Python Programming

2023 Spring; week 12

Instructor: Cheng-Chun Chang (張正春)
Department of Electrical Engineering

Textbook: Python程式設計：從入門到進階應用(第三版) 2020

課程助教

協助dubug，禁止同學看code照抄

第一排:

第二排:

第三排:

第四排:

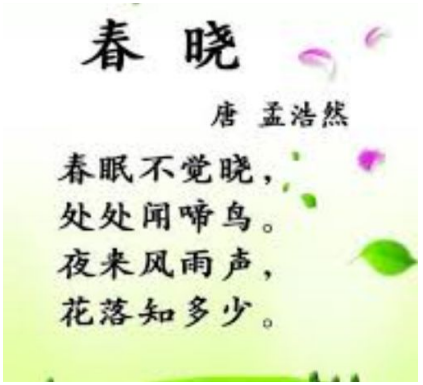
第五排:

第六排:

打游擊:

Programming: it's all about format …analogous to 唐詩宋詞

□低頭吃便當….



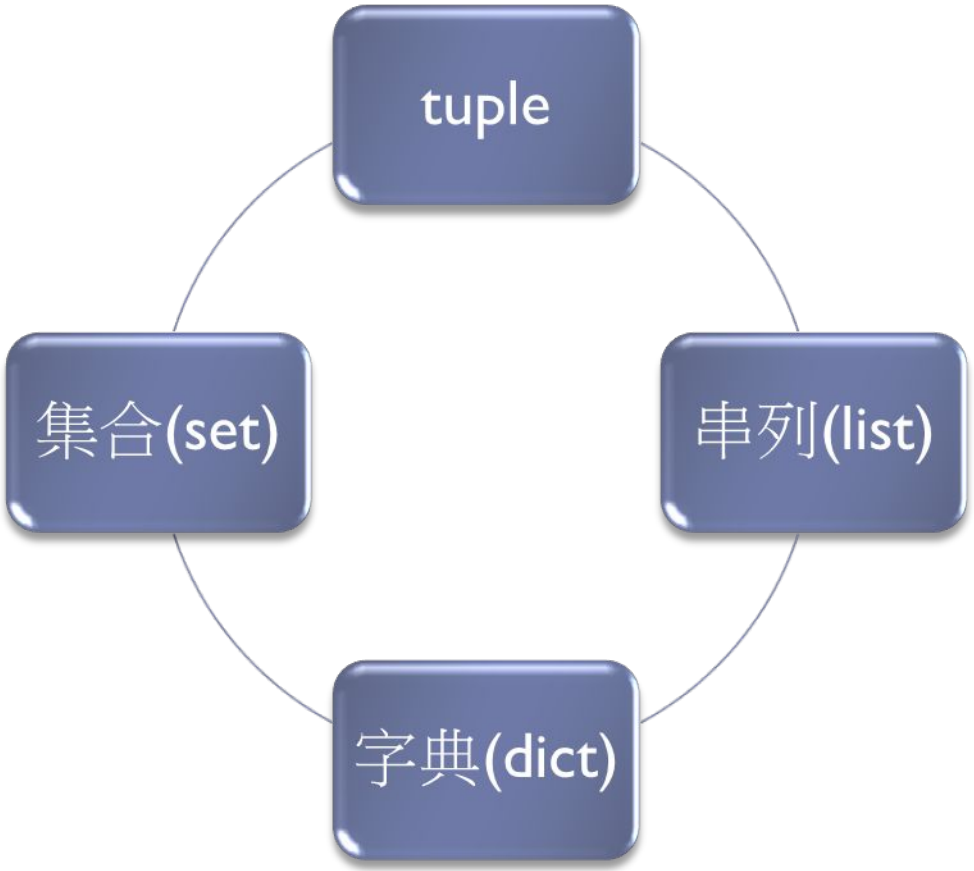
□處處蚊子咬….

KEY: 1) 用法 2) 用法 3)用法, and then you can modify it.

