



05：函式

2018.1.30

主題

- 自訂函式
- 函式參數、變數有效範圍
- 內建函式
- import 套件
- 常用函式

• 函式(Function)

- 在稍微大型的程式中，通常會將具有特定功能或經常重複使用的程式碼，撰寫成獨立的小單元，稱為函式。
- 函式有名稱，需要時可以呼叫該函式執行

```
2 def out():  
3     print("Hello World")  
4  
5 out()
```

3

• 函式

- 使用函式的程式設計方式有下列好處：
 - 將大程式切割後由多人撰寫，有利於團隊分工，可縮短程式開發的時間。
 - 可縮短程式的長度，程式碼也可重複使用，當再開發類似功能的產品時，只需稍微修改即可以套用。
 - 程式可讀性高，易於除錯和維護

4

● 自訂函式

- 建立函式的語法為：
 - `def` 函式名稱([參數1, 參數2, ...]):
 - 程式區塊
 - [`return` 回傳值1, 回傳值2, ...]

```
2 def fact(n):
3     if n == 1:
4         return 1
5     return n*fact(n-1)
```

5

● 自訂函式

- 函式建立後並不會執行，必須在程式中呼叫函式才會執行，呼叫函式的語法為：
 - [變數=] 函式名稱([參數串列])

```
2 # function
3 def factorial(n):
4     result = 1
5     for i in range(1, n+1):
6         result = result * i
7     return result
8
9 print(factorial(5))
```

6

參數預設值

- 建立函式時可以為參數設定預設值，呼叫函式時，如果沒有傳入該參數時，就會使用預設值。
 - 參數設定預設值的方法為「參數 = 值」，例如：

```
2 def get_area(width, height=12):  
3     return width * height  
4  
5 print(get_area(6))  
6 print(get_area(6, 10))
```

7

主題

- 自訂函式
- 函式參數、變數有效範圍
- 內建函式
- import 套件
- 常用函式

8

● 不定數目參數(1)

- 固定的參數數目有時並不方便
- 例如下列的加法函式，難以應用到多個數值相加：
 - `def add_num(a, b)`
 - `return int(a) + int(b)`

9

● 不定數目參數(2)

- Python 建立函式時可以讓函式接受沒有預先設定的參數個數，方法是在參數名稱前加星號「*」，語法為：
 - `def 函式名稱(*args):`
 - Python 會以數組將所有參數存於變數 `args` 中

10

範例

```
2 def add_sum(*args):
3     total_sum = 0
4
5     for i in args:
6         total_sum += i
7
8     return total_sum
9
10 print(add_sum(2, 3, 4))
11 print(add_sum(5, 5, 5, 5, 5, 5))
```

11

變數有效範圍

- 所謂變數的有效範圍是指能存取該變數的範圍

```
2 def add(n1, n2):
3     n3 = n1 + n2
4
5 print(n3)
```

12

變數

全域

定義在函式外的變數，其有效範圍是整個 Python 檔案

區域

定義在一個函式中的變數，有效範圍是在該函式內

13

撞名

- 若有相同名稱的全域與區域變數，以區域變數優先

```
2 def scope():
3     var1 = 1
4     print(var1, var2)
5
6 var1 = 10
7 var2 = 20
8 scope()
9 print(var1, var2)
```

```
2 def scope():
3     global var1
4     var1 = 1
5     var2 = 2
6     print(var1, var2)
7
8 var1 = 10
9 var2 = 20
10 scope()
11 print(var1, var2)
```

14

• 函式的參數傳遞方式

- C/C++ 的參數傳遞方式有兩種：
 - Pass by value：複製參數的值傳入，原參數內容不會被影響
 - Pass by reference：傳入參數的參考位址，會影響原參數內容

15

• Python ?

- Python 的傳遞方式比較特別，但是觀念上互通
 - 不可變更物件(**immutable object**)代表物件產生後就不可以被修改
 - e.g., boolean, int, float, complex, string, tuple
 - 可變更物件(**mutable object**)則是可以被修改
 - e.g., list, dict, set
- Python 採用的是 “pass by assignment”

16

範例

```
3 def change(input_str):
4     print("收到 input_str= ", input_str)
5     input_str = "台中"
6     print("修改 input_str= ", input_str)
7
8 init_str = "逢甲大學"
9
10 print("原始 init_str= ", init_str)
11 change(init_str)
12 print("更新後 init_str= ", init_str)
```

