

# 第七章 “Batteries included”

我們已經見過Python的基本語法以及許多有用的*built-ins*，但就好像裝好電池的數位相機一樣，我們希望拿到手上就立即能捕捉許多有趣、新鮮的畫面，相機之於攝影者是工具，程式語言的Python亦如斯。

“Batteries included”是Python語言的設計哲學之一，Python的標準安裝直接包含了許多有用的模組，**標準模組庫**提供各式各樣的應用，大體上，絕大多數的工作都能藉由引入標準模組庫的模組來完成。

有哪些模組呢？包括數學相關工具、網路及其協定相關工具、標記語言相關工具、檔案及目錄相關工具、壓縮相關工具、資料處理相關工具、作業系統相關工具、多媒體相關工具、圖形使用者介面相關工具、程式發展相關工具.....等等。

很多呢！不是嗎？引入的方式就跟自行定義的套件一樣，唯一不同的，標準模組庫所存放的位置在Python的安裝路徑內，所以直譯器自己就知道位置，我們只需引入就可。現在我們以常見的帳號密碼輸入做一個例子。

```
#-*- coding: UTF-8 -*-

from getpass import getpass

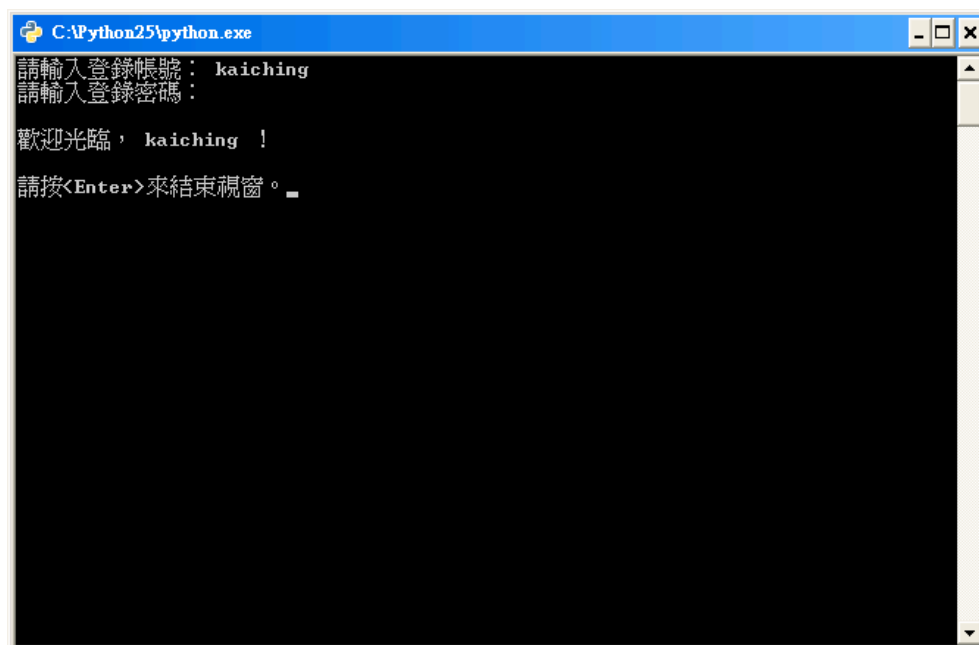
data = {"kaiching": "0000"}

def hello(name):
    print
    print "歡迎光臨", name, "!"

if __name__ == "__main__":
    name = raw_input("請輸入登錄帳號： ")
    word = getpass("請輸入登錄密碼： ")
    if data.has_key(name):
        if word == data[name]:
            hello(name)
        else:
            print "密碼錯誤！"
    else:
        print "註冊後才能登錄！"
    print
    raw_input("請按<Enter>來結束視窗。")
```

這個程式首先引入標準模組庫getpass模組中的getpass函數，這個函數會隱藏使用者輸入密碼時的顯示，然後將其輸入回傳為字串格式。我們用字典型態的變數data儲存已註冊的使用者資料，如果使用者輸入的帳號密碼都符合，便以hello函數顯示歡迎詞。

