Chapter 13
設計與應用模組

13-1: 將自建的函數儲存在模組中

- 在大型計畫中，每個人負責設計的小功能、函數或類別，可以儲存在模組 (module) 中，讓團隊中的其他人使用。
  - 模組又可稱作套件 (package)
- 通常分成 3 大類
  - 內建的模組，例如 os, random, zipeifle, … 等等
  - 外部模組，需使用 pip 安裝。
  - 我們自己程式建立的模組

13-1: 將自建的函數儲存在模組中

- 一個簡單可執行的程式

test_CH13.py > ...

```
1   def makeIceCream(iceCream, *toppings):
2       """列出製作冰淇淋的配料"""
3       print(f"{iceCream} 冰淇淋的配料如下：")
4       for topping in toppings:
5           print(f"---- {topping}")
6
7   def makeDrink(size, drink):
8       # 輸入飲料規格與種類，然後輸出飲料
9       print("點的飲料如下：")
10      print(f"---- {size.title()}")
11      print(f"---- {drink.title()}")
12
13  makeIceCream("草莓", "草莓果醬")
14  makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
15  makeDrink("large", "coke")
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

13-1: 將自建的函數儲存在模組中

- 將前 11 行存成 makefood.py

test.py　makefood.py ●

makefood.py > ...

```
1   def makeIceCream(iceCream, *toppings):
2       """列出製作冰淇淋的配料"""
3       print(f"{iceCream} 冰淇淋的配料如下：")
4       for topping in toppings:
5           print(f"---- {topping}")
6
7   def makeDrink(size, drink):
8       # 輸入飲料規格與種類，然後輸出飲料
9       print("點的飲料如下：")
10      print(f"---- {size.title()}")
11      print(f"---- {drink.title()}")
```

13-2:應用自己建立的函數模組

- 將前 11 行存成 makefood.py，並修改 test.py 的內容。

makefood.py > ...

```
1   def makeIceCream(iceCream, *toppings):
2       """列出製作冰淇淋的配料"""
3       print(f"{iceCream} 冰淇淋的配料如下：")
4       for topping in toppings:
5           print(f"---- {topping}")
6
7   def makeDrink(size, drink):
8       # 輸入飲料規格與種類，然後輸出飲料
9       print("點的飲料如下：")
10      print(f"---- {size.title()}")
11      print(f"---- {drink.title()}")
```

test.py

```
1   import makefood      # 導入模組使用
2
3   makefood.makeIceCream("草莓", "草莓果醬")
4   makefood.makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
5   makefood.makeDrink("large", "coke")
```

模組名稱為檔案名稱

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

13-2:應用自己建立的函數模組

- 導入模組內特定單一函數
  - from 模組 import 函數

```
#從模組導 makefood 入 makeIceCream 使用
from makefood import makeIceCream

makeIceCream("草莓", "草莓果醬")
makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
makeDrink("large", "coke")  #因為沒有導入此函數，因此會產生錯誤。
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
Traceback (most recent call last):
  File "e:\PythonTest\CH13\test.py", line 5, in <module>
    makeDrink("large", "coke")  #因為沒有導入此函數，因此會產生錯誤。
NameError: name 'makeDrink' is not defined
```

13-2:應用自己建立的函數模組

- 導入模組內多個函數
  - from 模組 import 函數1, 函數2, …

```
# 從模組 makefood 導入 makeIceCream 與 makeDrink 使用
from  makefood import makeIceCream, makeDrink

makeIceCream("草莓", "草莓果醬")
makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
makeDrink("large", "coke")
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

- 導入模組內所有函數
  - from 模組 import *

```
# 從模組 makefood 導入所有函數
from  makefood import *

makeIceCream("草莓", "草莓果醬")
makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
makeDrink("large", "coke")
```

13-2:應用自己建立的函數模組

- 使用as給函數指定替代名稱
  - 遇到函數名稱太長或是遇到模組中函數的名稱與自行設計的函數名稱相同，可以給函數一個替代名稱。
  - from 模組 import 函數 as 替代名稱

```
1   # 使用 makeIC 替代 makeIceCream 函數名稱
2   from makefood import makeIceCream as makeIC
3
4   makeIC("草莓", "草莓果醬")
5   makeIC("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
```

13-2:應用自己建立的函數模組

- 使用as給模組指定替代名稱
  - import 模組名稱 as 替代名稱

```
1   # 使用 m 替代 makefood 模組名稱
2   import makefood as m
3
4   m.makeIceCream("草莓", "草莓果醬")
5   m.makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓
6   m.makeDrink("large", "coke")
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

13-2:應用自己建立的函數模組

- 重新載入模組
  - import importlib
  - importlib.reload(module)

CH13 > makefood.py > ...

```
1   ICE = "Few Ice"
2
3   def makeIceCream(iceCream, *toppings):...
8
9   def makeDrink(size, drink, ice = "No Ice"):
10      # 輸入飲料規格與種類，然後輸出飲料
11      print("點的飲料如下：")
12      print(f"---- {size.title()}\
13  /{drink.title()}/{ice.title()}")
```

CH13 > test.py

```
1   import importlib
2   import makefood
3
4   makefood.makeDrink("large", "coke", makefood.ICE)
5
6   makefood.ICE = "Lots of Ice"
7   makefood.makeDrink("large", "coke", makefood.ICE)
8
9   importlib.reload(makefood)
10  makefood.makeDrink("large", "coke", makefood.ICE)
```

```
點的飲料如下：
---- Large/Coke/Few Ice
點的飲料如下：
---- Large/Coke/Lots Of Ice
點的飲料如下：
---- Large/Coke/Few Ice
```

13-2:應用自己建立的函數模組

- 將主程式放在main( )與__name__搭配的好處
  - 為了不希望將第一個範例程式當成模組被引用時，執行了主程式的內容，所以將主程式與函數分別存成兩個檔案，test.py 與 makefood.py。
  - 可以將第一個範例程式的主程式改寫，未來可以直接導入模組，不用再改寫。

13-2:應用自己建立的函數模組

- 將主程式放在main( )與__name__搭配的好處
  - 獨立執行 makefood.py 會去執行 main() 的內容。
  - 被當模組呼叫時，則不會執行 main()

makefood_new.py > ...

