

---

# Python程式設計

## 資料型態

---

高師大數學系

葉倚任

---

# Python 數值型態

---

- 整數
- 浮點數
- 布林型態
- 複數

# 整數型態

- 型態為 int，不再區分整數與長整數
- 整數的長度不受限制(除了硬體上的限制之外)

In [9]:

```
tmp = 1  
type(tmp)
```

← 不用事先宣告變數型態

Out[9]: int

← 用type()函式可以知道資料型態

# 整數型態-從其他形態轉成整數

- 想從字串、浮點數布林等型態轉成整數，可以使用 `int()` 函式。

```
In [19]: num = int('10')
          print(type(num), num)
          fnum = int(9.487)
          print(type(fnum), fnum)
          bnum = int(True)
          print(type(bnum), bnum)

<class 'int'> 10
<class 'int'> 9
<class 'int'> 1
```

從字串轉成整數時，可以指定幾進位

```
In [23]: num1 = int('100')
          print('num1 =', num1)
          num2 = int('100', 2)
          print('num2 =', num2)

num1 = 100
num2 = 4
```

# 浮點數型態

## ■ float 型態

```
In [27]: fnum1 = 95.27  
         print(type(fnum1), fnum1)  
         fnum2 = 9.527e1  
         print(type(fnum2), fnum2)  
  
         <class 'float'> 95.27  
         <class 'float'> 95.27
```

← 一樣不用事先宣告變數型態

← 可以用科學記號表示方式

# 布林型態

- bool 型態
- 只有 True 與 False 兩個值
- bool() 可將0轉換為False，而非0值轉換為 True

```
In [31]: bnum1 = True
          print(type(bnum1), bnum1)
          bnum2 = bool(1)
          print(type(bnum2), bnum2)
          bnum3 = bool(100)
          print(type(bnum3), bnum3)
          bnum4 = bool(0)
          print(type(bnum4), bnum4)

<class 'bool'> True
<class 'bool'> True
<class 'bool'> True
<class 'bool'> False
```

← bool型態指定用 True 或者 False

## 注意：

將 None、False、0、0.0、0j(複數)、""(空字串)、()(空 Tuple)、[] (空清單)、{}(空字典)等傳給 bool()，都會傳回 False，這些型態的其他值傳入bool()則都會傳回 True。

# 複數型態

- 型態為 complex
- 撰寫時使用  $a + bj$  的形式

```
In [35]: cnum1 = 3+3j
          cnum2 = 2+7j
          cnum = cnum1+cnum2
          print(type(cnum),cnum)
          print('cnum1*cnum2 =',cnum1*cnum2)
```

```
<class 'complex'> (5+10j)
cnum1*cnum2 = (-15+27j)
```

# 字串型態

- 可以使用 ' ' 或 " " 包括文字
- Python 3 之後的版本都是產生 str 實例
- 多數 Python 開發者的習慣是使用單引號

```
In [39]: mystr = 'my name is PPAP'  
print(type(mystr))  
print(mystr)
```

```
<class 'str'>  
my name is PPAP
```

也可以利用str()將數值轉成字串

```
In [57]: mystr = str(9.487)  
print(type(mystr),mystr)
```

```
<class 'str'> 9.487
```



# 跳脫字元

符號	說明
\\	反斜線。
\'	單引號，當你使用''來表示字串，又要表示單引號時使用，例如'Justin\'s Website'。
\"	雙引號，當你使用""來表示字串，又要表示單引號時使用，例如\"text\" is a string"。
\ooo	以 8 進位數字指定字元碼點( Code point )，最多三位數，例如'\101'表示字串'A'。
\xhh	以 16 進位數字指定字元碼點，至少兩位數，例如'\0x41'表示字串'A'。
\uhhhh	以 16 位元 16 進位值指定字元，例如'\u54C8\u56C9'表示'哈囉'。
\Uhhhhhhh	以 32 位元 16 進位值指定字元，例如'\U000054C8\U000056C9'表示'哈囉'。
\0	空字元，請別與空字串搞混，'\0'相當於'\x00'。
\n	換行。
\r	歸位。
\t	Tab。