若是輸入的帳號錯誤，就顯示「註冊後才能登錄！」，若是帳號正確而密碼錯誤，就顯示「密碼錯誤！」。我們來看看程式順利執行的情況吧！



## math

math模組提供許多有用的數學工具，包括計算log函數、三角函數值等，我們用一個例子來看看吧！

```
import math

def result(number):
    print
    print "大於%.2f，最接近的整數為%d" %(number, int(math.ceil(number)))
    print "小於%.2f，最接近的整數為%d" %(number, int(math.floor(number)))
    print "%.2f以2為底的log數為%.2f" %(number, math.log(number, 2))
    print "%.2f的平方根為%.2f" %(number, math.sqrt(number))
    print "%.2f度可轉換為%.2fπ" %(number, math.radians(number)/math.pi)
    print "sin(%.2f) = %.2f" %(number, math.sin(math.radians(number)))
    print "cos(%.2f) = %.2f" %(number, math.cos(math.radians(number)))
    print "tan(%.2f) = %.2f" %(number, math.tan(math.radians(number)))
    print

if __name__ == "__main__":
    while True:
        try:
            number = float(raw_input("請輸入0到360之間的數字： "))
            if number < 0:
                print "輸入的是負數，稍後會轉換為正數....."
                print
                number = math.fabs(number)
            if number > 360:
                print "絕對值大於360！"
                raise ValueError
            else:
                result(number)
                break
        elif number > 360:
            raise ValueError
        else:
```

```

        result(number)
        break
    except ValueError:
        print "請再輸入一次！"

raw_input("請按<Enter>來結束視窗。")

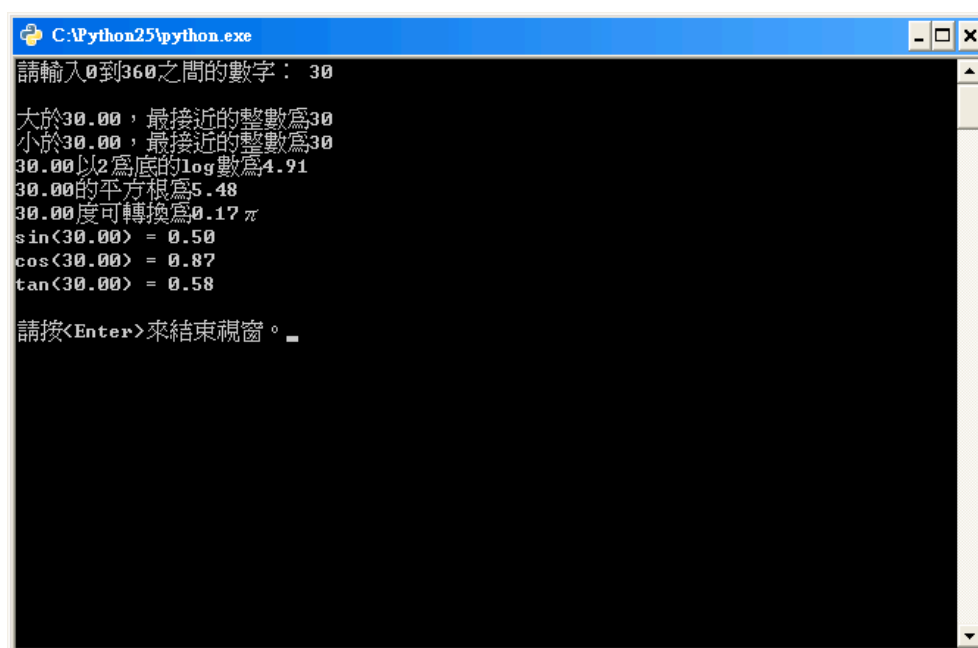
```

result函數用來顯示計算結果，使用者輸入的數字儲存到變數number之中，由上而下依序是計算大於number的最小整數，小於number的最大整數，接者是 $\log_2(\text{number})$ 、number的平方根、轉換number為弧度，然後計算 $\sin(\text{number})$ 、 $\cos(\text{number})$ 與 $\tan(\text{number})$ 。

這裡用了個小技巧，我們限制使用者輸入的數字介於0到360之間，而且避免使用者輸入的不是數字，「while True」迴圈與try...except...陳述會進行到使用者輸入正確為止。而如果使用者輸入的是負數，絕對值小於360，程式仍會接受並利用math模組中的fabs()函數，轉換負數為正數。

當使用者輸入是被允許的0到360之間的數字，或是絕對值小於360的複數，程式就會執行result函數印出計算結果，緊接呼叫完result函數後的break陳述，這是為了跳出「while True」，使程式可以結束。