```python
def makeIceCream(iceCream, *toppings):
    """列出製作冰淇淋的配料"""
    print(f"{iceCream} 冰淇淋的配料如下：")
    for topping in toppings:
        print(f"---- {topping}")

def makeDrink(size, drink):
    # 輸入飲料規格與種類，然後輸出飲料
    print("點的飲料如下：")
    print(f"---- {size.title()}")
    print(f"---- {drink.title()}")

def main():
    makeIceCream("草莓", "草莓果醬")
    makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
    makeDrink("large", "coke")

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

13-2:應用自己建立的函數模組

- import makefood_new

```python
# 使用 m 替代 makefood_new 模組名稱
import makefood_new as m

m.makeIceCream("草莓", "草莓果醬")
m.makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
m.makeDrink("large", "coke")
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

13-2:應用自己建立的函數模組

- import test_CH13

```
1  #
2  import test_CH13
3
4  test_CH13.makeIceCream("草莓", "草莓果醬")
5  test_CH13.makeIceCream("草莓巧克力", "草莓果醬", "巧克力醬", "新鮮草莓")
6  test_CH13.makeDrink("large", "coke")
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

```
草莓 冰淇淋的配料如下：
---- 草莓果醬
草莓巧克力 冰淇淋的配料如下：
---- 草莓果醬
---- 巧克力醬
---- 新鮮草莓
點的飲料如下：
---- Large
---- Coke
```

13-3:將自建的類別儲存在模組內

test.py > ...

```python
class Banks():
    """定義銀行類別"""
    def __init__(self, userName):
        self.__name = userName
        self.__money = 10000
        self.__bankName = "中國信託商業銀行"
    def saveMoney(self, money):
        self.__money += money
        print(f"存進 {money} 元，完成。")

    def withdrawMoney(self, money):
        self.__money -= money
        print(f"提 {money} 元，完成。")

    def showMoney(self):
        print(f"{self.__name} 目前餘額 {self.__money} 元")

    def showTitle(self):
        return self.__bankName
```

```python
class BanksTainan(Banks):
    def __init__(self, userName):
        self.bankName = "中國信託商業銀行 台南分行"

    def showTitle(self):
        return self.bankName

wuBank = Banks("Wu")
linBank = BanksTainan("Lin")
print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

```
Wu 在 中國信託商業銀行 開了個戶頭
Lin 在 中國信託商業銀行 台南分行 開了個戶頭
```

13-3:將自建的類別儲存在模組內

- 將兩個 class 存成 classBank.py，並加上 __name__ == '__main__' 的判斷。

classBanks.py > ...

```
1   class Banks():
2       """定義銀行類別"""
3   >   def __init__(self, userName):···
7
8   >   def saveMoney(self, money):···
11
12  >   def withdrawMoney(self, money):···
15
16  >   def showMoney(self):···
18
19  >   def showTitle(self):···
21
22  class BanksTainan(Banks):
23  >   def __init__(self, userName):···
25
26  >   def showTitle(self):···
```

```
29  def main():
30      wuBank = Banks("Wu")
31      linBank = BanksTainan("Lin")
32      print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
33      print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
34
35  if __name__ == '__main__':
36      main()
```

13-4:應用自己建立的類別模組

- 導入模組的單一類別
  - 觀念與導入函數一樣。
  - from 模組名稱 import 類別名稱

```
test.py > ...
1   # 導入 classBanks 模組中的 Banks 類別
2   from classBanks import Banks
3
4   wuBank = Banks("Wu")
5   #linBank = BanksTainan("Lin")
6   print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
7   #print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

```
Wu 在 中國信託商業銀行 開了個戶頭
```

13-4:應用自己建立的類別模組

- 導入模組的多個類別：from 模組 import 類別1, 類別2, ...

```python
# 導入 classBanks 模組的 Banks 與 BanksTainan 類別
from classBanks import Banks, BanksTainan

wuBank = Banks("Wu")
linBank = BanksTainan("Lin")
print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

```
Wu 在 中國信託商業銀行 開了個戶頭
Lin 在 中國信託商業銀行 台南分行 開了個戶頭
```

- 導入模組的所有類別：from 模組 import *

```python
# 導入 classBanks 模組的所有類別
from classBanks import *

wuBank = Banks("Wu")
linBank = BanksTainan("Lin")
print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

```
Wu 在 中國信託商業銀行 開了個戶頭
Lin 在 中國信託商業銀行 台南分行 開了個戶頭
```

13-4:應用自己建立的類別模組

- import 模組名稱 (as 別名)

test.py > ...

```python
# 導入 classBanks 模組
import classBanks

wuBank = classBanks.Banks("Wu")
linBank = classBanks.BanksTainan("Lin")
print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

```
Wu 在 中國信託商業銀行 開了個戶頭
Lin 在 中國信託商業銀行 台南分行 開了個戶頭
```

test.py > ...

