SCU LawTech

Class #02

LAWTECH
learn coding get outstanding

次數	Date	Content	Column
1	09/26/2020	課程目標介紹、安裝**Anaconda**、基本**Python**語法介紹	基本語法
	10/03/2020	中秋節連假	
	10/10/2020	國慶日連假	
2	10/17/2020	**Python** 基本語法	基本語法
3	10/24/2020	**Python** 基本語法	基本語法
4		爬蟲介紹	爬蟲
5	10/31/2020	爬蟲實作	爬蟲
	11/07/2020	休息	
	11/14/2020	期中考	
6	11/21/2020	爬蟲實作	爬蟲
7	11/28/2020	資料清整、應用	資料清理
8	12/05/2020	資料清整、應用（統計圖表）	資料清理
9		機器學習概念	機器學習
10	12/12/2020	機器學習實際演練	機器學習
11	12/19/2020	機器學習實際演練	機器學習
	12/26/2020	聖誕連假	
	01/02/2021	元旦放假	
	01/09/2021	休息	
	01/16/2021	期末考	

Content

- 簡單回顧
- 今日課程
- 作業

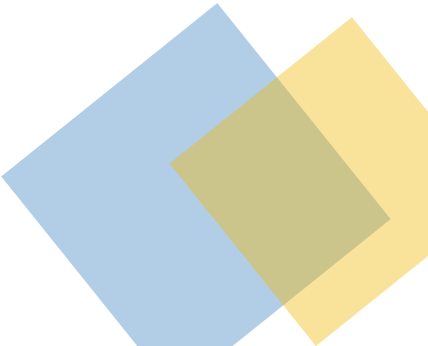
簡単回顧

變數

- 變數名稱只能包括：大小寫字母、中文、數字、底線
- 命名時請避開「系統保留字」或「函數名稱」

- 物件包含三個要素

ID	Type	Value
唯一識別	物件型態	物件的值



print( )

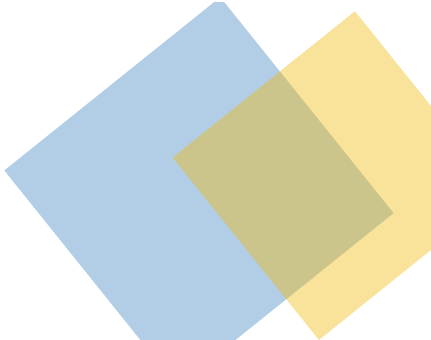
```
print(value, ..., sep=' ', end='\n')
```

```
print('a')
print('b', end='.')
print('c', end='/')
```

```
a
b.c/
```

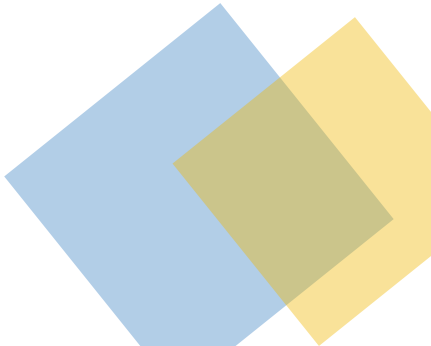
```
print('a', 'b', 'c')
print('a', 'b', 'c', sep='/')
```

```
a b c
a/b/c
```



資料型態

- 數值型態：int、float
- 字串型態：string
- 邏輯型態：bool
- 容器型態：list、dict、set



型態轉換

- int( )：轉為整數
- float( )：轉為浮點數
- str( )：轉為字串

```
t = 23.44555
type(t)
```

```
float
```

```
q = int(t)
q
```

```
23
```

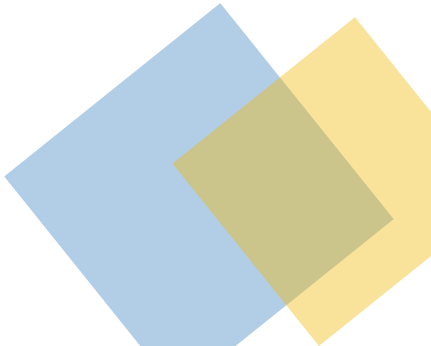
```
type(q)
```

```
int
```

eval( )

```
eval(str)
```

- 計算字串中有效的表達式，並返回結果
- 將字串轉成相對應的型態
- 將被轉換為字串的變數，反轉回變數



數值型態 - 運算符

```
#數學運算
print(32 + 45) #加
print(12 - 8) #減
print(12 * 2) #乘
print(12 ** 2) #指數
```

```
77
4
24
144
```

```
print(18 / 4) #除

print(18 // 4) #除取商
print(18 % 4) #除取餘
```

```
4.5
4
2
```

取商數時，在負值要小心!!