我們來看看輸入30的計算結果吧！



```

C:\Python25\python.exe
請輸入0到360之間的數字： 30
大於30.00，最接近的整數為30
小於30.00，最接近的整數為30
30.00以2為底的log數為4.91
30.00的平方根為5.48
30.00度可轉換為0.17  $\pi$ 
sin(30.00) = 0.50
cos(30.00) = 0.87
tan(30.00) = 0.58
請按<Enter>來結束視窗。

```

## Note

有關math模組的詳細資訊，詳情可參考Python Library Reference的[math](#)。

## time

time模組提供許多處理時間的工具，我們以印出現在時間為例。

```

#-*- coding: UTF-8 -*-

from time import sleep, asctime

weeks = {"Mon": "星期一", "Tue": "星期二", "Wed": "星期三", "Thu": "星期四", \

```

```
    "Fri": "星期五", "Sat": "星期六", "Sun": "星期日"}
months = {"Jan": "一月", "Feb": "二月", "Mar": "三月", "Apr": "四月", \
          "May": "五月", "Jun": "六月", "Jul": "七月", "Aug": "八月", \
          "Sep": "九月", "Oct": "十月", "Nov": "十一月", "Dec": "十二月"}
moment = asctime()

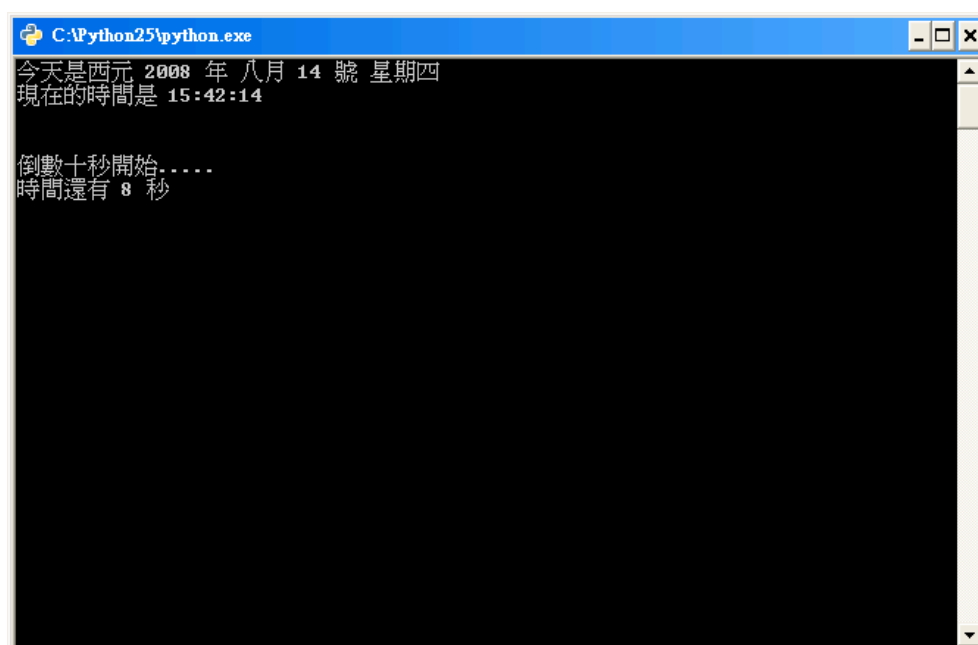
def now():
    print "今天是西元", moment[20:], "年", months[moment[4:7]],
    print moment[8:10], "號", weeks[moment[:3]]
    print "現在的時間是", moment[11:19]
    print

if __name__ == "__main__":
    now()
    print
    print "倒數十秒開始....."
    i = 10
    while i > 0:
        print "時間還有", i, "秒",
        sleep(1)
        print "\r",
        i = i - 1
    print
    print "十秒過後.....你看見消失的時間了嗎？"
    print "現在的時間是", asctime()[11:19]
    print
    raw_input("請按<Enter>來結束視窗。")
```

time模組中的asctime()函數會去抓現在電腦的時間，然後回傳一個字串，格式如下。  
'Thu Aug 14 15:42:14 2008'