Ch3 資料儲存容器tuple- 串列- 字典-集合

Ch3 資料儲存容器

- Python 的資料儲存容器，可以分為



Ch3 資料儲存容器

表 3-1　Python 的資料儲存容器說明

Python 的資料儲存容器	說明
tuple	tuple 用於依序儲存資料，可以依照順序取出資料，但不能更改，是不可變的物件
串列 (list)	串列 (list) 也是用於依序儲存資料，可以依照順序取出，也可以更改。
字典 (dict)	字典 (dict) 儲存的資料爲「鍵 (key)」與「值 (value)」對應的資料，使用「鍵」查詢「值」。 取出字典所有資料後，發現與建構字典時輸入資料的順序不同，字典儲存資料是沒有順序性的，字典也可視爲關聯性陣列 (associative array)。
集合 (set)	集合 (set) 儲存沒有順序性的資料，要找出資料是否存在，儲存不需要鍵與值對應的資料，就很適合使用集合。

Programming codes I

3-1 tuple

- **範例3-1-1**：ch3\3-1-tuple1.py

使用「( )」建立 tuple。

程式	執行結果
t1 = () print(t1)	()

也可以使用「,」串接資料形成 tuple。

程式	執行結果
t2 = 1, 2 ,3 print(t2)	(1, 2, 3)

範例3-1-1：ch3\3-1-tuple1.py

只使用「,」串接資料形成 tuple，但這樣不是很明確，一般而言會再加上 () 表示是 tuple。

程式	執行結果
t3 = (1, 2, 3) print(t3)	(1, 2, 3)

可以在 tuple 使用「[ ]」取出個別元素。

程式	執行結果
t3 = (1, 2, 3) print(t3[0])	1

範例3-1-1:ch3\3-1-tuple1.py

我們可以使用變數取出 tuple 中的元素，稱作 unpacking(開箱)。

程式	執行結果
t3 = (1, 2, 3) a, b, c= t3 print('a=', a, ',b=', b, ',c=', c)	a= 1 ,b= 2 ,c= 3

可以使用 tuple 交換兩數。

程式	執行結果
a = 10 b = 20 print(' 交換前 ','a=', a, ',b=', b) a, b = b, a print(' 交換後 ','a=', a, ',b=', b)	交換前 a= 10 ,b= 20 交換後 a= 20 ,b= 10

範例3-1-1:ch3\3-1-tuple1.py

可以使用函式 tuple 將串列轉換成 tuple，串列將於下一個單元介紹。

程式	執行結果
list1=[1,2,3,4] t4 = tuple(list1) print(t4)	(1, 2, 3, 4)

範例3-1-1:ch3\3-1-tuple1.py

tuple 中元素可以是 tuple，內部的 tuple 會被視爲一個元素，存取內部 tuple 需要使用兩層中括號 [] 進行存取。

程式	執行結果
t4 = (1,2,3,4) t5 = (t4,5,6) print(t5) print(len(t5)) print(t5[0][0])	 ((1, 2, 3, 4), 5, 6) 3 1

範例3-1-1:ch3\3-1-tuple1.py

若只有一個元素的 tuple 需在元素後面加上逗點「,」，沒有加上逗點「,」就不是 tuple。

程式	執行結果
t6 = ('z',) print(t6)	('z',)

Programming codes 2

3-2 串列(list)

- 串列為可修改的序列資料，可以修改元素資料、新增、刪除、插入與取出元素
  - 使用list 函式可以將資料轉換成串列，並可以使用[::] 取出串列的一部分

3-2-1 新增與修改串列

1. 使用「[]」建立新的串列。

程式	執行結果
shoplist = ['牛奶', '蛋', '咖啡豆', '西瓜', '鳳梨'] print('購物清單 shoplist 爲') print(shoplist)	購物清單 shoplist 爲 ['牛奶', '蛋', '咖啡豆', '西瓜', '鳳梨']

2. 使用「[索引值]」讀取個別元素。

程式	執行結果
shoplist = ['牛奶', '蛋', '咖啡豆', '西瓜', '鳳梨'] print('顯示 shoplist[0] 爲 ',shoplist[0])	顯示 shoplist[0] 爲 牛奶

ch3\3-2-1-list1.py

3. 使用「len 函式」讀取串列長度。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] print(' 購物清單 shoplist 的長度爲 ', len(shoplist))	購物清單 shoplist 的長度爲 5

4. 使用「串列 [索引值]= 元素值」修改個別元素。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] shoplist[1] = ' 皮蛋 ' print(" 執行 shoplist[1] = ' 皮蛋 ' 後 ") print(shoplist)	執行 shoplist[1] = ' 皮蛋 ' 後 [' 牛奶 ', ' 皮蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 ']

ch3\3-2-1-list1.py

5. 使用「函式 index」取出指定元素的索引值。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] index=shoplist.index(' 咖啡豆 ') print(" 執行 index=shoplist.index(' 咖啡豆 ') 後 ") print('index=', index)	執行 index=shoplist.index(' 咖啡豆 ') 後 index= 2

