後；我們以後可能會在括弧中增加參數。
    - 最後，冒號(:)顯示以下的縮排程式碼將成為函式的一部份。
- 請記得，我們所定義的函式不會自動執行。每當電腦遇到def定義函式的程式碼時，會自動跳過整個定義函式的程式碼。如果我們要啟動並執行定義的函式，就必需要先呼叫(called)它，要清楚地告訴電腦：請開始執行我們所呼叫的函式。如果我們不呼叫定義的函式，定義函式的程式碼永遠也不會被執行。
- 呼叫函式很簡單，每當程式中有需要使用函數的地方，寫下函式名稱與括號()就可以呼叫了。
    - 函數可以自己定義，也可以使用他人所定義的函數。
    - 他人定義的函數，包括Python內置函數，以及第三方模組。

2. **參數(Parameter)**是我們指派函式執行某些動作的輸入資料。

- 函式可以沒有參數，也可以有一個或多個參數。
- 當我們創立一個使用參數的函式時，這些函式可以讀取我們輸入的資料。

範例14：假設每次使用程式時，我們使用`print("Hello, person!")`來打招呼。

(1) 我們可以把打招呼的動作移到一個函式中。

```
## 方法（一）：定義函數，沒有參數的函數
def greet():
    print("Hello, person!")

## 呼叫函數
greet()
```

(2) 我們可以使用參數，例如寫出被問候者的名字，增加函式的邏輯性。

```
## 方法（二）：定義函數，有一個參數的函數
def greet(name):
    print(f"Hello, {name}!")

## 呼叫函數
greet("Francis")
greet("John")
```

(3)我們可以使用不只一個參數，例如對不同人（認識與不認識的）輸出不同的問候語，增加函式的邏輯性。

```
## 方法（三）：定義函數，有兩個參數的函數
def greet(name, is_new):
    if (is_new):
        print(f"Hello, {name}! Nice to meet you!!")
    else:
        print(f"What's up, {name}! Good to see you
again!!!")

## 呼叫函數
greet("Francis", True)
greet("John", False)
```

=> 在定義函式時，參數還沒有實際的資料，稱為形式參數(formal parameter)，又名**虛參**。當我們呼叫函式，傳入資料做為實際參數(actual parameter)時，所傳入的資料會自動指派給函式的形式參數，並且由函式內的程式存取。**實參又名引數(argument)**。

- 通常呼叫函數時，會依照定義函數時參數的位置順序來傳遞參數，因此要求呼叫函數時傳入的參數，必須和定義函數時的參數位置一一對應。像這種依照位置順序傳遞參數值的方式，稱為**位置參數**。

    - 位置參數的優點:用法簡單，直接案函數定義參數的順序一一為其賦值即可。
    - 位置參數的缺點:不夠靈活，必須知道每個參數的類型及含義，不能不賦值、更不能不按設定類型賦值。
- 如果呼叫函數時，直接使用參數名稱來指定參數值，而不依照參數的順序來指定，則稱為指名參數，或是**關鍵字參數**。
    - 參數名 = 值
    - 關鍵參數通常有一個默認值，如果調用函數時未設置抹個關鍵字參數，則將使用該默認值。
        - 例如print()函數的分隔符號，預設為空格，即sep = " "
        - 例如print()函數輸出完成後，預設為換行，即end = "\n"。
- **可變參數**，指的是函數中傳遞的參數數目無法確定，也就是可以輸入任意多個參數。
    - 例如求算一組數值的最大值，這組數值的個數可以是任意多個。
    - 可變參數的表示，一般寫成某個參數後面帶著三個點，`...`，代表前面的那個參數可以輸入任意多個值。
- 混搭時，要注意下面原則：
    - 位置參數要置於關鍵字參數的前面，這樣才可以計算位置，否則會產生語法錯誤。
    - 可變參數前面如果有參數，只能是位置參數
    - 可變參數後面如果有參數，只能是關鍵字參數
    - 關鍵字參數前面如果有參數，只能是位置參數或者可變參數
    - 關鍵字參數後面如果有參數，只能是關鍵字參數