Thu是Thursday的縮寫，Thursday則是星期四的英文，同樣的Aug是August的縮寫，而August是八月的縮寫，所以這個字串可看成星期、月、日、時、分、秒、年的排列。我們利用兩個字典型態英文縮寫與中文的相對名稱，然後利用變數moment儲存現在時間的字串。

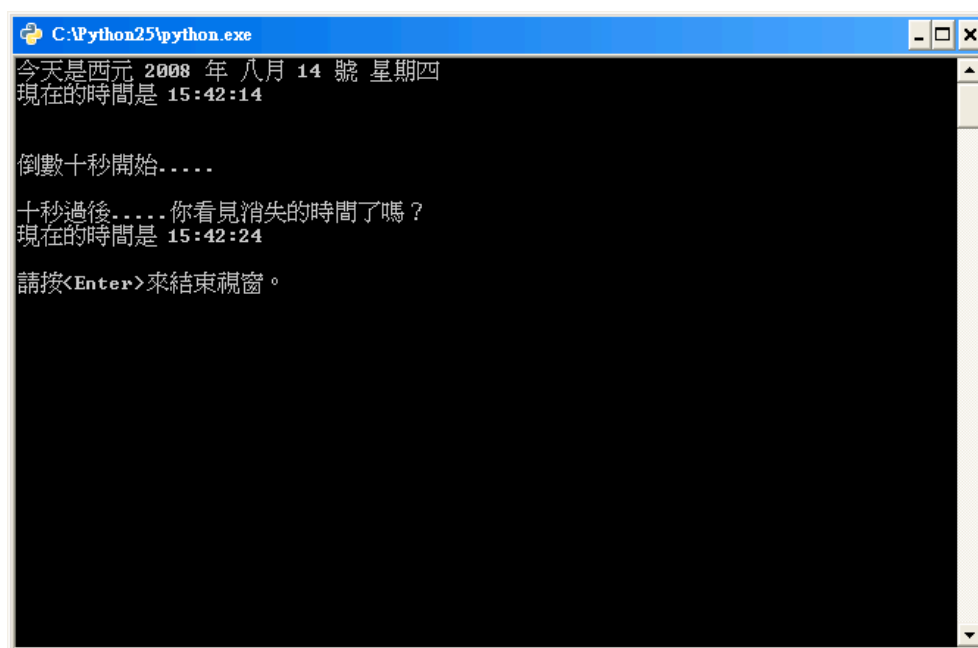
我們自行定義函數now印出現在時間，第一行是中文習慣的年、月、日、星期，第二行則是現在時間。然後利用time模組的另一個sleep()函數執行倒數計時，其參數為數字，單位為秒，在這個程式中每呼叫一次便暫停1秒，效果如下。



```
C:\Python25\python.exe
今天是西元 2008 年 八月 14 號 星期四
現在的時間是 15:42:14

倒數十秒開始.....
時間還有 8 秒
```

最後的執行結果如下。



## Note

有關time模組的詳細資訊，詳情可參考Python Library Reference的[time](#)。

## sys

sys模組提供了許多輔助程式執行的函數與變數，例如變數platform可以偵測作業系統類型，而變數version則可偵測Python版本，函數exit()則可用來結束程式。

我們利用「while True」迴圈與函數exit()寫選單式的程式。

```
#-*- coding: UTF-8 -*-
```

```
from sys import exit, platform, version
```

```
def whatplatform():
    if platform == "win32":
        print "您所使用的是MS-Windows作業系統。"
    elif platform == "linux2":
        print "您所使用的是Linux作業系統。"
    elif platform == "darwin":
        print "您所使用的是Mac作業系統。"
    else:
        print "程式未能偵測到您所使用的作業系統種類。"
```

```
def fun1():
    print "這是「選單一」。"
    raw_input("請按<Enter>繼續。")
```

```
def fun2():
```

```

print "這是「選單二」。"
raw_input("請按<Enter>繼續。")

if __name__ == "__main__":
    whatplatform()
    print "您的Python版本為", version[:5]
    print
    while True:
        print "(1) 選單一....."
        print "(2) 選單二....."
        c = raw_input("請輸入選擇，或按q離開程式： ")
        if c == "1":
            fun1()
        elif c == "2":
            fun2()
        elif c == "q":
            exit()
        else:
            print "您所輸入的不是1或2或q。"