```python
# 導入 classBanks 模組
import classBanks as cBanks

wuBank = cBanks.Banks("Wu")
linBank = cBanks.BanksTainan("Lin")
print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

13-4:應用自己建立的類別模組

- 模組內導入另一個模組的類別
  - 一個模組內太多 class，此時可考慮將類別拆成 2 或多個模組儲存。
  - 若拆開的模組有衍生/繼承的關係，則子類別也要將父類別導入。

classBanks.py > ...

```
1   class Banks():
2       """定義銀行類別"""
3   >   def __init__(self, userName):···
7
8   >   def saveMoney(self, money):···
11
12  >   def withdrawMoney(self, money):···
15
16  >   def showMoney(self):···
18
19  >   def showTitle(self):···
21
```

classBanksTainan.py > ...

```
1   from classBanks import Banks
2
3   class BanksTainan(Banks):
4   >   def __init__(self, userName):···
6
7   >   def showTitle(self):···
```

test.py > ...

```
1   from classBanks import Banks
2   from classBanksTainan import BanksTainan
3
4   wuBank = Banks("Wu")
5   linBank = BanksTainan("Lin")
6   print(f"Wu 在 {wuBank.showTitle()} 開了個戶頭")
7   print(f"Lin 在 {linBank.showTitle()} 開了個戶頭")
```

13-5:隨機數random模組

- 所謂的隨機數(Random number)是指平均散佈在某區間的數字
- 程式需 import random

函數名稱	說明
randint(x, y)	產生一大於等於 x 且小於等於 y 的隨機整數。
random()	產生一大於等於 0 且小於 1 的隨機浮點數。
uniform(x, y)	產生一大於等於 x 且小於 y 的隨機浮點數。
choice(串列)	隨機回傳串列中的一個元素。
shuffle(串列)	將"串列"元素重新排列。
sample(串列, 數量)	從"串列"中隨機回傳 "數量"個元素
seed(x)	X是種子值，未來每次可以產生相同的亂數。

13-5:隨機數random模組

- randint:
  - 建立一個程式分別產生各3組在1-100、500-1000、2000-3000的數字

```python
import random

n = 3
for i in range(n):
    print("1-100    :", random.randint(1, 100))

for i in range(n):
    print("500-1000 :", random.randint(500, 1000))

for i in range(n):
    print("2000-3000:", random.randint(2000, 3000))
```

```
1-100    : 21
1-100    : 75
1-100    : 45
500-1000 : 827
500-1000 : 910
500-1000 : 806
2000-3000: 2973
2000-3000: 2226
2000-3000: 2428
```

```
1-100    : 39
1-100    : 48
1-100    : 25
500-1000 : 641
500-1000 : 960
500-1000 : 793
2000-3000: 2344
2000-3000: 2823
2000-3000: 2454
```

13-5:隨機數random模組

- randint(): 猜數字
  - 先用randint( )方法產生一個1到10之間的數字
  - 如果猜的數值太小會要求猜大一些，然後如果猜的數值太大會要求猜小一些。

```python
import random

min, max = 1, 10
ans = random.randint(min, max)
while True:
    guessNum = int(input("請猜 1-10 之間的數字："))
    if guessNum == ans:
        print("猜對了～～～")
        break
    elif guessNum < ans:
        print("請猜大一點")
    else:
        print("請猜小一點")
```

```
請猜 1-10 之間的數字：5
猜對了～～～
```

```
請猜 1-10 之間的數字：5
請猜小一點
請猜 1-10 之間的數字：2
請猜大一點
請猜 1-10 之間的數字：4
請猜小一點
請猜 1-10 之間的數字：3
猜對了～～～
```

13-5:隨機數random模組

- randint():猜大小的遊戲
  - 程式執行初可以設定莊家的輸贏比例。
  - 程式執行過程會立即回應是否猜對。

```
1   import random
2
3   min, max = 1, 100
4   winPercent =  int(input("請輸入莊家贏的比例 (1~100)："))
5
6   while True:
7       guessAns = input("請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：")
8       if guessAns == "q" or guessAns == 'Q':
9           break
10      num = random.randint(min, max)
11      if num > winPercent:
12          print("恭喜猜對了！！！")
13      else:
14          print("猜錯了，請再試一次～")
```

```
請輸入莊家贏的比例 (1~100)：80
請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：l
猜錯了，請再試一次～
請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：s
猜錯了，請再試一次～
請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：s
猜錯了，請再試一次～
請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：s
猜錯了，請再試一次～
請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：s
猜錯了，請再試一次～
請猜大(l/L)或小(s/S)，或輸入 q/Q 離開程式：q
```

13-5:隨機數random模組

- choice():
  - 在一個串列(list)中隨機傳回一個元素。
  - 有一個水果串列，使用choice( )方法隨機選取一個水果。

```
1   import random
2
3   listFruit = ["蘋果","西瓜", "鳳梨", "香蕉"]
4
5   n = 3
6   for i in range(n):
7       print(random.choice(listFruit))
```

```
蘋果
蘋果
西瓜
```

```
蘋果
鳳梨
香蕉
```

13-5:隨機數random模組

- choice():
  - 骰子有6面點數各不相同，請擲骰子10次，將結果輸出。

```
1   import random
2
3   listDice = [1, 2, 3, 4, 5, 6]
4
5   n = 10
6   for i in range(n):
7       print(random.choice(listDice), end=" ")
```

```
3 5 1 6 3 2 5 1 5 6
4 6 5 3 2 2 4 4 1 1
```

  - 用 randint 也可以達成相同結果。

13-5:隨機數random模組

- shuffle():
  - 將串列元素重新排列。
  - 將串列內的樸克牌次序打亂，然後重新排列。

```python
import random

listPoker = ["A","2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"]

n = 3
for i in range(n):
    random.shuffle(listPoker)
    print(listPoker)
```

```
['2', 'Q', '9', 'K', '5', 'J', '3', '8', '4', '6', '7', '10', 'A']
['5', '9', 'Q', '6', '2', '10', 'A', '8', 'K', '4', 'J', '7', '3']
['3', 'Q', '5', '6', 'J', '8', '7', '2', 'A', 'K', '10', '9', '4']
```

13-5:隨機數random模組

- sample():
  - 隨機傳回第2個參數數量的串列元素。
  - sample(串列, 數量)