=> 函式內參數的對應，若未指定名稱則會依照位置順序填入。

3. **回傳值(Return Values)**是函式被呼叫使用後，輸出執行的結果，也就是回傳資料給我們。

- 通常，定義的函式具有計算功能時，除了需要給參數輸入資料之外，還需要輸出計算的結果。

表 6-1　函式的定義

分類	函式的定義語法	範例
不回傳值的函式	def 函式名稱 (參數 1 , 參數 2 , …): 　函式的敘述區塊	def hi(): 　print('hi')

回傳值的函式	def 函式名稱 (參數 1 , 參數 2 , …):
	　函式的敘述區塊
	　return 要傳回的變數或值
		def min(a,b):
		　if a > b:
		　　return b
		　else:
		　　return a

當函式需要傳回值使用指令return，表示函式回傳資料給原呼叫函式，若不需要回傳值的函式就不需要加上return，函式的定義與傳回值格式。

方法一：不回傳值函式的呼叫語法

函式名稱 (參數值 1, 參數值 2,…)

方法二：回傳值函式的呼叫語法

變數 = 函式名稱 (參數值 1, 參數值 2,…)

(1)沒有傳回值的函式呼叫

```
函式名稱(引數1, 引數2,.....)
```

沒有傳回值的函式呼叫，只是將呼叫敘述端的實引數傳給被呼叫函數端的**虛引數**。接著，虛引數將所得到的資料帶入函式內，經過運算處理後，直接將結果在函式內輸出顯示，不傳回到原來的呼叫敘述。

在函式內，有時你只想要執行某項功能，例如列印特定資訊，但無需傳回任何結果。Python允許函式不使用return來傳回值，此時Python會傳回None，其型別為NonType。

```
# 自訂函式hi，印出「hi」到螢幕上。
## 方法(一)：定義函數，沒有回傳值的函數
def hi():
    print("hi")
## 呼叫函數
hi()
```

(2)有傳回值的函式呼叫

等號右邊要先做完，利用函式名稱與參數來呼叫函式，最後函式回傳值給變數，變數就紀錄函式呼叫後的回傳值。

```
變數1,變數2,..... = 函式名稱（引數1,引數2,.....）
```

- 使用指定運算子「 = 」，會將等號右邊執行的結果指定給等號左邊的變數。函式可傳回一個或多個結果。
- 引數:實引數是傳遞給所呼叫函式使用的資料。實引數 的數量、資料型別，必須和被呼叫函式的虛引數一致。引數若是兩個以上，引數間使用逗號「，」隔開，實引數可以為變數、串列元素、串列，也可以是常數值或運算式。引數列若為0個時，（ ）仍需要保留不可省略。
- 電腦將實引數傳出給被呼叫函式（函式主體）的虛引數，傳入的虛引數將所得到的資料帶入函式主體內，經過運算處理後，在被呼叫函式的最後透過return敘述傳回一個或多個結果資料給呼叫敘述端的變數。被呼叫函式端的傳回值數量、資料型別，必須和呼叫敘述端的變數一致。

範例15:極小值

```
# 自訂函式min，輸入兩個參數a 與b，將較小的數字回傳回來。
## 方法（二）：定義函數，有回傳值的函數
## 自訂函式min，輸入兩個參數a 與b，將較小的數字回傳回來。
def min(a,b):
    if a > b:
        return b
    else:
        return a

## 呼叫函數
## 呼叫函式min，輸入數字2 與4，回傳較小的數字2，並使用函式print印出。
print(min(2,4))
```

範例16：計算長方形面積

使用者輸入長度與寬度，呼叫自訂的函式area 將長度與寬度傳入，回傳計算結果。

```
## 方法（一）有參數、無回傳值
def area(h, w): # 計算長方形的面積，高度h與寬度w是參數
```

```
    print("方形面積 =", h * w)
area(5, 6) # 呼叫area函數，將數值5傳給參數h，將數值6傳給參數w。

## 方法(二)有參數、有回傳值
def area(h, w):
    return h*w
print("方形面積 =", area(5,6))

## 方法(三)有參數、有回傳值，用變數指派
def area(h,w): # 函式area的定義，輸入參數h與w，回傳h乘以w的結果。
    return h*w
a = int(input('請輸入長度？')) # 經由input()函式輸入長度
b = int(input('請輸入寬度？')) # 經由input()函式輸入寬度
answer = area(a, b) # 呼叫area函式，使用a與b參數，將函式回傳值儲存入變數answer。
print('長方形面積為', answer) # 使用print()函式顯示「長方形面積」並串接答案
```