```

程式一開始印出作業系統型態與Python版本，然後進入「while True」迴圈，如果使用者鍵入1，程式跳到函數fun1印出「這是『選單一』。」，鍵入2則是跳到函數fun2印出「這是『選單二』。」，鍵入q則執行函數exit()結束程式，若非鍵入以上三個選項，就會印出「您所輸入的不是1或2或q。」

只有鍵入q執行函數exit()結束程式，「while True」迴圈也隨之結束，不然程式就會一直繼續下去。

```

C:\Python25\python.exe
您所使用的是MS-Windows作業系統。
您的Python版本為 2.5.2
(1) 選單一.....
(2) 選單二.....
請輸入選擇，或按q離開程式： 1
這是「選單一」。
請按<Enter>繼續。
(1) 選單一.....
(2) 選單二.....
請輸入選擇，或按q離開程式： 2
這是「選單二」。
請按<Enter>繼續。
(1) 選單一.....
(2) 選單二.....
請輸入選擇，或按q離開程式： 3
您所輸入的不是1或2或q。
(1) 選單一.....
(2) 選單二.....
請輸入選擇，或按q離開程式：

```

## Note

有關sys模組的詳細資訊，詳情可參考Python Library Reference的[sys](http://docs.python.org/2/library/sys.html)。

## OS

os模組提供了許多作業系統的服務，而且符合程式跨平台的需求。這裡，我們只引入函數system()改寫上面選單的例子。函數system()需要一個參數，其型態為字串，用來執行作業系統的指令。

Windows系統的命令列可用指令cls來清除螢幕，並將游標移到終端機視窗的最上面，在UNIX系統，包括Mac與Linux則是利用指令clear來達到相同目的。因此我們利用sys模組的platform變數做平台檢查，另外寫一個函數clear()，若是平台為Windows則回傳字串“cls”，而平台為Mac與Linux則回傳字串“clear”。

我們直接以clear()作為system()的參數，原本的「while True」迴圈改寫到函數main()之中，然後在main()、fun1()、fun2()開始時都呼叫system(clear())，以達到清除螢幕的效果。

程式碼如下。

```
#-*- coding: UTF-8 -*-

from sys import exit, platform, version
from os import system

def whatplatform():
    if platform == "win32":
        print "您使用的是MS-Windows作業系統。"
    elif platform == "linux2":
        print "您使用的是Linux作業系統。"
    elif platform == "darwin":
        print "您使用的是Mac作業系統。"
    else:
        print "程式未能偵測到您所使用的作業系統種類。"

def clear():
    if platform == "win32":
        return "cls"
    elif platform == "linux2" or "darwin":
        return "clear"
    else:
        return False

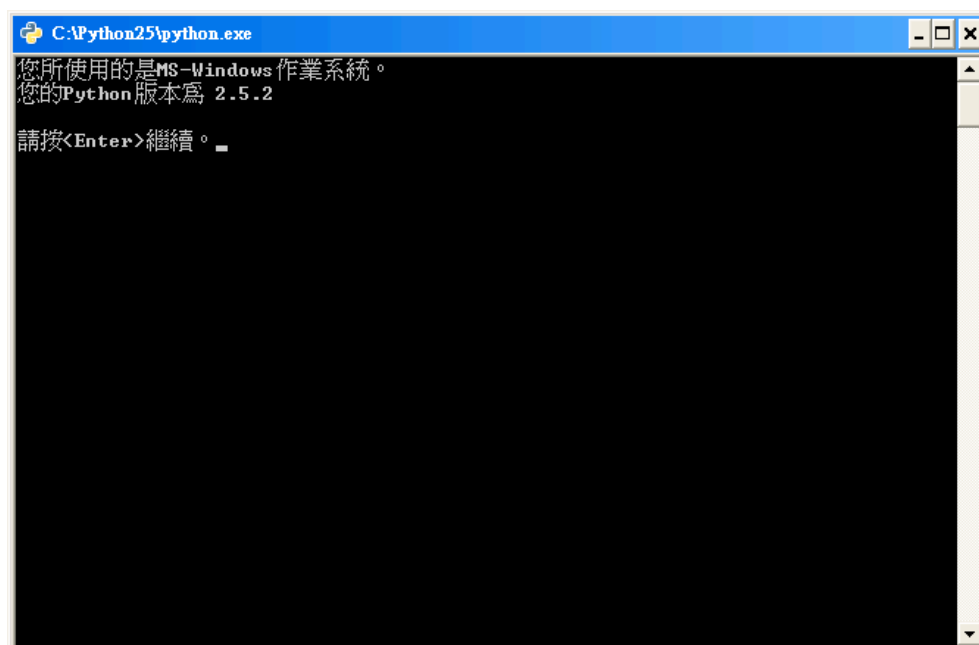
def fun1():
    try:
        system(clear())
        print "這是「選單一」。"
        raw_input("請按<Enter>繼續。")
    except TypeError:
        print "系統並無清除螢幕的指令，程式仍繼續運作....."
        print "這是「選單一」。"
        raw_input("請按<Enter>繼續。")

def fun2():
    try:
        system(clear())
        print "這是「選單二」。"
        raw_input("請按<Enter>繼續。")
    except TypeError:
        print "系統並無清除螢幕的指令，程式仍繼續運作....."
        print "這是「選單二」。"
        raw_input("請按<Enter>繼續。")

def main():
```