  - 設計大樂透彩卷號碼:
    - 由6個1-49數字組成，外加一個特別號。

```
1   import random
2
3   lotterys = random.sample(range(1, 50), 7)
4   specialNum = lotterys.pop()
5
6   print("本期大樂投開獎號碼為：", end = "")
7   for lottery in sorted(lotterys):
8       print(lottery, end = " ")
9   print(f"\n特別號為：{specialNum}")
```

```
本期大樂投開獎號碼為：3 9 19 29 45 48
特別號為：35
```

```
本期大樂投開獎號碼為：6 8 24 32 34 46
特別號為：20
```

13-5:隨機數random模組

- uniform():
  - uniform(x, y): 隨機產生大於等於 x，且小於 y 的浮點數。
    - x ≦ uniform(x, y) < y
  - 產生5筆 1-10之間隨機浮點數

```
1    import random
2
3    for i in range(5):
4        print(f"{i+1:2}. uniform(1, 10): {random.uniform(1, 10)}")
```

```
1. uniform(1, 10): 2.0498525218619057
2. uniform(1, 10): 4.332759800374222
3. uniform(1, 10): 4.343722058080104
4. uniform(1, 10): 4.043187225300045
5. uniform(1, 10): 8.944389475718246
```

```
1. uniform(1, 10): 8.552368707205744
2. uniform(1, 10): 7.331553839047453
3. uniform(1, 10): 5.372800108122020
4. uniform(1, 10): 9.255783127261590
5. uniform(1, 10): 8.612385077202426
```

13-5:隨機數random模組

- random():
  - 隨機產生大於等於 0 且小於 1 的浮點數。
    - 0 ≦ random() < 1
  - 產生5 筆0.0 – 1.0之間的隨機浮點數。

```
1    import random
2
3    for i in range(5):
4        print(f"{i+1:2}. random(): {random.random()}")
```

```
 1. random(): 0.44593259525414286
 2. random(): 0.689989750641941
 3. random(): 0.8052635110132584
 4. random(): 0.0365464139738695 96
 5. random(): 0.5334944088649892
```

```
 1. random(): 0.6003469253343282
 2. random(): 0.6728583951225393
 3. random(): 0.6194141946044467
 4. random(): 0.010011822533691483
 5. random(): 0.116040048591454 23
```

13-5:隨機數random模組

- seed():
  - seed(x): 設定 random 模組的種子值。
  - 讓每次使用隨機函數都會得到一樣的隨機值。

```
1    import random
2
3    random.seed(100)
4    for i in range(5):
5        print(f"{i+1:2}. random(): {random.random()}")
```

```
1. random(): 0.1456692551041303
2. random(): 0.45492700451402135
3. random(): 0.7707838056590222
4. random(): 0.705513226934028
5. random(): 0.7319589730332557
```

```
1. random(): 0.1456692551041303
2. random(): 0.45492700451402135
3. random(): 0.7707838056590222
4. random(): 0.705513226934028
5. random(): 0.7319589730332557
```

13-6:時間time模組

- time():
  - 回傳自1970年1月1日00:00:00AM以來的秒數。
  - 計算自1970年1月1日00:00:00AM以來的秒數。

```
1    import time
2
3    print(f"自1970年1月1日00:00:00AM以來的秒數 = {time.time()}")
4    print(f"自1970年1月1日00:00:00AM以來的秒數 = {time.time()}")
5    print(f"自1970年1月1日00:00:00AM以來的秒數 = {int(time.time())}")
```

```
自1970年1月1日00:00:00AM以來的秒數 = 1637939400.1361785
自1970年1月1日00:00:00AM以來的秒數 = 1637939400.1371727
自1970年1月1日00:00:00AM以來的秒數 = 1637939400
```

13-6:時間time模組

- time():
  - 計算花多少時間猜對數字

```python
import time
import random

min, max = 1, 10
ans = random.randint(min, max)
startTime = time.time()
while True:
    guessNum = int(input("請猜 1-10 之間的數字："))
    if guessNum == ans:
        print("猜對了～～～")
        break
    elif guessNum < ans:
        print("請猜大一點")
    else:
        print("請猜小一點")
endTime = time.time()
print(f"花了 {endTime - startTime} 秒猜對！！")
```

```
請猜 1-10 之間的數字：5
請猜小一點
請猜 1-10 之間的數字：3
猜對了～～～
花了 3.5144846439361572 秒猜對！！
```

```python
    endTime = time.time()
    print(f"花了 {int(endTime - startTime)} 
```

13-6:時間time模組

- sleep():
  - 讓工作暫停，這個方法的參數單位是秒。對於設計動畫非常有幫助。

```
1   import time
2
3   listFruits = ["蘋果","西瓜", "鳳梨", "香蕉"]
4   for fruit in listFruits:
5       print(fruit)
6       time.sleep(1)
```

```
蘋果
西瓜
鳳梨
香蕉
```

13-6:時間time模組

- localtime():
  - 用元組的資料結構傳回日期與時間。
  - 可用索引方式獲得個別內容：

索引	名稱	說明
0	tm_year	西元的年
1	tm_mon	月份，值是1~12
2	tm_mday	日期，值是1~31
3	tm_hour	小時，值是0~23
4	tm_min	分鐘，值是0~59
5	tm_sec	秒鐘，值是0~59
6	tm_wday	星期幾，0表示星期一，1表示星期二，...
7	tm_yday	代表這是一年中的第幾天
8	tm_isdst	夏令時間(日光節約時間)，0代表不是，1代表是。