6. 使用「函式 append」將元素增加到串列的最後。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] shoplist.append(' 麵包 ') print(" 執行 shoplist.append(' 麵包 ') 後 ") print(shoplist)	執行 shoplist.append(' 麵包 ') 後 [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 ', ' 麵包 ']

ch3\3-2-1-list1.py

7. 使用「函式 insert」將元素插入到串列的指定位置。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] shoplist.insert(4, ' 蘋果 ') print(" 執行 shoplist.insert(4, ' 蘋果 ') 後 ") print(shoplist)	執行 shoplist.insert(4, ' 蘋果 ') 後 [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 蘋果 ', ' 鳳梨 ']

8. 使用「函式 remove」將指定的元素從串列中移除。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] shoplist.remove(' 蛋 ') print(" 執行 shoplist.remove(' 蛋 ') 後 ") print(shoplist)	執行 shoplist.remove(' 蛋 ') 後 [' 牛奶 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 ']

ch3\3-2-1-list1.py

9. 使用「函式 del」將串列中第幾個元素刪除。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] del shoplist[0] print(" 執行 del shoplist[0] 後 ") print(shoplist)	執行 del shoplist[0] 後 [' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 ']

ch3\3-2-1-list1.py

10. 使用「函式 pop」將串列中第幾個元素刪除，若不指定元素則刪除最後一個元素。

程式	執行結果
shoplist = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] shoplist.pop(0) print(" 執行 shoplist.pop(0) 後 ") print(shoplist) shoplist.pop() print(" 執行 shoplist.pop() 後 ") print(shoplist) shoplist.pop(-1) print(" 執行 shoplist.pop(-1) 後 ") print(shoplist)	執行 shoplist.pop(0) 後 [' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 '] 執行 shoplist.pop() 後 [' 蛋 ', ' 咖啡豆 ', ' 西瓜 '] 執行 shoplist.pop(-1) 後 [' 蛋 ', ' 咖啡豆 ']

ch3\3-2-1-list1.py

11. 使用「函式 sort」排序串列元素。

程式	執行結果
shoplist = ['milk', 'egg', 'coffee', 'watermelon'] shoplist.sort() print(" 執行 shoplist.sort() 後 ") print(shoplist)	執行 shoplist.sort() 後 ['coffee', 'egg', 'milk', 'watermelon']

12. 串列以包含各種資料型別的元素。

程式	執行結果
list = [1,2.0,3,'Python'] print(" 串列可以包含各種資料型別的元素 ") print(list)	串列可以包含各種資料型別的元素 [1, 2.0, 3, 'Python']

ch3\3-2-1-list1.py

13. 使用「for 變數 in 串列」可以讀取串列所有元素到「變數」，將在第 5 章詳細介紹 for 迴圈的各種應用。

程式	執行結果
shoplist = ['milk', 'egg', 'coffee', 'watermelon'] for item in shoplist: print(item)	milk egg coffee watermelon

3-2-2 串接兩個串列

- ch3\3-2-2-list2.py
- 使用「+」串接兩個串列

程式	執行結果
shoplist1 = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 '] shoplist2 = [' 西瓜 ', ' 鳳梨 '] shoplist_all = shoplist1 + shoplist2 print(shoplist_all)	[' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 ']

3-2-3 產生串列

- ch3\3-2-3-list3.py

1. 使用「函式 list」產生串列，函式 list 可以輸入字串或 tuple。

程式	執行結果
list1 = list('python') print(list1) tuple2 = ('a', 'b', 1, 2) list2 = list(tuple2) print(list2)	['p', 'y', 't', 'h', 'o', 'n'] ['a', 'b', 1, 2]

2. 使用「函式 split」也會回傳串列。

程式	執行結果
list3 = "2016/1/1".split('/') print(list3)	['2016', '1', '1']

Programming codes 3

3-3 字典(dict)

- 字典(dict) 儲存的資料為「鍵(key)」與「值(value)」對應的資料
  - 使用「鍵」可以搜尋對應的「值」，取出字典的所有資料時，發現與建構字典時輸入資料的順序不同，字典儲存資料是沒有順序性的，字典中的「鍵」需使用不可以變的元素
  - 例如：數字、字串與tuple。字典可以新增、刪除、更新與合併兩個字典

