3. Python進階

3.1 函式、模組與套件
3.2 容器型態
3.3 類別與物件

3.1 函式、模組與套件

3-1函式、模組與套件

1.函式定義與呼叫

建立一個基本函式「printMsg」，
為沒有參數和傳回值的範例 。

步驟1

```
1 def printMsg():
2     print("hello world")
3
4 printMsg()
```

➢ 執行結果

```
In [1]: runfil
Users/ted/Desk
hello world

In [2]:
```

3-1函式、模組與套件

1.函式定義與呼叫

建立帶有參數的函數式「printMsg()」

步驟2

```
1 def printMsg(s):
2     print(s)
3
4 printMsg('hello Python')
```

➢ 執行結果

```
In [8]: runfile('C:/Users/ted/Desktop/test/
3.1.2.py', wdir='C:/Users/ted/Desktop/test')
hello Python
```

3-1函式、模組與套件

1.函式定義與呼叫

步驟3

建立帶有參數與回傳值的函數式「printMsg()」
呼叫函式時需要傳入相對參數。
函式區塊中可使用「return」傳回資料並結束函式。

```
1 def printMsg(s):
2     return('Hi ' + s)
3
4 print(printMsg('hello Python'))
```

➢ 執行結果

```
In [9]: runfile('C:/Users/ted/Desktop/test/
3.1.3.py', wdir='C:/Users/ted/Desktop/test')
Hi hello Python
```

3-1函式、模組與套件

2.使用模組與套件

1.匯入模組與套件

Python使用「import」關鍵字匯入模組或套件，
以下範例匯入亂數「random」模組，
接著呼叫模組函式產生亂數值。

```
1 import random
2
3 print(random.random())
4 print(random.randint(1,100))
```

產生0~1之間亂數
產生0~100之間亂數整數

➢ 執行結果

```
In [12]: runfile('C:/Users/ted/Desktop/test/
3.1.4.py', wdir='C:/Users/ted/Desktop/test')
0.1217050019577767
15
```

3-1函式、模組與套件

2.使用模組與套件

2.模組與套件別名

Python所匯入的模組或套件，除了使用名稱來呼叫之外，實作上為了方便，也可使用「as」關鍵字替模組取「別名」，之後可使用別名呼叫模組。

```
1 import random as R
2
3 print(R.random())
4 print(R.randint(1,100))
```

- 執行結果

```
In [12]: runfile('C:/Users/ted/Desktop/test/
3.1.4.py', wdir='C:/Users/ted/Desktop/test')
0.1217050019577667
15
```

3-1函式、模組與套件

2.使用模組與套件

3.匯入部分模組或套件

Python所匯入的模組或套件預設為該模組或套件的全部內容，
但是在實務上通常不會用到全部的內容，
這時就只要匯入部分內容即可，使用「from … import …」

```
1 from os import path
2
3 file   = path.realpath(__file__)
4 folder = path.dirname(file)
5 newfile= path.join(folder, 'test.txt')
6
7 print(file)
8 print(folder)
9 print(newfile)
```

➢ 執行結果

```
C:\Users\ted\Desktop\test\3.1.6.py
C:\Users\ted\Desktop\test
C:\Users\ted\Desktop\test\test.txt
```

3.2 容器型態

3-2 容器型態

容器型態(Containers)可儲存一系列資料，
容器型態就像一個放東西的盒子，將資料儲存在盒子中。

Python支援的容器型態包含:

- 元組(Tuple)：　　( ) 集合
- 清單(List)：　　[ ] 中括號
- 字典(Dictionary)：　{ } 大括號
- 集合(Set)：　　{ } 大括號

3-2 容器型態

1.元組()

- 「元組」(Tuple)資料是唯讀的，一旦程式指定元組的資料，就不允許更改它的內容。
- 元組通常用在像是紀錄原點(0,0)或是確定內容不會改變的資料
- Python使用()來建立元組，每一個項目使用逗號分隔

```
1 tp = (3,4,5,6)
2 print(type(tp))      # <class 'tuple'>
3 print(tp)            # (3, 4, 5, 6)
4 print(tp[0])         # 3
5 print(tp[1])         # 4
6 print(tp[-1])        # 6
7
8 for ele in tp:
9     print(ele, end = " ")
10
11                     # 3 4 5 6
```

3-2 容器型態

2.清單[ ]