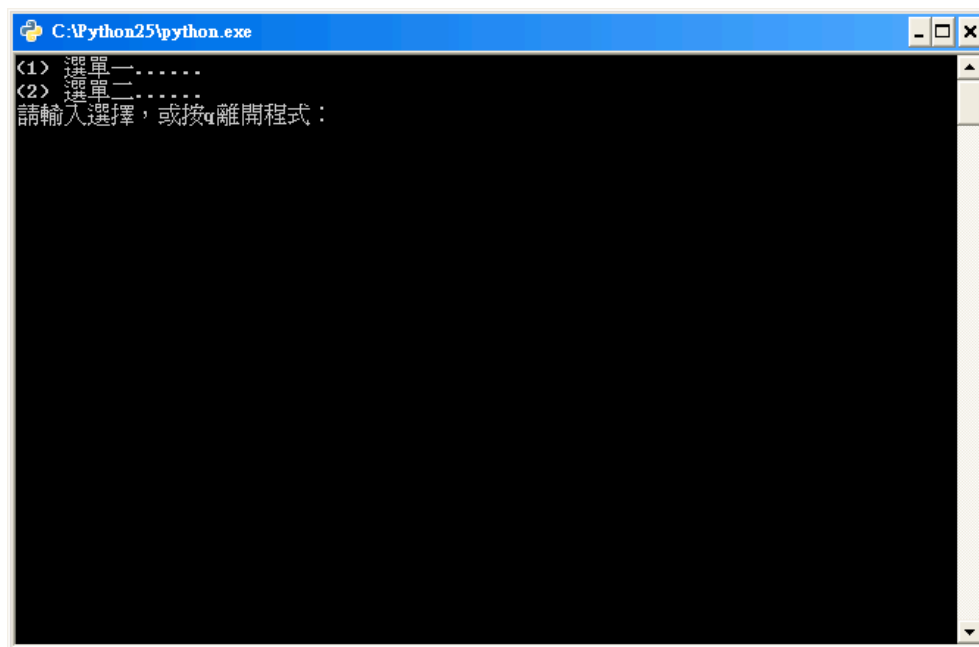
```
while True:
    try:
        system('clear()')
        print "(1) 選單一....."
        print "(2) 選單二....."
        c = raw_input("請輸入選擇，或按q離開程式： ")
        if c == "1":
            fun1()
        elif c == "2":
            fun2()
        elif c == "q":
            exit()
        else:
            print "您所輸入的不是1或2或q。"
            raw_input("請按<Enter>繼續。")
    except TypeError:
        print "系統並無清除螢幕的指令，程式仍繼續運作....."
        print "(1) 選單一....."
        print "(2) 選單二....."
        c = raw_input("請輸入選擇，或按q離開程式： ")
        if c == "1":
            fun1()
        elif c == "2":
            fun2()
        elif c == "q":
            exit()
        else:
            print "您所輸入的不是1或2或q。"
            raw_input("請按<Enter>繼續。")
if __name__ == "__main__":
    whatplatform()
    print "您的Python版本為", version[:5]
    print
    raw_input("請按<Enter>繼續。")
    main()
```