13-6:時間time模組

- localtime():

```python
import time

tmTime = time.localtime()
print(tmTime)

#也可用 tmTime.tm_year
print(f"年: {tmTime[0]}")
print(f"月: {tmTime[1]}")
print(f"日: {tmTime[2]}")
print(f"時: {tmTime[3]}")
print(f"分: {tmTime[4]}")
print(f"秒: {tmTime[5]}")
print(f"星期幾: {tmTime[6]}")
print(f"第幾天: {tmTime[7]}")
print(f"夏令時間: {tmTime[8]}")
```

```
time.struct_time(tm_year=2021, tm_mon=11, tm_mday=26, tm_hour=23, tm_
min=46, tm_sec=31, tm_wday=4, tm_yday=330, tm_isdst=0)
年: 2021
月: 11
日: 26
時: 23
分: 46
秒: 31
星期幾: 4
第幾天: 330
夏令時間: 0
```

13-6:時間time模組

- asctime():
  - 列出目前系統時間。

```
1   import time
2
3   #print(time.asctime(time.localtime()))
4   print(time.asctime())
```

```
Fri Nov 26 23:32:19 2021
```

13-6:時間time模組

- ctime()
  - 以字串顯示日期與時間。

```
1   import time
2
3   #print(time.ctime(time.time()))
4   print(time.ctime())
5
6   print(time.ctime(1668528000))
7
8   #print(time.asctime(time.localtime()))
9   print(time.asctime())
```

```
Tue Nov 15 23:44:28 2022
Wed Nov 16 00:00:00 2022
Tue Nov 15 23:44:28 2022
```

13-6:時間time模組

time slot

- process_time()
  - 取得系統與程式執行的 CPU 時間的總和。此時間會排除 CPU 沒有運作的時間，例如：等待使用者輸入或是 sleep() 的時間。
    - 可連續呼叫兩次，計算中間差值得到執行某段程式碼所花時間。

```python
import time

x = 1000000
pi = 0.0
sign = 1
for i in range(1, x+1):
    pi += 4*sign/(2*i-1)
    sign = -sign
    if (i % 100000) == 0:
        e_time = time.process_time()
        print(f"When i = {i:7d}, Pi = {pi}, time elapsed = {e_time}")
```

```
When i =  100000, Pi = 3.1415826535897198, time elapsed = 0.078125
When i =  200000, Pi = 3.1415876535897618, time elapsed = 0.09375
When i =  300000, Pi = 3.141589320256464, time elapsed = 0.125
When i =  400000, Pi = 3.141590153589744, time elapsed = 0.140625
When i =  500000, Pi = 3.141590653589692, time elapsed = 0.15625
When i =  600000, Pi = 3.1415909869230147, time elapsed = 0.1875
When i =  700000, Pi = 3.141591225018261, time elapsed = 0.203125
When i =  800000, Pi = 3.1415914035897172, time elapsed = 0.234375
When i =  900000, Pi = 3.141591542478651, time elapsed = 0.25
When i = 1000000, Pi = 3.1415916535897743, time elapsed = 0.28125
```

13-6:時間time模組

- time, perf_counter, process_time

```python
import time

print(f"time()方法：{time.time()}")
print(f"perf_counter()方法：{time.perf_counter()}")
print(f"process_time()方法：{time.process_time()}")
t0 = time.time()
c0 = time.perf_counter()
p0 = time.process_time()
r = 0
for i in range(10000000):
    r += i
time.sleep(2)
print(r)
t1 = time.time()
c1 = time.perf_counter()
p1 = time.process_time()
spend1 = t1 - t0
spend2 = c1 - c0
spend3 = p1 - p0
print(f"time()方法用时：{spend1}s")
print(f"perf_counter()用时：{spend2}s")
print(f"process_time()用时：{spend3}s")
```

```
time()方法：1638202735.4989376
perf_counter()方法：0.0188387
process_time()方法：0.0625
49999995000000
time()方法用时：2.8421225547790527s
perf_counter()用时：2.8417293s
process_time()用时：0.828125s
```

13-7:系統sys模組

- version和version_info屬性

```
1   import sys
2
3   print(f"目前 Python 的版本是 {sys.version}")
4   print(f"目前 Python 的版本是 {sys.version_info}")
```

```
目前 Python 的版本是 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
目前 Python 的版本是 sys.version_info(major=3, minor=10, micro=5, releaselevel='final', serial=0)
```

13-7:系統sys模組

- stdin/stdout物件
  - standard input/output 的縮寫，是指從螢幕輸入/輸出。
  - 可搭配 readline()/write() 方法從螢幕輸入/輸出資料。

```
1 import sys
2
3 print("請輸入字串，輸入完按 Enter: ", end="")
4 msg = sys.stdin.readline()
5 print(msg)
```

```
請輸入字串，輸入完按 Enter: 歡迎大家來學 Python
歡迎大家來學 Python
```

```
1 import sys
2
3 print("請輸入字串，輸入完按 Enter: ", end="")
4 msg = sys.stdin.readline(8)
5 print(msg)
```

```
請輸入字串，輸入完按 Enter: 歡迎大家來學 Python
歡迎大家來學 P
```

```
1    import sys
2
3    sys.stdout.write("歡迎大家來學 Python")
```

```
歡迎大家來學 Python
```

13-7:系統sys模組

- platform屬性
  - 回傳目前 python 的使用平台。

```
1   import sys
2
3   print(sys.platform)
```
```
win32
```

13-7:系統sys模組

- path屬性
  - 是一串列資料，紀錄模組所在目錄。
  - 使用 import 的時候，會去這些目錄尋找檔案，然後匯入。

```
1   import sys
2
3   for dirPath in sys.path:
4       print(dirPath)
```

```
c:\[E]\PythonTest
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\python310.zip
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\DLLs
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\lib
C:\Users\Curtis\AppData\Local\Programs\Python\Python310
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\lib\site-packages
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\lib\site-packages\win32
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\lib\site-packages\win32\lib
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\lib\site-packages\Pythonwin
```

13-7:系統sys模組

- getwindowsversion()
  - 列出目前的Windows作業系統版本。

```
1   import sys
2
3   print(sys.getwindowsversion())
```

```
sys.getwindowsversion(major=10, minor=0, build=22621, platform=2, service_pack='')
```