數值型態 – 比大小

- 大小比較：== 、 > 、< 、>= 、<= 、!=

```
In [32]: 2 > 3

Out[32]: False

In [33]: 2 < 3

Out[33]: True

In [34]: 2 == 3

Out[34]: False
```

浮點數的精確度問題

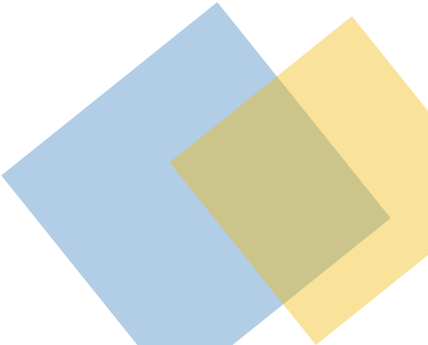
- 電腦是二進位制
- 實數的無限精度跟有限記憶體之間的矛盾

```
In [4]: 1/10
Out[4]: 0.1
```

一般來說，不建議直接進行比較

```
In [1]: 23.4 + 434.2
Out[1]: 457.59999999999997
```

```
In [5]: 0.1*3 == 0.3
Out[5]: False
```



字串型態

- 可以是「單引號」、「雙引號」或是「三個雙引號」

```
str1 = 'apple'
str2 = "apple"
str3 = """ Hello
world"""
```

字串型態

```
str1 + str2
```

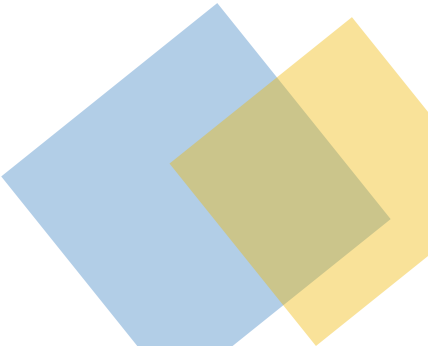
```
'appleballon'
```

```
str1 * 3
```

```
'appleappleapple'
```

```
len(str1)
```

```
5
```



字串型態

- 找位置（index）相關的，皆使用 [ ]
- Index 從 0 開始，取頭不取尾


```
str1 = 'apple'
```

```
str1[0]
```

```
str1[-1]
```

```
str1[1:-2]
```

```
str1[::3]
```

字串型態

【 Notes 】 `str [ start : end : step ]`

- start：字符的起始位置
- end：字符的結尾位置
- step：字符的間隔（步長）

```
str2 = 'hello, how are you?'
```

```
str2[1:9]
```

```
'ello, ho'
```

```
str2[1:9:2]
```

```
'el,h'
```

今日課程

字串型態

【 Notes 】 `str.split(分割符, 最大分割次數)`

- 分割符不會出現在output上，他只是切分的依據

```
# 切割字串
str3 = 'hello, how are you?'
str3.split('o')
```

```
['hell', ', h', 'w are y', 'u?']
```

```
# 切割字串
str3 = 'hello, how are you?'
str3.split('o', maxsplit=1)
    #maxsplit：可以指定分割次數
```

```
['hell', ', how are you?']
```

字串型態

【 Notes 】 `str.replace(被替代, 替代, 替代次數)`

```
# 取代
str3 = 'hello, how are you?'
str3.replace('l', 'i') #次數沒指定，預設全部取代
```

```
'heiio, how are you?'
```

字串型態

- 字串格式化：把字串中的變數替換成變數值

str.format( )

```
# 字串格式化（基本）
text = 'world'

print('hello {}'.format(text))
```

```
hello world
```

f-string

```
a = 123
print(f"測試{a}")
```

```
測試123
```

邏輯型態

- 邏輯運算符：is、not、and、or

AND	T	F
T	T	F
F	F	F

OR	T	F
T	T	T
F	T	F

```
print(True and False)
```

```
False
```

```
print(True or False)
```

```
True
```