# 跳脫字元範例

```
In [43]: mystr1 = 'my name is \\PPAP\\'
          print(mystr1)
          mystr2 = 'my name is \"PPAP\"'
          print(mystr2)
          mystr3 = 'my name \n is PPAP'
          print(mystr3)
          mystr4 = 'my name \t is PPAP'
          print(mystr4)
```

```
my name is \PPAP\
my name is "PPAP"
my name
  is PPAP
my name          is PPAP
```

# 三重引號

- 可以使用 ' ' 或 " " 表示字串時，不可以換行。
- 如果字串內容必須跨越數行，可以使用三重引號
- 在三重引號間輸入的任何內容，在最後的 字串會照單全收，像是包括換行、縮排等

```
In [52]: mystr = '''Hello! World!!  
          My name is PPAP'''  
print(mystr)
```

```
Hello! World!!  
    My name is PPAP
```

# 還記得C語言的怎麼格式化字串嗎？

printf() 函數的格式

```
printf("格式字串", 項目1, 項目2, ...);
```

我記得 C 的字串格式化是這樣使用，那Python的呢？



---

# print() 函式

---

- 在 python 中，print 是將字串顯示出來。
- print() 函式的顯示預設是會換行
- print() 有個 end 參數，在指定的字串顯示之後，end 參數指定的字串就會輸出
- 預設的分隔符號是一個空白字元，如果想要指定其他字元的話，可以指定 sep 參數

# print()的一些範例

```
In [69]: myname = 'Ian'
print('Hello')
print(myname)
print('-----')
print('Hello',end='')
print(myname)
print('-----')
print('Hello, ',end='')
print(myname)
print('-----')
print('Hello',myname)
print('Hello',myname,sep=',')
```

```
Hello
Ian
-----
HelloIan
-----
Hello, Ian
-----
Hello Ian
Hello,Ian
```

# Python 格式化字串

- 目前的Python 3 支援兩種格式化方式，我們這邊只提新式的。

佔位符號用{}，若當中沒有數字或者名稱，format()方法就要依序指定對應之值

```
In [74]: mystr1 = 'my name is {} {}'.format('Yi-Ren', 'Yeh')
          print(mystr1)
          mystr2 = 'my name is {2} {3}'.format('Yi-Ren', 'Yeh', 'Lady', 'Gaga')
          print(mystr2)
```

```
my name is Yi-Ren Yeh
my name is Lady Gaga
```

可在{}指定對應之順序，從索引0開始

# Python 格式化字串-控制符號

在{}中用:來加入控制符號

```
In [80]: mystr1 = 'my number is {0:3.5f}'.format(95.27,100)
          print(mystr1)
          mystr2 = 'my numbers are {f:3.5f} and {n:3d}'.format(f = 95.27,n = 100)
          print(mystr2)
```

可以直接用變數來指定所要的值

my number is 95.27000  
my numbers are 95.27000 and 100



# 格式化控制符號表

符號	說明
<code>%%</code>	因為 <code>%</code> 符號已經被用來作為控制符號前置，所以規定使用 <code>%%</code> 才能在字串中表示 <code>%</code> 。
<code>%d</code>	10 進位整數。
<code>%f</code>	10 進位浮點數。
<code>%g</code>	10 進位整數或浮點數。
<code>%e, %E</code>	以科學記號浮點數格式化， <code>%e</code> 表示輸出小寫表示，如 <code>2.13 e+12</code> ， <code>%E</code> 表示大寫表示。
<code>%o</code>	8 進位整數。
<code>%x, %X</code>	以 16 進位整數格式化， <code>%x</code> 表示字母輸出以小寫表示， <code>%X</code> 則以大寫表示。
<code>%s</code>	字串格式符號。
<code>%r</code>	以 <code>repr()</code> 函式取得的結果輸出字串，本章稍後會談到 <code>repr()</code> 。

# input函數

- `input([prompt])`函數可接受一個字串作為參數，該字串會在作為問題向使用者查詢，並且在`stdout`上顯示。

```
In [40]: x = input('Your input: ')  
         print(type(x),x)
```

```
Your input: 12  
<class 'str'> 12
```

```
In [41]: x = input('Your input: ')  
         print(type(x),x)
```

```
Your input: Yeh  
<class 'str'> Yeh
```