- executable
  - 列出目前電腦Python可執行檔案路徑。

```
1   import sys
2
3   print(sys.executable)
```

```
C:\Users\Curtis\AppData\Local\Programs\Python\Python310\python.exe
```

13-7:系統sys模組

- 獲得getrecursionlimit( )與設定setrecursionlimit( )迴圈次數

```
1   import sys
2
3   sys.setrecursionlimit(100)
4   print(sys.getrecursionlimit())
```

```
100
```

13-7:系統sys模組

- DOS命令列(command line)引數
  - 列出命令列引數

```
1    import sys
2
3    print(f"命令列參數：{sys.argv}")
```

```
PS C:\[E]\PythonTest> python c:/[E]/PythonTest/test.py
命令列參數：['c:/[E]/PythonTest/test.py']
```

```
PS C:\[E]\PythonTest> python test.py 123 Hello World "Hello World"
命令列參數：['test.py', '123', 'Hello', 'World', 'Hello World']
```

13-8:keyword模組

- kwlist屬性
  - 列出所有Python關鍵字

```
1    import keyword
2
3    print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break'
, 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally',
'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal',
 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

13-8:keyword模組

- iskeyword()
  - 回傳參數的字串是否是關鍵字，是則回傳 True，否則回傳 False。

```
1   import keyword
2
3   listKw = ["as", "while", "break", "sse", "Python"]
4   for kw in listKw:
5       print(f"{kw:>8s} {keyword.iskeyword(kw)}")
```

```
      as True
   while True
   break True
     sse False
  Python False
```

13-9:日期calendar模組

- 包含一些日曆資料。
- 先 import calendar。

- 列出某年是否潤年isleap()

```
1   import calendar
2
3   print(f"2020 是閏年: {calendar.isleap(2020)}")
4   print(f"2021 是閏年: {calendar.isleap(2021)}")
```

```
2020 是閏年: True
2021 是閏年: False
```

13-9:日期calendar模組

- 印出月曆month()

```
1   import calendar
2
3   print(calendar.month(2022, 11))
```

```
    November 2022
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

13-9:日期calendar模組

- 印出年曆calendar()

```
1    import calendar
2
3    print(calendar.calendar(2022))
```

```
                                  2022

      January                 February                March
Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su
                1  2        1  2  3  4  5  6        1  2  3  4  5  6
 3  4  5  6  7  8  9     7  8  9 10 11 12 13     7  8  9 10 11 12 13
10 11 12 13 14 15 16    14 15 16 17 18 19 20    14 15 16 17 18 19 20
17 18 19 20 21 22 23    21 22 23 24 25 26 27    21 22 23 24 25 26 27
24 25 26 27 28 29 30    28                      28 29 30 31
31

      April                   May                     June
Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su
             1  2  3                    1           1  2  3  4  5
 4  5  6  7  8  9 10     2  3  4  5  6  7  8     6  7  8  9 10 11 12
11 12 13 14 15 16 17     9 10 11 12 13 14 15    13 14 15 16 17 18 19
18 19 20 21 22 23 24    16 17 18 19 20 21 22    20 21 22 23 24 25 26
25 26 27 28 29 30       23 24 25 26 27 28 29    27 28 29 30
                        30 31

       July                   August                  September
Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su
             1  2  3     1  2  3  4  5  6  7              1  2  3  4
 4  5  6  7  8  9 10     8  9 10 11 12 13 14     5  6  7  8  9 10 11
11 12 13 14 15 16 17    15 16 17 18 19 20 21    12 13 14 15 16 17 18
18 19 20 21 22 23 24    22 23 24 25 26 27 28    19 20 21 22 23 24 25
25 26 27 28 29 30 31    29 30 31                26 27 28 29 30

      October                 November                December
Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su    Mo Tu We Th Fr Sa Su
                1  2        1  2  3  4  5  6              1  2  3  4
 3  4  5  6  7  8  9     7  8  9 10 11 12 13     5  6  7  8  9 10 11
10 11 12 13 14 15 16    14 15 16 17 18 19 20    12 13 14 15 16 17 18
17 18 19 20 21 22 23    21 22 23 24 25 26 27    19 20 21 22 23 24 25
24 25 26 27 28 29 30    28 29 30                26 27 28 29 30 31
31
```

13-10:幾個增強Python功力的模組

- collections模組
  - defaultdict()
    - 為新建立的字典設定預設值。
    - 參數是一個函數
      - 如果為 int，表示是 int()，預設值會回傳 0。
      - 參數是 list 或是 dict，預設值會分別回傳[] 與 {}。
      - 若省略參數，預設會回傳 None。

```
1   from collections import defaultdict
2
3   fruits = defaultdict(int)
4   fruits["apple"] = 100
5   fruits["banana"]
6
7   print(fruits["apple"])
8   print(fruits["banana"])
9   print(fruits)
```

```
100
0
defaultdict(<class 'int'>, {'apple': 100, 'banana': 0})
```

13-10:幾個增強Python功力的模組

- collections模組
  - defaultdict()
    - 參數也可為自行設計的函數。

```
1   from collections import defaultdict
2
3   def price():
4       return 10
5
6   fruits = defaultdict(price)
7   fruits["apple"] = 100
8   fruits["banana"]
9
10  print(fruits["apple"])
11  print(fruits["banana"])
12  print(fruits)
```

```
100
10
defaultdict(<function price at 0x000001606D86F1F0>, {'apple': 100, 'banana': 10})
```

13-10:幾個增強Python功力的模組

- collections模組
  - defaultdict()
    - 參數也可為自行設計的函數 (lambda function)。

```
1   from collections import defaultdict
2
3   fruits = defaultdict(lambda:10)
4   fruits["apple"] = 100
5   fruits["banana"]
6
7   print(fruits["apple"])
8   print(fruits["banana"])
9   print(fruits)
```

```
100
10
defaultdict(<function <lambda> at 0x0000023D6BAE3E20>, {'apple': 100, 'banana': 10})
```