原始 init_str= 逢甲大學
收到 input_str= 逢甲大學
修改 input_str= 台中
更新後 init_str= 逢甲大學

原始字串
未變更

17

範例

```
3 def change(input_list):
4     print("收到 input_list= ", input_list)
5     input_list.append("香吉士")
6     print("修改 input_list= ", input_list)
7
8 init_list = ["魯夫", "索隆"]
9
10 print("原始 orig_list= ", init_list)
11 change(init_list)
12 print("更新後 orig_list= ", init_list)
```

原始 orig_list= ['魯夫', '索隆']
收到 input_list= ['魯夫', '索隆']
修改 input_list= ['魯夫', '索隆', '香吉士']
更新後 orig_list= ['魯夫', '索隆', '香吉士']

原始串列
已變更

18

注意

```
3 def change(input_list):
4     print("收到 input_list= ", input_list)
5     input_list = ["白鬍子", "艾斯"]
6     print("修改 input_list= ", input_list)
7
8 init_list = ["魯夫", "索隆"]
9
10 print("原始 orig_list= ", init_list)
11 change(init_list)
12 print("更新後 orig_list= ", init_list)
```

原始 orig_list= ['魯夫', '索隆']
收到 input_list= ['魯夫', '索隆']
修改 input_list= ['白鬍子', '艾斯']
更新後 orig_list= ['魯夫', '索隆']

原始串列
未變更

19

主題

- 自訂函式
- 函式參數、變數有效範圍
- 內建函式
- import 套件
- 常用函式

20

軍火庫

- Python 有許多內建函式(build-in function)供我們使用
 - e.g., int(), range()

21

內建函式(1)

函式	功能	範例	範例結果
abs(x)	取得 x 的絕對值	abs(-5)	5
chr(x)	取得整數 x 的字元	chr(65)	A
divmod(x, y)	取得 x 除以 y 的商及餘數的元組	divmod(44, 6)	(7,2)
float(x)	將 x 轉換成浮點數	float("56")	56.0
hex(x)	將 x 轉換成十六進位數字	hex(34)	0x22
★ int(x)	將 x 轉換成整數	int(34.21)	34
★ len(x)	取得元素個數	len([1,3,5,7])	4
★ max(參數串列)	取得參數中的最大值	max(1,3,5,7)	7
★ min(參數串列)	取得參數中的最小值	min(1,3,5,7)	1

22

內建函式(2)

函式	功能	範例	範例結果
oct(x)	將 x 轉換成八進位數字	oct(34)	0o42
★ ord(x)	回傳字元 x 的 Unicode 編碼值	ord("我")	25105
pow(x, y)	取得 x 的 y 次方	pow(2,3)	8
★ round(x)	以四捨六入法取得 x 的近似值	round(45.8)	46
★ sorted(串列)	由小到大排序	sorted([3,1,7,5])	[1,3,5,7]
★ str(x)	將 x 轉換成字串	str(56)	56 (字串)
★ sum(串列)	計算串列元素的總和	sum([1,3,5,7])	16
type(物件)	取得物件的資料型態	type(34.0)	float

23

內建函式(3)

- $\text{pow}(x, y, z) \rightarrow x^y \% z$
 - $\text{pow}(3, 4, 7) \# 4$
- $\text{round}(x, y) \rightarrow x$ 的小數點後 y 位四捨五入
 - $\text{round}(3.1415, 3) \# 3.142$
- $\text{sorted}([], \text{reverse}=\text{True}) \rightarrow$ 由大到小排序
 - $\text{sorted}([3, 1, 7, 5], \text{reverse}=\text{True}) \# [7, 5, 3, 1]$

24

主題

- 自訂函式
- 函式參數、變數有效範圍
- 內建函式
- **import 套件**
- 常用函式

25

import 命令

- Python 為人稱道的優勢之一就是擁有許多**內建套件** (**package**，又稱為 **module**)，也有很多第三方開發功能強大的套件。
 - **內建套件**只要使用「import」命令就可匯入
 - **第三方套件**要先安裝才能使用「import」命令匯入

26

• import 命令

- import 命令的語法為：
 - import 套件名稱
- 通常套件中有許多函式供設計者使用，使用這些函式的語法為：
 - 套件名稱.函式名稱

```
2 import math
3
4 print(math.pi)
```

27

• import 命令

- 有時候套件的名稱很長：
 - 例如繪圖表的套件 matplotlib
- 折衷辦法是使用另一個語法：
 - from 套件名稱 import *

28

• import 命令

- from 套件名稱 import *
- 此種方法雖然方便，卻隱藏著極大風險：
 - 每一個套件擁有眾多函式，若兩個套件具有相同名稱的函式，由於未輸入套件名稱，使用函式時可能造成錯誤。
- 為兼顧便利性及安全性，可使用 import 命令的第三種語法：
 - from 套件名稱 import 函式1, 函式2, ...