我們來看看程式在Windows執行的結果。

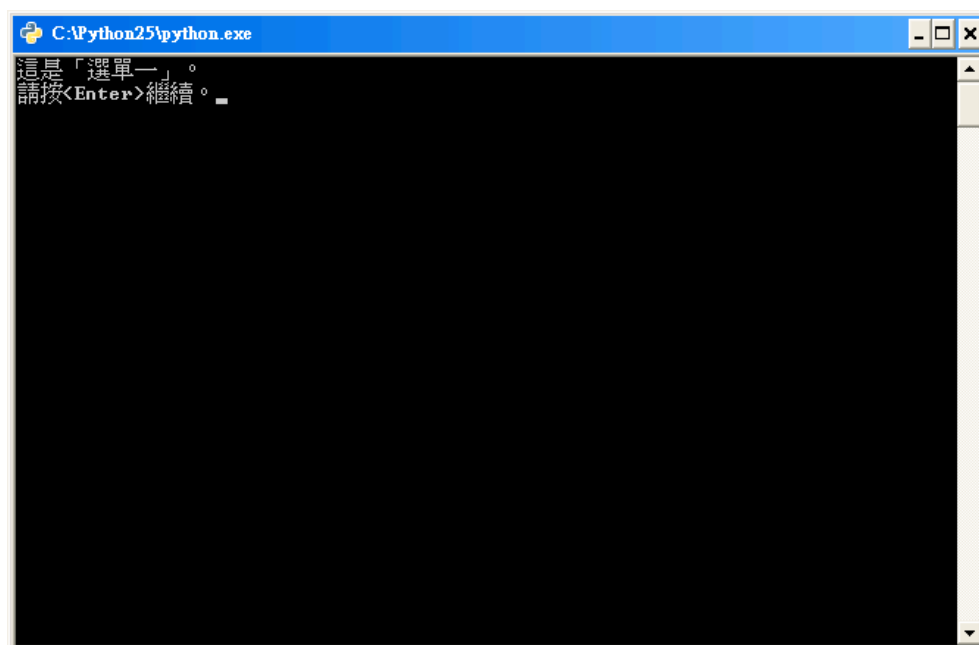




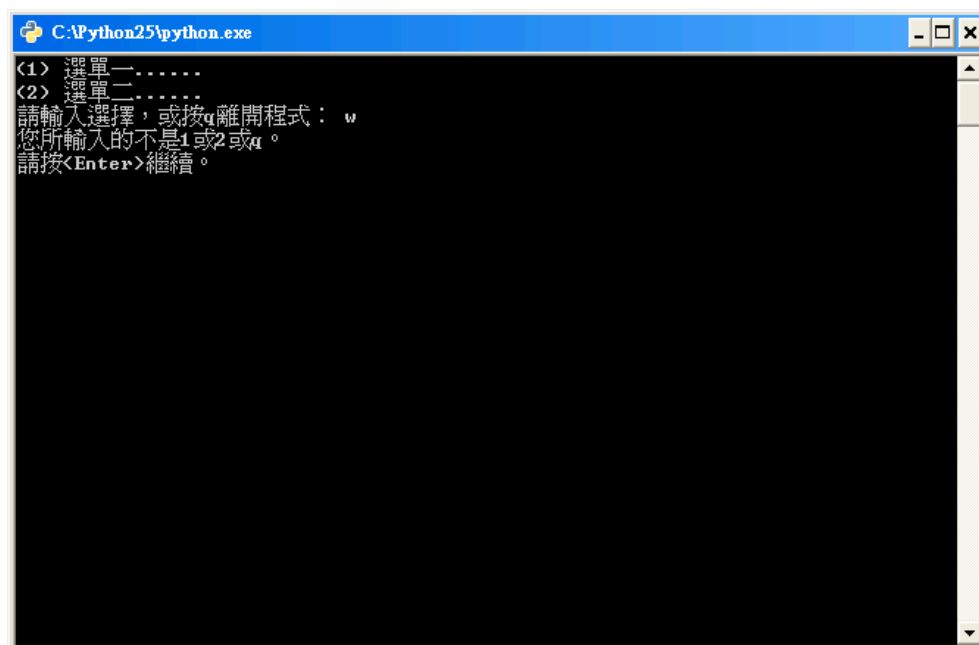
接著按<Enter>。



進入了主選單，接著按入數字「1」，進入選單一。



按下<Enter>，然後回到了主選單。假如我們輸入的是其他的字母呢？



程式只接受數字「1」、「2」或字母「q」的輸入。

## Note

有關os模組的詳細資訊，詳情可參考Python Library Reference的[os](#)。

## random

random模組則提供產生擬隨機數的函數供我們運用，我們引入其中的函數randint()來寫一個猜數字遊戲。

```
#-*- coding: UTF-8 -*-  
  
from random import randint  
  
answer = randint(1,99)  
  
if __name__ == "__main__":  
    print "***猜數字遊戲***"  
    print  
    state = True  
    while state:  
        try:  
            guess = int(raw_input("請輸入1到99的數字： "))  
            if guess == answer:  
                print  
                print "正確答案！"  
                print  
                state = False  
            elif answer < guess < 100:  
                print "太大囉！再試一次。"  
            elif 0 < guess < answer:  