13-10:幾個增強Python功力的模組

- collections模組
  - defaultdict()
  - 使用defaultdict(int)時，可以利用此特性建立計數器。

```
1   from collections import Counter, defaultdict
2
3   dictFruits = defaultdict(lambda:0)
4   for fruit in ["apple", "banana", "apple"]:
5       if fruit not in dictFruits:
6           dictFruits[fruit]
7       dictFruits[fruit] += 1
8
9   for fruit, count in dictFruits.items():
10      print(fruit, count)
```

```
apple 2
banana 1
```

13-10:幾個增強Python功力的模組

- Counter()
  - 將串列元素轉成字典的 key，值則是元素在串列中出現的次數。
  - 資料型態是 Counter，內容元素是字典。

```
1   from collections import Counter
2
3   listFruits = ["apple", "banana", "apple"]
4   cnterFruits = Counter(listFruits)
5
6   print(cnterFruits)
7
8   for fruit, count in cnterFruits.items():
9       print(fruit, count)
```

```
Counter({'apple': 2, 'banana': 1})
apple 2
banana 1
```

13-10:幾個增強Python功力的模組

- Counter()
  - most_common()
    - most_common(n)方法如果省略參數 n，可以參考鍵:值的數量由大排到小傳回。n 是設定傳回多少元素。

```
1   from collections import Counter
2
3   listFruits = ["apple", "banana", "apple"]
4   cnterFruits = Counter(listFruits)
5   print(type(cnterFruits))
6   myFruitsAll = cnterFruits.most_common()
7   print(myFruitsAll)
8
9   myFruits0 = cnterFruits.most_common(0)
10  print(myFruits0)
11  myFruits1 = cnterFruits.most_common(1)
12  print(myFruits1)
13  myFruits2 = cnterFruits.most_common(2)
14  print(myFruits2)
```

```
<class 'collections.Counter'>
[('apple', 2), ('banana', 1)]
[]
[('apple', 2)]
[('apple', 2), ('banana', 1)]
```

13-10:幾個增強Python功力的模組

- Counter()
  - Counter物件的加與減
    - 可以使用加法 + 與減法 -，將2個 Counter 物件相加。
    - 相加的方式是所有元素相加，若是有重複的元素則鍵的值會相加。
    - 相減是所有元素相減，若是變負的就將元素刪除。

```
1   from collections import Counter
2
3   listFruitsA = ["apple", "banana", "apple"]
4   listFruitsB = ["grape", "banana", "banana", "grape"]
5   cnterFruitsA = Counter(listFruitsA)
6   cnterFruitsB = Counter(listFruitsB)
7
8   cnterFruitsAdd = cnterFruitsA + cnterFruitsB
9   print(cnterFruitsAdd)
10
11  cnterFruitsSub1 = cnterFruitsA - cnterFruitsB
12  print(cnterFruitsSub1)
13
14  cnterFruitsSub2 = cnterFruitsB - cnterFruitsA
15  print(cnterFruitsSub2)
```

```
Counter({'banana': 3, 'apple': 2, 'grape': 2})
Counter({'apple': 2})
Counter({'grape': 2, 'banana': 1})
```

13-10:幾個增強Python功力的模組

- Counter()
  - Counter物件的交集與聯集。
    - 可以使用&當作交集符號，|是聯集符號。
    - 聯集與加法不一樣它不會將數量相加，只是取多的部分。
    - 交集則是取數量少的部分。

```
1   from collections import Counter
2
3   listFruitsA = ["apple", "banana", "apple"]
4   listFruitsB = ["grape", "banana", "banana", "grape"]
5   cnterFruitsA = Counter(listFruitsA)
6   cnterFruitsB = Counter(listFruitsB)
7
8   cnterFruitsAnd = cnterFruitsA & cnterFruitsB
9   print(cnterFruitsAnd)
10
11  cnterFruitsUnion = cnterFruitsA | cnterFruitsB
12  print(cnterFruitsUnion)
```

```
Counter({'banana': 1})
Counter({'apple': 2, 'banana': 2, 'grape': 2})
```

13-10:幾個增強Python功力的模組

- deque()
  - 資料結構中的雙頭序列。
  - 具有堆疊stack與序列queue的功能。
    - 可以從左右兩邊增加元素，也可以從左右兩邊刪除元素。
  - pop() 方法可以移除右邊的元素並回傳，popleft() 可以移除左邊的元素並回傳。

13-10:幾個增強Python功力的模組

- deque()
  - ”回文(palindrome)”，從左右兩邊往內移動，如果相同就一直比對到中央，如果全部相同就是回文，否則不是回文。

```
1   from collections import deque
2
3   def palindrome(word):
4       wd = deque(word)
5       while(len(wd) > 1):
6           if wd.pop() != wd.popleft():
7               return False
8       return True
9
10  print(f"{palindrome('x') = }")
11  print(f"{palindrome('abccba') = }")
12  print(f"{palindrome('radar') = }")
13  print(f"{palindrome('python') = }")
```

```
palindrome('x') = True
palindrome('abccba') = True
palindrome('radar') = True
palindrome('python') = False
```

```
3   def palindrome(word):
4       """wd = deque(word)
5   >   while(len(wd) > 1):…
8       return True"""
9       return word == word[::-1]
```