邏輯型態

- 運算子的優先順序

分類	運算子
次方	**
乘法	*、/、//、%
加減	+、-
關係（大小）	==、!=、<、>、>=、<=
邏輯	not
邏輯	and
邏輯	or

容器型態

- 小括號：tuple (元組)
- 中括號：list (列表、串列)
- 大括號：dict (字典)
  set (集合)

```
Tuple = (1,2,3)
List = [1,2,3,4]
Dictionary = {'key':'value'}
Set = {1,2,3,4,5,6}
```

容器型態 - tuple

- 可放不同型態
- 不可變動
- 有序

```
1, 'Bob', 32
```

```
(1, 'Bob', 32)
```

容器型態 - tuple

```
test = (1, 'Bob', 32)
```

```
# 取值
test[2]
```

```
32
```

```
len(test)
```

```
3
```

```
(1, 'Bob', 32) + (2, 'Bobb', 322)
```

```
(1, 'Bob', 32, 2, 'Bobb', 322)
```

```
(1, 'Bob', 32)*2
```

```
(1, 'Bob', 32, 1, 'Bob', 32)
```

容器型態 - list

- 可放不同型態
- 可以變動
- 有序

```
a = [1, 2, 3, 4, 5]
a
```

```
[1, 2, 3, 4, 5]
```

容器型態 - list

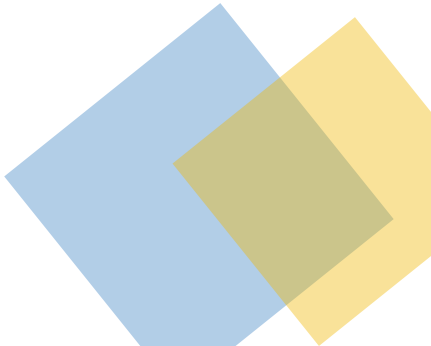
```
# 相加、相乘
a = [1,3,3]
b = [23,4,5]
```

```
a + b
```

```
[1, 3, 3, 23, 4, 5]
```

```
a*3
```

```
[1, 3, 3, 1, 3, 3, 1, 3, 3]
```



容器型態 - list

- 在最後面增加元素

```
x = [1, 23, 34]
```

```
a.append(x)
a
```

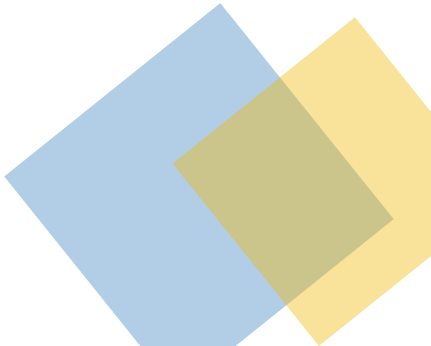
```
[1, 2, 3, 4, 5, [1, 23, 34]]
```

```
a.extend(x) #拆開
a
```

```
[1, 2, 3, 4, 5, 1, 23, 34]
```

練習

- 請用 3 種不同的方法，生成一個 [1,2,"Bob", [1,2]]



容器型態 - list

- 指定位置插入

```
list.insert(index, 要插入的值)
```

```
a.insert(0, 23)
a
```

```
[23, 1, 2, 3, 4, 5]
```

容器型態 - list

- 提取：取出，並在 list 中將他刪除
- 預設，從最尾端

```
list.pop(index)
```

```
a.pop(0)
```

```
1
```

```
a
```

```
[2, 3, 4, 5]
```

容器型態 - list

- 刪除，遇到的第一筆

```
list.remove(要刪除的值)
```

```
a.remove(2)
```

```
a
```

```
[1, 3, 4, 5]
```

容器型態 - list

- 找出 x 的第一個 index (位置)

```
list.index(要尋找的值)
```

```
a.index(2)
```

```
1
```

容器型態 - list

- 找出 x 的第一個 index (位置)

```
list.count(要計算次數的值)
```

```
a.count(1)
```

```
2
```

容器型態 - list

- 排序

`list.sort()` ：排序

`list.reverse()` ：反轉

```
b = [1,75,3,65,19]
```

```
b.sort()
b
```

```
[1, 3, 19, 65, 75]
```

```
b.reverse()
b
```

```
[75, 65, 19, 3, 1]
```

Python 內建排序

`sorted()`

- 不會動到原始資料

```
print(sorted(a))
print(a)
```

```
[1, 1, 2, 3, 4, 5, 23, 34]
[1, 2, 3, 4, 5, 1, 23, 34]
```