- 「清單」(List)類似其他語言的「陣列」(Array)，名為清單、串列或列表。
- 清單允許更改內容，可以新增、插入、刪除和更改清單的項目

1.清單的基本使用

- 「清單」使用[ ]中括號來括起多個子項目，每一個項目使用「，」逗號分開

```
1 lst = [3,4,5]
2 print(lst)                # [3, 4, 5]
3 print(lst[0])             # 3
4 print(lst[-1])            # 5
5 lst[2] = 'py'
6 print(lst)                # [3, 4, 'py']
7 lst.append('foo')
8 print(lst)                # [3, 4, 'py', 'foo']
```

list[0]為第一個元素
list[-1]為倒數第一個元素

append(子項目)，新增子項目到list尾端

3-2 容器型態

2.清單[ ]

2.切割清單

- 可以在中括號內使用「:」符號語法，指定開始和結束點，將清單分割成子清單。

```
lst = list(range(5))
print(lst)                  # [0, 1, 2, 3, 4]
print(lst[2:4])             # [2, 3]
print(lst[2:])              # [2, 3, 4]
print(lst[:2])              # [0, 1]
print(lst[:])               # [0, 1, 2, 3, 4]
print(lst[:-1])             # [0, 1, 2, 3]
lst[2:4] = [8,9]
print(lst)                  # [0, 1, 8, 9, 4]
```

3-2 容器型態

2.清單[ ]

2.切割清單

- 如果需要顯示清單內各個項目的索引值(排序順位)和內容，可使用「enumerate(清單)」

➢ 執行結果

```
1 ani = ['cat', 'dog', 'bird']
2 for idx , item in enumerate(ani) :
3     print(idx, item)
```

```
0 cat
1 dog
2 bird
```

3-2 容器型態

2.清單包含

- 「清單包含」使用簡潔語法建立清單，
可以適當地在[ ]中使用for迴圈產生所需要的清單項目

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

- 中括號第一個x不僅可以是變數，也可以是個運算式

```
1 b = [x+1 for x in range(10)]
2 print (b)   # [1, 2, 3, 4, 5, 6, ,7, 8, 9, 10]
```

3-2 容器型態

2.清單包含

- 「清單包含」使用簡潔語法建立清單，
  可以適當地在[ ]中使用for迴圈產生所需要的清單項目，
  還可加上if條件篩選所需的項目

- Ex:只顯示奇數項目

```
1 c = [x for x in range(10) if x % 2 == 1]
2 print (c)   # [1, 3, 5, ,7, 9,]
```

- 也可使用運算式產生項目

```
1 e = [x**2 for x in range(10) if x % 2 == 0]
2 print (e)   # [0, 4, 16, ,36, 64,]
```

3-2 容器型態

3.字典

1.字典的基本用途

- 「字典」常用來儲存鍵(key)和值(value)，
  可以使用鍵(key)來取用和更改內容值(value)，
  也可透過鍵和值來新增或刪除項目，類似於其他語言的陣列。
- Python的字典是使用「{ }」來定義鍵和值(key-value-pairs)，
  每一對要使用「,」逗號分格，而鍵和值是使用「:」 分隔。

```
1a = {'cat':'miao', 'dog':'wang'}# 建立字典
2print(a['dog'])                 # wang
3print('dog' in a)               # True
4a['bird'] = 'juju'              # 新增 bird 項目
5print(a.get('eagle', 'N/A'))    # 取出 eagle 項目
6print(a.get('bird', 'N/A'))     # 取出 bird 項目
7del a['bird']                   # 刪除 bird 項目
8print(a.get('bird', 'N/A'))     # 取出 bird 項目
```

取出key為「eagle」的值，若不存在，則印出「N/A」

➢ 執行結果

```
wang
True
N/A
juju
N/A
```

3-2 容器型態

3.字典

2.走訪字典

- 如清單一樣，python程式一樣使用for迴圈，
  透過key來走訪字典，取得對應值。

```
1 a = {'bird':2, 'dog':4, 'cat':4}
2
3 for item in a:
4     leg = a[item]
5     print(item, leg)
```

➢ 執行結果

```
bird 2
dog 4
cat 4
```

3-2 容器型態

3.字典

2.走訪字典

- 如果需要同時走訪字典的鍵和值，
  這時則需要使用`item()` 方法

```
1 a = {'bird':2, 'dog':4, 'cat':4}
2
3 for idx, leg in a.items():
4     print('動物 %s 有 %d 隻腳' % (idx, leg))
```