---

# 群集型態

---

- 撰寫程式時，會需要不同的資料結構來收集資料，像是
  - 有序的清單、不重複的集合、鍵值對應的字典等。
- 在Python中，常用的資料結構在與語法上有直接的支援，像是
  - 清單 (list)
  - 集合 (set)
  - 字典 (dict)
  - Tuple (tuple)

# 清單 (list)

- 型態是 list
- 特性為有序、具備索引，內容與長度可以變動
- 要建立串列，可以使用 [] 實字，串列中每個元素，使用逗號「,」區隔。

```
In [1]: num = [9,5,2,7]
        print(num)
        print(type(num))
```

```
[9, 5, 2, 7]
<class 'list'>
```

list 的索引值從0開始

```
In [10]: num = [9,5,2,7]
         print(num[0])
         print(num[3])
```

```
9
7
```

# 清單 (list)

- 可以使用 [] 建立長度為0的list
- list這個類別也支援許多method
  - list.append(): 追加成員
  - list.remove(): 刪除成員
  - list.extend(L): 像串列中追加另一個串列 L
  - list.index(x): 獲得 x 在串列中的位置
  - 還有很多...

```
In [23]: mynum = []  
          print(mynum)  
          mynum.append(9)  
          print(mynum)  
          mynum.extend([5, 2, 7, 3])  
          print(mynum)  
          mynum.remove(3)  
          print(mynum)  
          print(mynum.index(7))  
  
[]  
[9]  
[9, 5, 2, 7, 3]  
[9, 5, 2, 7]  
3
```

Google “python list method” 來找出更多的method

# 集合 (set)

- 無序、元素不重複
- 可以使用{}包括元素，元素間使用「,」區隔，這會建立 set 實例

只留下不重複的元素

```
In [29]: myset = {1, 'hello', 3.2, 1}
print(myset)
print(type(myset))

{1, 3.2, 'hello'}
<class 'set'>
```

想建立空集合，必須使用 set()

```
In [35]: myset = set()
print(myset)
myset.add('hello')
myset.add(9527)
print(myset)
myset.remove(9527)
print(myset)

set()
{9527, 'hello'}
{'hello'}
```

# 集合 (set)

- 因為集合必須保證內容不重複，所以並非任何元素，都能放到集合
  - 像是 list 跟 set 就不行

```
In [37]: myset={ [1,2,3] }
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-37-642086aba3cd> in <module>()  
----> 1 myset={ [1,2,3] }  
  
TypeError: unhashable type: 'list'
```

list 跟 set 都是 unhashable 的型態，所以不能當成集合中的一個元素

# 集合 (set)

- 從其他可迭代的物件中建立 set，像是 字串、list 或 Tuple 等，可以使用 `set()`。
- 亦即把字串或者list等物件之內容拆解成集合的元素。

```
In [38]: myset=set([1,2,3])  
         print(myset)
```

```
{1, 2, 3}
```

```
In [39]: myset=set([1,2,3,3,2,2,2,1])  
         print(myset)
```

```
{1, 2, 3}
```

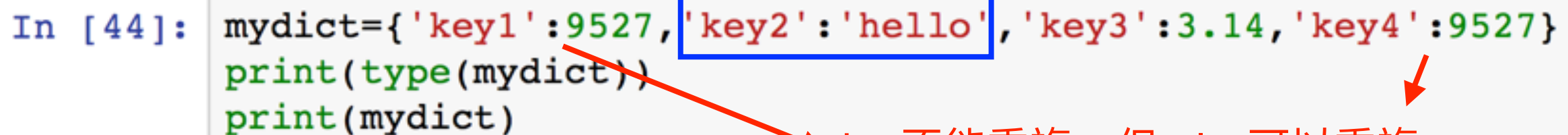


# 字典(dict)

- 儲存兩兩對應的鍵與值，為 dict 型態
- dict 中的鍵不重複，必須是 hashable

配對方式為 **key : value**

```
In [44]: mydict={'key1':9527,'key2':'hello','key3':3.14,'key4':9527}
          print(type(mydict))
          print(mydict)
```



```
<class 'dict'>
{'key2': 'hello', 'key3': 3.14, 'key1': 9527, 'key4': 9527}
```

```
In [46]: mydict={'key1':9527,'key2':'hello','key3':3.14,'key4':9527}
          print(mydict['key2'])
```

```
hello
```