                print "太小囉！再試一次。"  
            else:
```

```

        raise ValueError
    except ValueError:
        print "請不要輸入小於1或大於99的整數，或是除了整數以外的符號。"

```

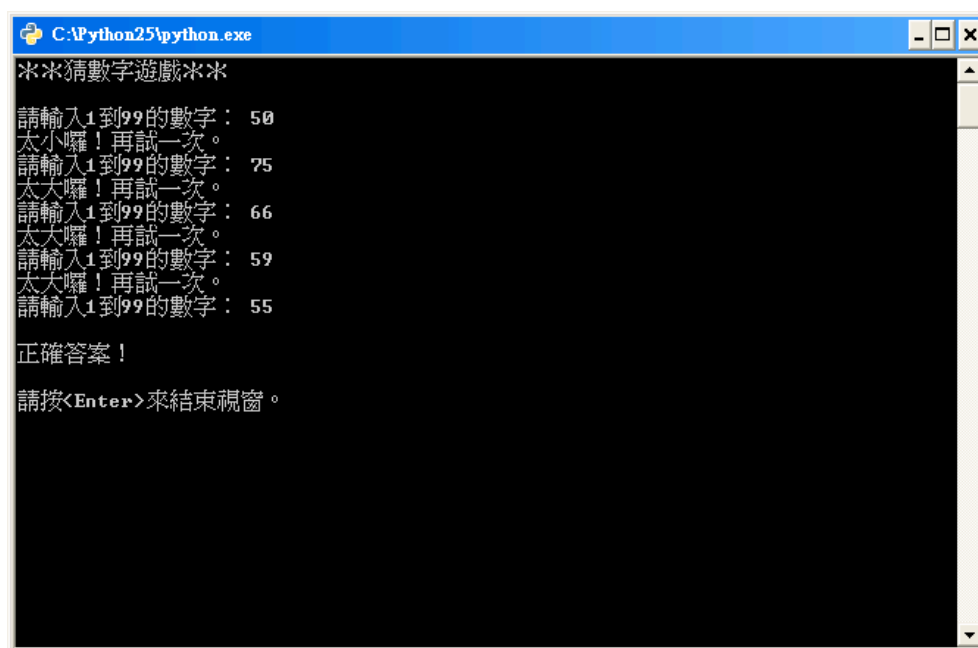
```
raw_input("請按<Enter>來結束視窗。")
```

函數randint()需要兩個整數型態的參數，然後隨機傳回介於之間的一個整數，因此當我們希望被猜的數字介於1到99之間時，就可以將參數分別設為1和99，並且將其存入變數answer之中。

我們仍是利用「while True」迴圈來進行遊戲，變數state控制迴圈是否繼續進行。若是使用者輸入的並非整數，函數int()便會引發ValueError，所以這裡我們仍是用try...except...來處理這項例外。

使用者所猜的數字存入變數guess之中，若是與answer相同，印出「正確答案！」，變數state轉為False，遊戲也隨之結束，若是大於answer且小於100，或小於answer且大於0，分別印出提示訊息。而若是使用者輸入非預期的數字，小於1或大於99，程式自動引發ValueError。

我們來玩玩看吧！



```

C:\Python25\python.exe
**猜數字遊戲**
請輸入1到99的數字： 50
太小囉！再試一次。
請輸入1到99的數字： 75
太大囉！再試一次。
請輸入1到99的數字： 66
太大囉！再試一次。
請輸入1到99的數字： 59
太大囉！再試一次。
請輸入1到99的數字： 55
正確答案！
請按<Enter>來結束視窗。

```

## Note

有關random模組的詳細資訊，詳情可參考Python Library Reference的[random](#)。