隨堂練習6：計算1到100的總和

請用(1)沒有參數、沒有回傳值 (2)沒有參數、有回傳值 (3)有參數、有回傳值

```
## 定義total函式: 沒有參數、沒有回傳值
def total():
    sum = 0
    for i in range(1,101):
        sum += i
    print("summation of 1 to 100:", sum) # 列印計算總和結果

def main():
    total()

main()

## 定義total函式: 沒有參數、有回傳值
def total():
    sum = 0
    for i in range(1, 101):
        sum += i
    return sum  # 有回傳值，用return敘述
```

```
def main():
    t = total()  # 以變數t接收total函數的回傳值
    print("summation of 1 to 100: ", t) # 列印回傳值

main()

## 定義total函式: 有參數、有回傳值
## 使用者輸入兩個數值區間的總和。
def total(a,b): # 接收兩個形式參數a,b
    sum = 0
    for i in range(a, b + 1):
        sum += i
    return sum  # 總和以return回傳

def main():
    x = eval(input("Enter start number: "))
    y = eval(input("Enter end number: "))
    t = total(x,y)  # 以變數t接收total函數的回傳值
    print("Summation of %d to %s: %d" %(x, y, t))

main()
```

Summary

~ Python 的函數有兩種，分別是內建函數與自訂函數。

1. 系統本身所提供的內建函式，您可以直接直接呼叫使用。

- 之前就已經用過許多內建函數了，例如：str( )、int( )、float( )、len( )、max( )、min( )、print()與input() 等。
- 在呼叫內建函數時，函數名稱後面要緊跟著小括號，中間不留空白字元。

2. 使用者自訂（user-defined）的函式包含兩個部分，分別是「函式的定義」與「函式的呼叫」。

- 「函式的定義」是實作函式的功能，輸入參數與回傳處理後的結果。
    - 自訂函數的命名方式（Function_name）同變數，不可用數字當字首。

    - 在定義函數的小括號中，Pra = parameters 。
        - 參數可以沒有，也可以很多個。
        - 沒有參數傳遞時，小括號中是空的，但小括號不可省略。
- 「函式的呼叫」是從其他程式中呼叫自訂函式，然後執行自訂函式。
    - 在呼叫函數的小括號中，Arg = argument。
        - 呼叫函式的引數和定義函數的參數，兩者的數量是一致的，但名稱可以不同。
        - 當呼叫函數時，Arg1 就將值傳給對應的Pra1、Arg2 就將值傳給對應的 Pra2，依此類推。
- 自訂函數一定要先定義方可呼叫使用，所以定義是放在主程式碼的前面，而呼叫是在後面。

5-3. 函數的輸入與輸出

```
def 函式名 （參數1, 參數2）：
  return 回傳答案
```

使用語法分成兩部分，(1)傳入資料（參數）：定義一個名字來接傳入的資料；(2)回傳答案(回傳值):使用return來將答案回傳，一執行到return就會結束。

(1)函數的輸入：參數

在定義函式時，參數還沒有實際的資料，稱為形式參數(formal parameter)，又名虛參。
當我們呼叫函式，並傳入資料做為實際參數(actual parameter)時，所傳入的資料會自動指派給函式的形式參數，並可由函式內的程式所存取。實際參數又名引數(argument)。

- 在呼叫函式時的傳遞參數方式，一般是依照參數定義的順序來傳遞，像這種依照位置順序傳遞參數值的方式，稱為位置參數。
- 如果直接使用參數名稱來指定參數值，不依照參數順序的指定方式，稱為指名參數，或是1關鍵字參數。
- 混搭時，位置參數要置於指名參數的前面，這樣才可以計算位置，否則會有語法錯誤訊息。

~ 函式內參數的對應，若未指定名稱會依照位置順序填入。

範例17：函數有參數

```
## 定義函數：有x, y, z三個參數，z為預設參數放後面。
def func(x, y, z=9):
    print("x=" , x , "y=" , y , "z=" , z)

## 呼叫函式func
func(1, 2)       # 依照順序放入對應的參數
func(1, 2, 3)    # 依照順序放入對應參數，預設值被輸入值取代
func(x=3, y=4)   # 使用指定方式設定參數，不用依照順序
func(y=5, x=6)   # 使用指定方式設定參數，輸入順序與參數的對應順序可以不同
func(x=3, z=6) # 發生TypeError，沒有預設值的參數y一定要有輸入值
```