可用 `[]` 取得 key 所對應的 value

# 字典(dict)

- 直接使用[]指定鍵要取得值時，若 dict 中 並沒有該鍵的存在，會發生 KeyError

```
In [52]: print(mydict['key5'])
```

```
-----  
KeyError                                Traceback (most recent call last)  
<ipython-input-52-ef0a2d47e76d> in <module>()  
----> 1 print(mydict['key5'])  
  
KeyError: 'key5'
```

- 可以用 .get() 測試是否存在此 key

```
In [51]: print(mydict.get('key1'))  
         print(mydict.get('key5'))
```

```
9527  
None
```

# 字典(dict)

- 也可以使用 dict() 來建立字典

```
In [53]: mydict = dict(key1=9527, key2='hello', key3=3.14)
          print(mydict)
```

使用dict()

```
{'key2': 'hello', 'key3': 3.14, 'key1': 9527}
```

```
In [44]: mydict={'key1':9527, 'key2':'hello', 'key3':3.14, 'key4':9527}
          print(type(mydict))
          print(mydict)
```

使用{}

```
<class 'dict'>
{'key2': 'hello', 'key3': 3.14, 'key1': 9527, 'key4': 9527}
```

---

# 字典(dict)

---

- `dict.items()`: 列出每一對 key 與 value
- `dict.keys()`: 列出每一個 key
- `dict.values()`: 列出每一個 value

```
In [64]: mydict = dict(key1=9527, key2='hello', key3=3.14)
          print(mydict.items())
          print(mydict.keys())
          print(mydict.values())
```

```
dict_items([('key2', 'hello'), ('key3', 3.14), ('key1', 9527)])
dict_keys(['key2', 'key3', 'key1'])
dict_values(['hello', 3.14, 9527])
```

# Tuple (tuple)

- 跟 list 很像
- Tuple 建立之後，就不能變動了
- 建立 tuple 的方式為在某個值後面加一個逗號「,」就可以
- 最後一個逗號可以省略
- 通常會配合小括號一起用，比較好辨識這是一個tuple

```
In [62]: mytup1 = 10,  
          print(type(mytup1),mytup1)  
          mytup2 = (10,'hello',3.14)  
          print(type(mytup2),mytup2)  
  
<class 'tuple'> (10,)  
<class 'tuple'> (10, 'hello', 3.14)
```

---

# Tuple (tuple)

---

- 可以將 Tuple 中的元素拆解(Unpack)

```
In [65]: mytup = (10, 'hello', 3.14)
          myint, mystr, myfloat = mytup
          print(myint)
          print(mystr)
          print(myfloat)
```

```
10
hello
3.14
```

# Python 中的變數

- 變數始終是個參考至實際物件的名稱，指定運算只是改變了變數的參考對象

```
In [67]: x=9527
          y=x
          print(id(x),id(y))
          y=2.0
          print(id(x),id(y))

4364789712 4364789712
4364789712 4366012584
```

← id()可以用來查詢變數之記憶體位址

```
In [68]: x=[1,2,3]
          y=x
          print(y)
          x[2]=9527
          print(y)

[1, 2, 3]
[1, 2, 9527]
```

# Python 中的變數-list之例子

- 用[]串接兩 list，實際上會產生新的 list，然後 將原有的兩個 list 中之元素參考，複製至新產生的 list 上

```
In [14]: num1 = [1,2]
          num2 = [3,4]
          num_plus = [num1,num2]
          print(num_plus)
          num1[0],num1[1] = 'GG','QQ'
          print(num_plus)
```

```
[[1, 2], [3, 4]]
[['GG', 'QQ'], [3, 4]]
```