➢ 執行結果

```
動物 bird 有 2 隻腳
動物 dog 有 4 隻腳
動物 cat 有 4 隻腳
```

3-2 容器型態。

3.字典1

3.字典包含

「字典包含」是一種簡潔語法來建立字典的方法，
我們可以在{ }中使用for迴圈，來產生字典項目，
還可以再加上if 條件子句來篩選所需要的項目。

```
1 dict = {x : x**2 for x in range(5)}
2
3 print(dict) # {0:0, 1:1, 2:4, 3:9, 4:16}
```

```
1 dict = {x : x**2 for x in range(10) \
2         if x % 2 == 0}
3
4 print(dict) # {0:0, 2:4, 4:16, 6:36, 8:64}
```

3-2 容器型態

4.集合

- 「集合」(Sets)是一種沒有順序的元素集合，每一個元素都是唯一，不能重複，
- 可更新、新增和刪除項目，並可執行交集、聯集和差集等集合運算

1.集合的基本功能

- 跟字典相同，集合也使用「{ }」大括號，每一個元素使用「,」逗號分隔。

```
1 s = {'dog', 'cat', 'bird'}
2 print('dog' in s)          # True
3 print('monkey' in s)       # False
4 s.add('monkey')
5 print('monkey' in s)       # True
6 print(len(s))              # 4
7 s.add('dog')                  # 新增已存在的「dog」元素
8 print(len(s))              # 4  # 印出集合s的長度，會發現新加的「dog」不會重複
9 s.remove('dog')
10 print(len(s))             # 3
```

3-2 容器型態

4.集合

2.走訪集合

- 「走訪集合」和「走訪清單」作法相同，但因集合沒有順序，所以不能使用順序來走訪，必須要用「enumerate(集合)」

```
1 s = {'dog', 'cat', 'bird'}
2 for idx, ani in enumerate(s):
3     print('#%d: %s' %(idx+1, ani))
```

➢ 執行結果

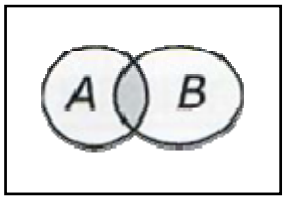
```
#1: dog
#2: bird
#3: cat
```

3-2 容器型態

4.集合

3. 集合運算

- **交集(Intersection)**
- 聯集(Union)
- 差集(Difference)
- 對稱差集(Symmetric Difference)



交集

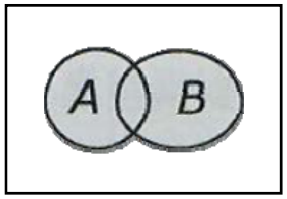
```
1 a = {1,2,3,4,5}
2 b = {4,5,6,7,8}
3
4 c = a & b
5 print(c)                  # {4, 5}
6
7 d = a.intersection(b)
8 print(d)                  # {4, 5}
9
```

3-2 容器型態

4.集合

3. 集合運算

- 交集(Intersection)
- **聯集(Union)**
- 差集(Difference)
- 對稱差集(Symmetric Difference)



聯集

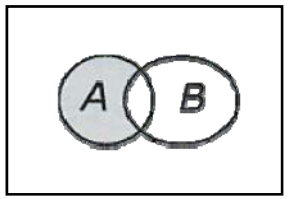
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}

c = a | b
print(c)        # {1,2,3,4,5,6,7,8}

d = a.union(b)
print(d)        # {1,2,3,4,5,6,7,8}

```

3-2 容器型態

4.集合

3. 集合運算

- 交集(Intersection)
- 聯集(Union)
- **差集(Difference)**
- 對稱差集(Symmetric Difference)



差集

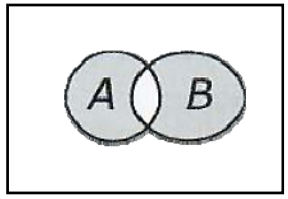
```
1 a = {1,2,3,4,5}
2 b = {4,5,6,7,8}
3
4 c = a - b
5 print(c)                # {1,2,3}
6
7 d = a.difference(b)
8 print(d)                # {1,2,3}
```

3-2 容器型態

4.集合

3. 集合運算

- 交集(Intersection)
- 聯集(Union)
- 差集(Difference)
- **對稱差集(Symmetric Difference)**



對稱差集