~ 函式有許多參數時，預設參數會放在後面。

- 呼叫函式時，若輸入新的參數數則取代原來的預設值。
- 又，預設值是不可變的常數，所以不能是串列或字典等可以修改的資料結構。

```
## 第一種
def f(s, count=1):      # 定義函式f，輸入參數有s 與count，參數count 預設值為1
    print(s * count)    # 使用print 印出count 個字串s到螢幕

f('Hi')                 # 呼叫函數

## 第二種
def f(s, count=2):
    print(s * count)

f('Hi')

## 第三種
def f(s, count=1):
    print(s * count)

f('Hi',3)
```

```
## 第四種
def f(s, count=2):
    print(s * count)

f('Hi',4)
```

範例18：將某一函式所計算的總和與平均數回傳。

```
## 只有一個預設參數n2
def sumAndAverage(n1, n2=100): # 形式參數n1、n2
    total = 0
    average = 0.0
    for i in range(n1, n2 + 1):
        total += i
    average = total / (n2 - n1 + 1)
    return total, average

def main():
    s, a = sumAndAverage(1) # 輸入一個實際參數1傳給形式參數n1，而形式參數n2用預設值
    print(f"sum = {s}, aveage = {a}")
    s, a = sumAndAverage(1, 10) # 輸入兩個實際參數，不用預設
    print(f"sum = {s}, aveage = {a}")
main()
```

```
## 把所有形式參數都設為預設參數
def sumAndAverage(n1=1, n2=100):
    total = 0
    average = 0.0
    for i in range(n1, n2 + 1):
        total += i
    average = total / (n2 - n1 + 1)
    return total, average

def main():
    s, a = sumAndAverage()
    print("sum = %d, average = %d" %(s,a))
```

```
    s, a = sumAndAverage(2)
    print("sum = %d, average = %d" %(s,a))
    s, a = sumAndAverage(1,10)
    print("sum = %d, average = %d" %(s,a))

main()
```

(2)函式的輸出 - 回傳值

程式經由函式呼叫，將資料傳入定義函式，定義函式執行完後傳回給呼叫程式，將結果給後續的程式使用，這時候需要用到回傳值的概念，Python用return關鍵字處理。

表 6-1　函式的定義

分類	函式的定義語法	範例
不回傳值的函式	def 函式名稱 (參數 1 , 參數 2 , …): 　函式的敘述區塊	def hi(): 　print('hi')

回傳值的函式	def 函式名稱 (參數 1 , 參數 2 , …):
	　函式的敘述區塊
	　return 要傳回的變數或值
		def min(a,b):
		　if a > b:
		　　return b
		　else:
		　　return a

~基本上，函數的回傳值是一個，但Python允許多個回傳值。

- 如果函數有多個回傳值時，可以使用tuple回傳多個資料。
- 回傳的tuple 可以使用tuple 開箱(tuple uppacking) 取得回傳的多個參數。

範例19：將某一函式所計算的總和與平均兩個數回傳。

```
##  定義函數：計算總和與平均
def sumAndAverage(n1, n2): # 形式參數n1, n2
    total = 0
    average = 0.0
    for i in range(n1, n2 + 1):
        total += i
    average = total / (n2-n1+1)
    return total, average # 回傳總和與平均數
```

```
def main():
    x, y = eval(input("Enter start and end number: ")) #
輸入計算區間
    s, a = sumAndAverage(x,y) # 呼叫sumAndAverage函數，x,y為
參數，回傳值為s,a
    print("sum = %d, average = %d" %(s,a))

main()
```

5-4. 函式的變數

(1) 變數的作用範圍

變數作用範圍分成全域變數與函式內的區域變數。

- 定義在函式外面的變數稱為**全域變數（global variable）**。
- 定義在函式內部的變數稱為**區域變數（local variable）**。

~函式會先使用區域變數，若沒有區域變數才會使用全域變數。

範例20:全域與區域變數

```
##全域變數g
g = 5

## 定義函式（一）：內無區域變數g，會往函式外去找尋
def f1():
    print(g) # 列印全域變數g的值
## 呼叫函數
f1()
```