29

• import 命令

- 另一個選擇是為套件名稱另取一個簡短的別名，語法為：
 - import 套件名稱 as 別名

30

主題

- 自訂函式
- 函式參數、變數有效範圍
- 內建函式
- import 套件
- 常用函式

31

Import math

函式	描述
fab(x)	以浮點數回傳 x 的絕對值
ceil(x)	回傳大於 x 的最小整數
floor(x)	回傳小於 x 的最大整數
★ exp(x)	回傳 e^x
★ log(x)	回傳 $\ln(x)$
★ log(x, base)	回傳指定基底的對數值
sqrt(x)	回傳 $x^{1/2}$

32

• Import math

函式	描述
<code>sin(x)</code>	回傳以弧度為單位的正弦函數值
<code>asin(x)</code>	回傳以弧度為單位的反正弦函數值
<code>cos(x)</code>	回傳以弧度為單位的餘弦函數值
<code>acos(x)</code>	回傳以弧度為單位的反餘弦函數值
<code>tan(x)</code>	回傳以弧度為單位的正切函數值
<code>degree(x)</code>	將 x 角度從弧度(radian)轉為度數(degree)
<code>radian(x)</code>	將 x 角度從度數轉為弧度

33

• import random

- 可利用 random 模組來產生「偽」隨機數字(pseudo-random number)
 - `random()` # 隨機產生 0 ~ 1 間的亂數
 - `randint(a, b)` # 隨機產生 a ~ b 間的整數，包含 a, b

```
3 import random as ra
4
5 # 隨機生成 4 個介於 0~1 之間的亂數
6 random_num = [ra.random() for _ in range(4)]
7 # [0.14023412985656325, 0.4241773080193538,
8 # 0.08866258822152395, 0.26922801067217816]
```

34

種子數 (seed)

```
3 import random as ra
4
5 ra.seed(10)          # 把種子數設為 10
6 print(ra.random())   # 產生亂數 0.5714025946899135
7
8 ra.seed(10)          # 再把種子數設為 10
9 print(ra.random())   # 產生同一個亂數
10
11 # ra.random 預設採用目前的系統時間做為種子數
12
13 # 在 3~10之間隨機挑一個整數
14 print(ra.randint(3, 10))
15
16 # 隨機排列
17 num = [x for x in range(7)]
18 ra.shuffle(num)
19 print(num)           # [5, 2, 6, 1, 0, 4, 3]
```

35

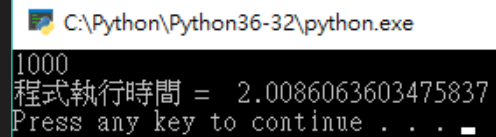
取後不放回 vs 放回

```
23 # 從串列中隨機挑選項目
24 russian_roulette = ra.choice(["索隆", "羅賓", "布魯克"])
25
26 # 取後不放回
27 lottery = [x for x in range(1, 49)]
28 ra.seed() # 記得先設定 seed
29 win_num = ra.sample(lottery, 6)
30 print(win_num)      # [42, 23, 12, 26, 4, 20]
31
32 # 取後放回
33 ra.seed()
34 num = [ra.choice(lottery) for _ in range(6)]
35 print(num)          # [42, 48, 23, 12, 26, 48]
```

36

• import time

```
1 # 計算執行時間
2
3 import time
4
5 start = time.clock()
6
7 time.sleep(2)
8 for x in range(1000):
9     x += 1
10    print (x)
11
12    end = time.clock()
13
14    print ("程式執行時間 = ", end - start)
```



C:\Python\Python36-32\python.exe
1000
程式執行時間 = 2.0086063603475837
Press any key to continue . . .

37

• PY3-0002

五、次方運算

2. 設計說明：

- (1) 請撰寫一程式，讓使用者輸入兩個整數，接著呼叫函式 `compute()`，此函式接收兩個參數 `a`、`b`，並回傳 a^b 的值。

38

● PY3-0002

六、眾數

2. 設計說明：

(1) 請撰寫一程式，讓使用者輸入十個整數作為樣本數，輸出眾數（樣本中出現最多次的數字）及其出現的次數。

* 提示：假設樣本中只有一個眾數。

TO BE CONTINUED...