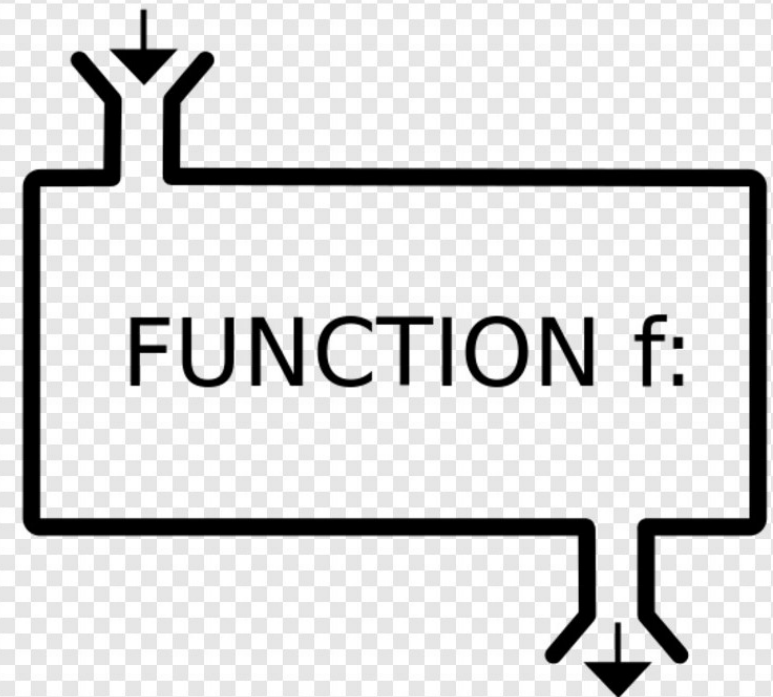


FUNCTION AND IMPORTING PACKAGES

Chih-Chung Hsu (許志仲)
Institute of Data Science
National Cheng Kung University
<https://cchsui.info>



INPUT x



OUTPUT $f(x)$

自訂函式

- 在一個較大型的程式中，通常會將具有特定功能或經常重複使用的程式，撰寫成獨立的小單元，稱為「函式」，並賦予函式一個名稱，當程式需要時就可以呼叫該函式執行。
- 使用函式的程式設計方式具有下列的好處：
 - 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。

自訂函式

■ 自訂函式

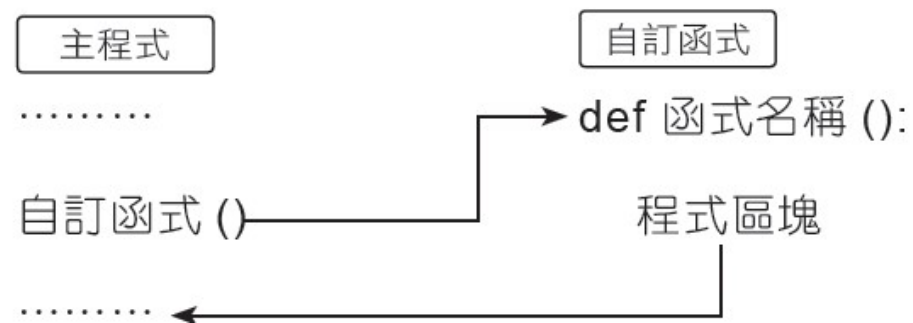
- Python 是以 `def` 命令建立函式，不但可以傳送多個參數給函式，執行完函式後也可返回多個回傳值。自行建立函式的語法為：

```
def 函式名稱 ([ 參數 1, 參數 2, …… ]):  
    程式區塊  
    [return 回傳值 1, 回傳值 2, ……]
```

- 參數 (參數 1, 參數 2, ……)：參數可以傳送一個或多個，也可以不傳送參數。參數是用來接收由呼叫函式傳遞進來的資料，如果有多個參數，則參數之間必須用逗號「,」分開。
- 回傳值 (回傳值 1, 回傳值 2, ……)：回傳值可以是一個或多個，也可以沒有回傳值。回傳值是執行完函式後傳回主程式的資料，若有多個回傳值，則回傳值之間必須用逗號「,」分開，主程式則要有多個變數來接收回傳值

Function 使用時機

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



- 函式有傳回值，
 - 可以使用變數來儲存返回值，
 - 多個傳回值用逗號分開 (半形)，然後有回傳，主程式中要承接時就必須要用相同數量的變數來存
- 函式沒有回傳值：不需要承接變數

Function 定義範例

- 函式本身有輸入值：用變數輸入，多變數用逗號間隔
- 函式沒有需要輸入：直接 ()
- 回傳結果的函式：以 **return** 加上要回傳的變數，多變數用逗號分隔
- 冒號【:】也是需要的，去 **identify** 可作用範圍

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

Function的輸入參數

■ 參數預設值

- 為了避免使用函式時因未傳入正確參數而產生錯誤，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**」，設定預設值的參數必須置於參數串列最後，否則執行時會產生錯誤數。

```
6 ✓ def convert(a=10):  
7     print(a+10)  
8  
9     convert()
```