~若函式內有一個與全域變數名稱一樣的區域變數，在初始化區域變數之前，如果讀取同名變數，則全域變數和區域變數會打架，產生 UnboundLocalError 錯誤。

```
##全域變數g
g = 5

## 定義函式（二）：有區域變數g，但在被賦值前就呼叫
def f2():
    print(g)
    g = 10
## 呼叫函數
f2()
```

全域變數g 與區域變數g，是兩個不同的變數。

- 函式內區域變數g作用範圍在函式內，全域變數g 作用範圍為整個檔案。
- 當函式內區域變數與全域變數有相同的變數名稱時，函式會優先使用區域變數，若找不到區域變數，才會去找全域變數。

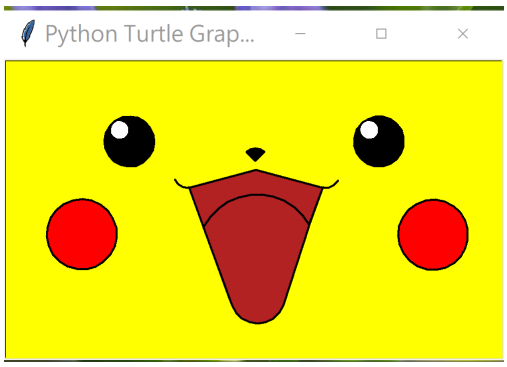
```
## 宣告全域變數g，初始化為5。
g = 5

## 定義函式（三）：宣告區域變數g，並初始化為10，印出區域變數g的值
def f2():
    g = 10 # 區域變數
    print(g)
## 呼叫函數
f2()

## 印出全域變數g
print(g)
```

～函式內如果沒有要找的區域變數，就會往函式外，從全域變數中去尋找。因此，在函數內可以使用global，讓該區域變數明確指向全域變數。也就是，宣告為global的區域變數會指向相同名稱的全域變數。

```
## 宣告全域變數g，初始化為5
g = 5

## 定義函式（四）：宣告g為全域變數並列印出g值
def f():
    global g
    print(g)
```

```
    g = 10
    print(g)
## 呼叫數
f()

## 列印全域變數g的值
print(g)
```