```
1 a = {1,2,3,4,5}
2 b = {4,5,6,7,8}
3
4 c = a ^ b
5 print(c)                  # {1,2,3,6,7,8}
6
7 d = a.symmetric_difference(b)
8 print(d)                  # {1,2,3,6,7,8}
```

實作練習2:

1. 請建立一個函數，回傳10個1~50之間的隨機整數；
2. 再建立另一個函數，輸入10個參數，會輸出這10個參數的平方和。

- 執行結果

10個隨機整數為：[30, 22, 46, 36, 40, 22, 50, 37, 42, 2]
平方和 = 12517

補充資料(不教學)

3.3 類別與物件

3-3 類別與物件

1.定義類別並建立物件

1. 定義類別

- Python是物件導向程式語言，
  所有內建的資料型態都是物件，
  包括模組、函式也都是物件
- 類別(Class)是物件的藍圖，
  我們需要先定義類別，才能按照類別來建立物件。

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
```

說明

第 1 行：定義「Univ」類別。類別內的「self」是指類別本身，有加上「self」才算是類別內的成員。例如：「self.name」代表類別內的 name，「show(self)」代表類別內的 show 函式。

第 3 行：類別「建構子」，包含必要的「self」欄位和 2 個自訂參數。初始化時必須傳入該 2 個自訂參數：「name」及「rank」。有關建構子，請參考如下更多說明。

第 4 行：設定「成員變數 name」的值為「參數 name 值」。成員變數意思為類別內的變數。

第 5 行：設定「成員變數 rank」的值為「參數 rank 值」。

第 7 行：定義 show(self) 函式，功能是印出「name」與「rank」，包含必要的「self」欄位，呼叫時不用傳入任何參數。

第 8 行：印出「成員變數 name」。

第 9 行：印出「成員變數 rank」。因為「rank」為數字，所以要加上「str( )」才能轉成字串型式。

第 11 行：定義 getName(self) 函式，包含必要的「self」，功能是返回「name」值。

第 12 行：返回「成員變數 name」。

3-3 類別與物件

1.定義類別並建立物件

2. 類別建構子

- 類別建構子

是每一次透過類別建立新物件時就會執行的特定方法，
python的類別建構子名稱固定為「__init__」，
不能改名，在init前後是連續兩個「_」

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
```

說明

上述建構子寫法和 Python 函式相同，只是前後多了 2 個「_」，並且名稱固定為「init」。建立新物件時，也可以使用參數來指定資料欄位「name」和「rank」初值。

3-3 類別與物件

1.定義類別並建立物件

3. 建構子的self變數

- python的類別建構子與各種方法的第一個參數固定是self變數，這是一個特殊變數，功能相當於其他語言的this關鍵字。

Tips: self 不是 Python 關鍵字，只是約定俗成變數名稱。self 變數值是新建立物件的自身參考，以建構子 __init__() 方法來看，參數 self 的值即為新建立的物件參考。

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
```

3-3 類別與物件

1.定義類別並建立物件

4. 資料欄位

- 類別的資料欄位或稱為「成員變數」，
  定義時不需特別語法，只需要以self開頭存取的變數，
  就是「資料欄位」

在「Univ」類別的資料欄位有「name」和「rank」。

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
```

說明

上述程式碼「self.name」和「self.rank」就是新物件資料欄位「name」和「rank」的值。

3-3 類別與物件

1.定義類別並建立物件

5. 方法

- 類別的「方法」就是python的函式，
只是第一個參數一定要是「self」變數，
而且在存取資料欄位時，一定要使用self變數存取。

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
```

說明

第7行：類別內的方法，記得要加「self」。

第8行：存取內部資料欄位時，記得也要加「self」。

第9行：使用「str()」，將「self.rank」轉成字串。

3-3 類別與物件

1.定義類別並建立物件

6. 有了類別才能建立物件

- 定義類別後，便可以透過類別來建立物件。
所建立的物件也稱為實例(Instances)，
同一類別可以建立多個物件

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
13
14 u1 = Univ('麻省理工學院', 1)
```

說明

第 1 ～ 12 行：類別定義。

第 14 行：建立一個名稱為「u1」的 Univ 物件，並傳入 2 個參數。

3-3 類別與物件

1.定義類別並建立物件

- 上述程式碼建立物件u1並傳入兩個參數，
  接下來便可使用「.」點運算子存取資料和呼叫物件方法。

7. 使用物件

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
13
14 u1 = Univ('麻省理工學院', 1)
15
16 print(u1.name)
17 print(u1.rank)
```