20

Function的輸入參數

■ 變數有效範圍

- 變數依照其有效範圍分為全域變數及區域變數：
 - 全域變數：定義在函式外的變數，其有效範圍是整個 Python 檔案。
 - 區域變數：定義在一個函式中的變數，其有效範圍是在該函式內。
- 若有相同名稱的全域變數與區域變數，以區域變數優先：在函式內，會使用區域變數，在函式外，因區域變數不存在，所以使用全域變數。

```
5 global_var = 100
6 def convert(a=10):
7     local_var = 10
8     print(a+local_var+global_var)
9
10 convert()
11 print(global_var)
12 print(local_var)
```

```
120
100
```

NameError: name 'local_var' is not defined

多重參數輸入 Multiple augments

■ *args and **kwargs

- *args是彈性的positional arguments列表
- **kwargs是彈性的keyword arguments列表。
- 兩個可以同時使用，但在使用時，*args必須在**kwargs的前面，因為positional arguments，有位置順序的對應，必須位於keyword arguments之前。

```
1 def example(a, *args, b=1):  
2     print('a=', a)  
3     print('b=', b)  
4     print('*args=', args)  
5     return a  
6  
7 example(22, 33, 44, b=4)  
8 # print(local_var)
```

```
a= 22  
b= 4  
*args= (33, 44)  
22
```


多重參數輸入 Multiple augments

■ **kwargs

- 範例程式類似，但注意提順序
 - *args回傳 tuple, **kwargs回傳 Dictionary (why)

```
1  def example(a, *args, **kwargs):  
2      print('a=', a)  
3      print('*args=', args)  
4      print '**kwargs=', kwargs)  
5      return a  
6  
7  example(22, 33, 44, b=4, c=5)  
8  # print(local_var)
```

```
· a= 22  
  *args= (33, 44)  
  **kwargs= {'b': 4, 'c': 5}  
22
```

數值函式

- 數值相關函式整理

- Python 中常用的數值函式有 (僅針對純Python data type)

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

數值函式

函式	功能	範例	範例結果
max(參數串列)	取得參數串列中的最大值	max(1,3,5,7)	7
min(參數串列)	取得參數串列中的最小值	min(1,3,5,7)	1
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

重點數值計算函式

- 指數、商數、餘數及四捨六入

- pow 函式

- pow 函式不但可以做指數運算，還可以計算餘數，語法為：

`pow(x, y[, z])`

- 如果只有 x 及 y 參數，傳回值為 x 的 y 次方，若有 z 參數，意義為 x 的 y 次方除以 z 的餘數。

- divmod 函式

- divmod 函式會同時傳回商數及餘數，語法為：

`divmod(x, y)`

round 函式

- round 函式以四捨六入法取得 x 的近似值，語法為：

```
round(x[, y])
```

- 四捨六入是 4 以下(含)捨去，6 以上(含)進位，5 則視前一位數而定：前一位數是偶數就將 5 捨去，前一位數是奇數就將其進位。
- 如果只有 x 參數，傳回值為 x 的四捨六入整數值

```
round(3.4)    #3
```

```
round(3.6)    #4
```

```
round(3.5)    #4, 前一位是奇數，進位
```

```
round(4.5)    #4, 前一位是偶數，捨去
```

- 若有 y 參數，y 是設定小數位數

```
round(3.75, 1)  #3.8
```

```
round(3.65, 1)  #3.6
```

最大值、最小值、總和及排序

- 最大值及最小值

- **max** 函式可取得一群數值的最大值，**min** 函式可取得一群數值的最小值，兩者用法相同。以 **max** 函式為例，其參數可以是多個參數，也可以是串列，語法為：

`max(數值 1, 數值 2, ……)` 或者
`max(串列)`

- 計算總和

- **sum** 函式可計算串列中所有數值的總和，語法為：

`sum(串列 [, 額外數值])`

- 如果有傳入「額外數值」參數，則此額外數值也會被加入總和之中

排序

- `sorted` 函式可將串列中的值排序，語法為：

```
sorted( 串列 [, reverse=True|False])
```

- `reverse` 參數的預設值 `False`，即沒有傳入 `reverse` 參數時，預設是由小到大排序。若是以「`reverse=True`」做為第 2 個參數傳入，就會由大到小排序。

```
sorted(iterable, key=None, reverse=False)
```

- `key` 是一種彈性運用，結合 `Function` 可以指定複雜格式的排序方法


Sorted by key

```
mydata = [
    ('Kim', 88, 60),
    ('Karen', 73, 94),
    ('Min', 99, 67),
    ('Jimmy', 87, 87),
    ('Steven', 81, 87)
]

## Assumed that the second column is for math and the last
## one stands for programming's grade.

def keysort(item):
    weighted_score = 0.3*item[1] + 0.7*item[2]
    return weighted_score

sorted_list = sorted(mydata, key=keysort, reverse=True)
print(sorted_list)
```



Lambda: simplified function

Lambda: simplified function

- Lambda: 一種簡單的function單行表示法

`lambda arguments : expression`

```
1  x = lambda a, b : a * b
2
3  def x2(a, b):
4      return a*b
5
6  print(x(5, 10))
7  print(x2(5, 10))
8
```

50

50