學習愉快 - 烏龜皮卡秋




```
## 皮卡秋 - 函數function

import turtle as t
def nose():
    t.penup()
    t.seth(90)
    t.fd(100)
    t.pendown()
    t.begin_fill()
    t.fillcolor("black")
    t.seth(45)
    t.fd(25)
    t.seth(135)
    t.circle(25,90)
    t.seth(315)
    t.fd(25)
    t.end_fill()

def eyes(seth,fd,c):
    t.penup()
    t.seth(seth)
    t.fd(fd)
    t.pendown()
```

```
    t.begin_fill()
    t.fillcolor('black')
    t.circle(50)
    t.end_fill()
    t.penup()
    t.circle(50,c)
    t.pendown()
    t.begin_fill()
    t.fillcolor('white')
    t.circle(20)
    t.end_fill()

def face(seth,fd):
    t.penup()
    t.seth(seth)
    t.fd(fd)
    t.pendown()
    t.begin_fill()
    t.fillcolor('red')
    t.circle(70)
    t.end_fill()

def lip():
    t.penup()
    t.seth(135)
    t.fd(250)
    t.pendown()
    t.seth(-300)
    t.circle(30,-65)
    t.begin_fill()
    t.fillcolor('Firebrick')
    t.seth(165)
    t.fd(140)
    t.seth(195)
    t.fd(140)
    t.seth(-360)
    t.circle(30,-65)
    t.penup()
    t.seth(-60)
    t.circle(30,65)
    t.pendown()
```

```
    t.seth(-70)
    t.fd(240)
    t.circle(55,140)
    t.seth(70)
    t.fd(240)
    t.end_fill()
    t.seth(-110)
    t.fd(80)
    t.begin_fill()
    t.fillcolor('Firebrick')
    t.seth(120)
    t.circle(120,123)
    t.seth(-70)
    t.fd(165)
    t.circle(55,140)
    t.seth(72)
    t.fd(165)
    t.end_fill()

def setting():
    t.pensize(4)
    t.hideturtle()
    t.setup(1000,600)
    t.speed(10)
    t.screensize(bg='yellow')

def main():
    setting()
    nose()
    eyes(160,250,60)
    eyes(-9.5,530,230)
    face(195,600)
    face(-11,720)
    lip()
    t.done()

if __name__ == '__main__':
    main()
```

```
# 烏龜鐘錶
import turtle
from datetime import *

# 擡起畫筆，向前運動一段距離放下
def Skip(step):
    turtle.penup()
    turtle.forward(step)
    turtle.pendown()

def mkHand(name, length):
    # 註冊Turtle形狀，建立錶針Turtle
    turtle.reset()
    Skip(-length * 0.1)
    # 開始記錄多邊形的頂點。當前的烏龜位置是多邊形的第一個頂點。
    turtle.begin_poly()
    turtle.forward(length * 1.1)
    # 停止記錄多邊形的頂點。當前的烏龜位置是多邊形的最後一個頂點。將
與第一個頂點相連。
    turtle.end_poly()
    # 返回最後記錄的多邊形。
    handForm = turtle.get_poly()
    turtle.register_shape(name, handForm)

def Init():
    global secHand, minHand, hurHand, printer
    # 重置Turtle指向北
    turtle.mode("logo")
    # 建立三個錶針Turtle並初始化
    mkHand("secHand", 135)
    mkHand("minHand", 125)
    mkHand("hurHand", 90)
    secHand = turtle.Turtle()
    secHand.shape("secHand")
    minHand = turtle.Turtle()
    minHand.shape("minHand")
    hurHand = turtle.Turtle()
    hurHand.shape("hurHand")

    for hand in secHand, minHand, hurHand:
        hand.shapesize(1, 1, 3)
```

```
        hand.speed(0)

    # 建立輸出文字Turtle
    printer = turtle.Turtle()
    # 隱藏畫筆的turtle形狀
    printer.hideturtle()
    printer.penup()

def SetupClock(radius):
    # 建立表的外框
    turtle.reset()
    turtle.pensize(7)
    for i in range(60):
        Skip(radius)
        if i % 5 == 0:
            turtle.forward(20)
            Skip(-radius - 20)

            Skip(radius + 20)
            if i == 0:
                turtle.write(int(12), align="center",
font=("Courier", 14, "bold"))
            elif i == 30:
                Skip(25)
                turtle.write(int(i/5), align="center",
font=("Courier", 14, "bold"))
                Skip(-25)
            elif (i == 25 or i == 35):
                Skip(20)
                turtle.write(int(i/5), align="center",
font=("Courier", 14, "bold"))
                Skip(-20)
            else:
                turtle.write(int(i/5), align="center",
font=("Courier", 14, "bold"))
            Skip(-radius - 20)
        else:
            turtle.dot(5)
            Skip(-radius)
        turtle.right(6)
```

```
def Week(t):
    week = ["星期一", "星期二", "星期三",
            "星期四", "星期五", "星期六", "星期日"]
    return week[t.weekday()]

def Date(t):
    y = t.year
    m = t.month
    d = t.day
    return "%s %d%d" % (y, m, d)

def Tick():
    # 繪製錶針的動態顯示
    t = datetime.today()
    second = t.second + t.microsecond * 0.000001
    minute = t.minute + second / 60.0
    hour = t.hour + minute / 60.0
    secHand.setheading(6 * second)
    minHand.setheading(6 * minute)
    hurHand.setheading(30 * hour)

    turtle.tracer(False)
    printer.forward(65)
    printer.write(Week(t), align="center",
                  font=("Courier", 14, "bold"))
    printer.back(130)
    printer.write(Date(t), align="center",
                  font=("Courier", 14, "bold"))
    printer.home()
    turtle.tracer(True)

    # 100ms後繼續呼叫tick
    turtle.ontimer(Tick, 100)

def main():
    # 開啟/關閉龜動畫，併為更新圖紙設定延遲。
    turtle.tracer(False)
    Init()
    SetupClock(160)
    turtle.tracer(True)
    Tick()
```

```
    turtle.mainloop()

if __name__ == "__main__":
    main()
```