說明

第 1 ～ 12 行：類別定義。

第 14 行：建立一個名稱為「u1」的 Univ 物件，並傳入 2 個參數。

第 16 行：用「.」運算子存取資料「self.name」，使用「u1.name」。

第 17 行：用「.」運算子存取資料「self.rank」，使用「u1.rank」。

➢ 執行結果

```
麻省理工學院
1
```

3-3 類別與物件

1.定義類別並建立物件

7. 使用物件

- 存取物件資料時，雖然「u1.name」寫法很方便，但實作時通常不希望這些資料隨便被存取。這時候可以透過函式來達成

```
1 class Univ:
2
3     def __init__(self, name, rank):
4         self.name = name
5         self.rank = rank
6
7     def show(self):
8         print('學校 ' + self.name)
9         print('排名 ' + str(self.rank))
10
11    def getName(self):
12        return self.name
13
14 u1 = Univ('麻省理工學院', 1)
15
16 u1.show()
17
```

說明

第 1～12 行：類別定義。

第 14 行：建立一個名稱為「u1」的 Univ 物件，並傳入 2 個參數。

第 16 行：使用函式「show( )」存取內部資料。

➢ 執行結果

```
學校 麻省理工學院
排名 1
```

3-3 類別與物件

1.定義類別並建立物件

除了使用函式外，
要隱藏資料欄位也可以在變數前面加上「__」(兩個「_」)，
即可以從「公開」變成「非公開」

7. 使用物件

以下新增一個非公開資料欄位「__population」

```
1 class Univ:
2
3     def __init__(self, name, rank, pop):
4         self.name = name
5         self.rank = rank
6         self.__population = pop
7
8     def show(self):
9         print('學校 ' + self.name)
10        print('排名 ' + str(self.rank))
11
12    def getName(self):
13        return self.name
14
15 u1 = Univ('麻省理工學院', 1, 1000)
16
17 print(u1.__population)
```

說明

第 3 行：新增一個參數「pop」。
第 6 行：新增一個非公開資料「__population」。
第 15 行：產生成「u1」物件，並傳入 3 個參數。
第 17 行：直接存取非公開資料「__population」，可造成錯誤訊息如執行結果。

➢ 執行結果

```
AttributeError: 'Univ' object has no
attribute '__population'
```

說明

錯誤訊息：找不到 '__population'，表示非公開成員真的被隱藏了。

3-3 類別與物件

1.定義類別並建立物件

如果資料被隱藏起來，便要透過函式能從外部看到

7. 使用物件

```
1  class Univ:
2
3      def __init__(self, name, rank, pop):
4          self.name = name
5          self.rank = rank
6          self.__population = pop
7
8      def show(self):
9          print('學校 ' + self.name)
10         print('排名 ' + str(self.rank))
11
12     def getPop(self):
13         return self.__population
14
15 u1 = Univ('麻省理工學院', 1, 1000)
16
17 print(u1.getPop())        # 1000
```

說明

第 12 ～ 13 行：新增一個函式「getPop(self)」，功能是回傳非公開變數「__population」。

第 15 行：生成「u1」物件，並傳入 3 個參數。

第 17 行：透過「getPop( )」存取非公開資料「__population」。

- 執行結果

```
1000
```

3-3 類別與物件

1.定義類別並建立物件

7. 使用物件

不只資料欄位能夠以「__」隱藏，
「方法(函式)」也可以用同樣方式隱藏。

```
1 class Univ:
2
3     def __init__(self, name, rank, pop):
4         self.name = name
5         self.rank = rank
6         self.__population = pop
7
8     def show(self):
9         print('學校 ' + self.name)
10        print('排名 ' + str(self.rank))
11
12    def getPop(self):
13        return self.__population
14
15    def __caculate():
16        print('This is internal.')
```

說明

第 15 行：使用「__」定義一個非公開函式「__caculate()」。

3-3 類別與物件

1.定義類別並建立物件

非公開函式只能在物件內部呼叫

7. 使用物件

```
1 class Univ:
2
3     def __init__(self, name, rank, pop):
4         self.name = name
5         self.rank = rank
6         self.__population = pop
7
8     def show(self):
9         print('學校 ' + self.name)
10        print('排名 ' + str(self.rank))
11        self.__caculate()
12
13    def getPop(self):
14        return self.__population
15
16    def __caculate(self):
17        print('This is internal.')
18
19 u1 = Univ('麻省理工學院', 1, 1000)
20
21 u1.show()
```

說明

第 11 行：物件內部呼叫非公開函式「__caculate()」，前面記得加「self」關鍵字。

第 16 行：使用「__」定義一個非公開函式「__caculate()」。

第 19 行：生成「u1」物件，並傳入 3 個參數。

第 21 行：外部呼叫「show()」函式，show() 再呼叫內部函式「__caculate()」。

➢ 執行結果

```
學校 麻省理工學院
排名 1
This is internal.
```

說明

「__caculate()」是非公開函式，只能在物件內部呼叫。