容器型態 - list

- 多維 list

```
#多維list
a = ['hey', '3', 'hi', [7, ['hello', 5, '9'], 9]]
len(a)
```

```
4
```

```
a[3][0]
```

```
7
```

容器型態 - dict

dict：字典

- 映射類型：{'key':'value'}，一個key僅能對應到一個value
- key 必需是不可變動的

```
#宣告一個dict變數
dict1 = {}
type(dict1)
```

```
dict
```

【Notes】

`{ }`：在查詢時，速度會比list快上很多

因為使用hash的技術，可以直接對應到我們要找的值

容器型態 - dict

```
dict1 = {'A':1, 'B':[1,2,3]}
dict1
```

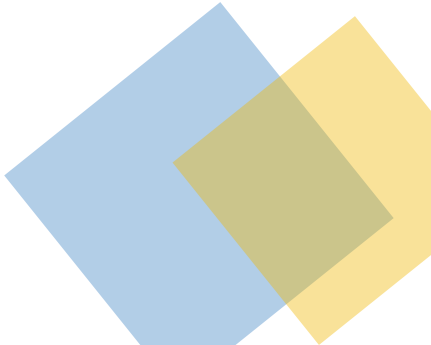
```
{'A': 1, 'B': [1, 2, 3]}
```

```
dict1['C'] = [5,6,7] #新增
dict1
```

```
{'A': 1, 'B': [1, 2, 3], 'C': [5, 6, 7]}
```

```
dict1['B'] = [7,8,9] #更改
dict1
```

```
{'A': 1, 'B': [7, 8, 9], 'C': [5, 6, 7]}
```



容器型態 - dict

```
# 所有的key（鍵）
dict1.keys()
```

```
dict_keys(['A', 'B', 'C'])
```

```
# 所有的values（值）
dict1.values()
```

```
dict_values([1, [7, 8, 9], [5, 6, 7]])
```

```
# 所有 key&values（鍵值對）
dict1.items()
```

```
dict_items([('A', 1), ('B', [7, 8, 9]), ('C', [5, 6, 7])])
```

容器型態 - dict

- 刪除：指定 key 刪除，其相對應的 value 也會隨之刪除

```
dict1.pop('A')
```

```
1
```

```
dict1
```

```
{'B': [7, 8, 9], 'C': [5, 6, 7]}
```

容器型態 - dict

- 判斷 key 是否存在

```
"A" in dict1
```

```
True
```

```
dict.get(key值, 若key值不存在的回傳值)
```

```
dict1.get('A')
```

```
1
```

```
dict1.get('1', "NO")
```

```
'NO'
```

list v.s. dict

list

- 查詢跟插入的時間會隨著元素的增加而增加
- 佔用空間小，浪費的記憶體少

dict

- 查詢跟插入的速度快，不會隨著 key 增加而變慢
- 需要佔用大量的記憶體

若直接對 dict 做操作，操作對象是 key

容器型態 - set

- 集合，無序、不重複的元素集
- 像是只有 key 的 dict

```
Set = set()
type(Set)
```

```
SET = {1,2,3,4}
type(SET)
```

```
set
```

容器型態 - set

```
# List轉set
a = [1, 2, 2, 1, 4, 2, 3]
setA = set(a)
setA
```

```
{1, 2, 3, 4}
```

```
# 增加元素
SET.add(34)
SET
```

```
{1, 2, 3, 4, 34}
```

```
# 刪除元素
SET.remove(2)
SET
```

```
{1, 3, 4, 34}
```

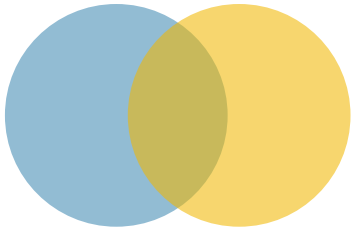
```
# 判斷元素是否存在於集合中
1  not in SET
```

```
False
```

容器型態 - set

```
# 聯集
setA = {1, 2, 3, 4}
setB = {2, 3, 4}
setA | setB
```

```
{1, 2, 3, 4}
```



```
# 交集
setA & setB
```

```
{2, 3, 4}
```

```
# 插集（注意！順序）
setA - setB
```

```
{1}
```

```
setB - setA
```

```
set()
```

THANKS

LAWTECH
learn coding get outstanding