13-10:幾個增強Python功力的模組

- pprint模組
  - pprint() 用法與print() 相同，不過 pprint() 會執行一行輸出一個元素，結果比較容易閱讀。

13-10:幾個增強Python功力的模組

- pprint模組

```python
import sys
from pprint import pprint
print("使用 print()")
print(sys.path)
print("使用 pprint()")
pprint(sys.path)
```

```
使用 print()
['c:\\[E]\\PythonTest', 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\python310.
zip', 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\DLLs', 'C:\\Users\\Curtis\\A
ppData\\Local\\Programs\\Python\\Python310\\lib', 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Pyt
hon\\Python310', 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-package
s', 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\win32', 'C
:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\win32\\lib', 'C:
\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\Pythonwin']
使用 pprint()
['c:\\[E]\\PythonTest',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\python310.zip',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\DLLs',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\win32',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\win32\\lib',
 'C:\\Users\\Curtis\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\Pythonwin']
```

13-10:幾個增強Python功力的模組

- itertools模組
  - chain()
    - 這個方法可以將chain( )參數的元素內容一一迭代出來

```
1   import itertools
2   for i in itertools.chain([1,2,3], ('a', 'b')):
3       print(i)
```

```
1
2
3
a
b
```

  - cycle()
    - 會產生無限迭代

```
1   import itertools
2   for i in itertools.cycle([1,2,3]):
3       print(i)
```

```
1
2
3
1
2
3
1
Traceback (most recent call last):
  File "e:\PythonTest\CH13\test.py", line 3, in <module>
    print(i)
KeyboardInterrupt
```

13-10:幾個增強Python功力的模組

- itertools模組
  - accumulate( )
    - 如果 accumulate( )只有一個參數，會列出累計的值。
    - 如果accumulate( )有2個參數，則第2個參數是函數，可以依照此函數列出累計的計算結果。

```
1   import itertools
2
3   for i in itertools.accumulate((1, 2, 3, 4, 5)):
4       print(i)
5
6   for i in itertools.accumulate((1, 2, 3, 4, 5), lambda x, y: x * y):
7       print(i)
```

```
1
3
6
10
15
1
2
6
24
120
```

13-10:幾個增強Python功力的模組

- itertools模組
  - combinations( )
    - 有兩個參數，第一個參數是可迭代物件，第二個參數是長度 r。
    - 會回傳一個長度為 r 的子序列，內容是各種元素的組合。

```
1   import itertools
2
3   x = ['a', 'b', 'c']
4   r = 2
5   comb = itertools.combinations(x, r)
6   print(list(comb))
```

```
[('a', 'b'), ('a', 'c'), ('b', 'c')]
```

13-10:幾個增強Python功力的模組

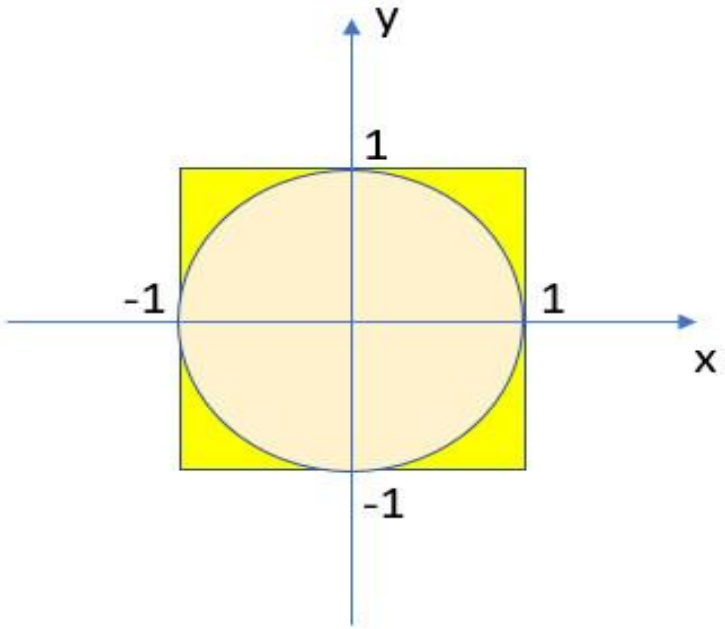
- string 模組
  - 內含一系列程式設計相關字串。

```
1   import string
2
3   print(string.digits)
4   print(string.hexdigits)
5   print(string.octdigits)
6   print(string.ascii_letters)
7   print(string.ascii_lowercase)
8   print(string.ascii_uppercase)
```

```
0123456789
0123456789abcdefABCDEF
01234567
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

13-11:動手練習

- 蒙地卡羅模擬
  - 使用蒙地卡羅模擬計算PI值，首先繪製一個外接正方形的圓，圓的半徑是1。



13-11:動手練習

- 蒙地卡羅模擬

由上圖可以知道矩形面積是 4，圓面積是 PI。

如果我們現在要產生 1000000 個點落在方形內的點，可以由下列公式計算點落在圓內的機率：

圓面積 / 矩形面積 = PI / 4

落在圓內的點個數(Hits) = 1000000 * PI / 4

如果落在圓內的點個數用 Hits 代替，則可以使用下列方式計算 PI。

PI = 4 * Hits / 1000000

程式實例 ch13_49.py：蒙地卡羅模擬隨機數計算 PI 值，這個程式會產生 100 萬個隨機點。

13-11:動手練習

- 蒙地卡羅模擬

```python
import random

trials = 1000000
Hits = 0
for i in range(trials):
    x = random.random() * 2 - 1
    y = random.random() * 2 - 1
    if x * x + y * y <= 1:
        Hits += 1
PI = 4 * Hits / trials
print(f"{PI = }")
```

```
PI = 3.14324
```

```
PI = 3.13996
```

```python
trials = 5000000
Hits = 0
```

```
PI = 3.1415224
```

```
PI = 3.1411624
```

13-11:動手練習

- 設計一個不公平的發牌機：
  - 公平的發牌機：每張牌的出現機率相等。
  - 不公平的發牌機：2~10 出現機率較高， A、K、Q、J 較低
  - 請用 random 設計一發牌機，讓2~10跟A、K、Q、J出現的機率不一樣。