3-3-1 新增與修改字典

- 使用「{}」建立新的字典, 字典以「鍵(key): 值(value)」表示一個元素

程式	執行結果
dict1={} print(dict1) lang={' 早安 ':'Good Morning', ' 你好 ':'Hello' } print(lang)	{} {' 你好 ': 'Hello', ' 早安 ': 'Good Morning'}

ch3\3-3-1-dict1.py

程式	執行結果
lang={' 早安 ':'Good Morning', ' 你好 ':'Hello'} print(' 「你好」的英文爲 ',lang[' 你好 '])	「你好」的英文爲 Hello

程式	執行結果
lang={' 早安 ':'Good Morning', ' 你好 ':'Hello'} print(' 「你好嗎」的英文爲 ',lang[' 你好嗎 '])	Traceback (most recent call last): File "G:\ch3\3-3-1-dict1.py", line 6, in <module> print(' 「你好嗎」的英文爲 ',lang[' 你好嗎 ']) KeyError: ' 你好嗎 '

程式	執行結果
lang={' 早安 ':'Good Morning', ' 你好 ':'Hello'} print(' 「你好」的英文爲 ',lang.get(' 你好 ')) print(' 「你好嗎」的英文爲 ',lang.get(' 你好嗎 ')) print(' 「你好嗎」的英文爲 ',lang.get(' 你好嗎 ',' 不在字典內 '))	「你好」的英文爲 Hello 「你好嗎」的英文爲 None 「你好嗎」的英文爲 不在字典內

ch3\3-3-1-dict1.py

程式	執行結果
lang={' 早安 ':'Good Morning', ' 你好 ':'Hello'} lang[' 你好 ']='Hi' print(lang) lang[' 學生 ']='Student' print(lang)	{' 早安 ': 'Good Morning', ' 你好 ': 'Hi'} {' 學生 ': 'Student', ' 早安 ': 'Good Morning', ' 你好 ': 'Hi'}

ch3\3-3-1-dict1.py

- 使用「del 字典['鍵']」會將字典中指定的「鍵」刪除, 所對應的「值」也會刪除

程式	執行結果
lang={' 早安 ':'Good Morning', ' 你好 ':'Hello'} del lang[' 早安 '] print(lang)	{' 你好 ': 'Hello'}

ch3\3-3-1-dict1.py

- 使用「函式clear」清空整個字典

程式	執行結果
lang={' 早安 ':'Good Morning', ' 你好 ':'Hello'} lang.clear() print(lang)	{}

ch3\3-3-2-dict2.py

程式	執行結果
a=[[' 早安 ','Good Morning'],[' 你好 ','Hello']] dict1=dict(a) print(dict1) b=[(' 早安 ','Good Morning'),(' 你好 ','Hello')] dict2=dict(b) print(dict2) c=([' 早安 ','Good Morning'],[' 你好 ','Hello']) dict3=dict(c) print(dict3) d=((' 早安 ','Good Morning'),(' 你好 ','Hello')) dict4=dict(d) print(dict4)	{' 早安 ': 'Good Morning', ' 你好 ': 'Hello'} {' 早安 ': 'Good Morning', ' 你好 ': 'Hello'} {' 早安 ': 'Good Morning', ' 你好 ': 'Hello'} {' 早安 ': 'Good Morning', ' 你好 ': 'Hello'}

程式	執行結果
lang1={' 你好 ':'Hello'} lang2={' 學生 ':'Student'} lang1.update(lang2) print(lang1) lang1={' 早安 ':'Good Morning',' 你好 ':'Hello'} lang2={' 你好 ':'Hi'} lang1.update(lang2) print(lang1)	{' 學生 ': 'Student', ' 你好 ': 'Hello'} {' 早安 ': 'Good Morning', ' 你好 ': 'Hi'}

ch3\3-3-4-dict4.py

程式	執行結果
lang1={' 早安 ':'Good Morning',' 你好 ':'Hello'} lang2 = lang1 lang2[' 你好 ']='Hi' print('lang1 爲 ', lang1) print('lang2 爲 ', lang2) lang1={' 早安 ':'Good Morning',' 你好 ':'Hello'} lang3 = lang1.copy() lang3[' 你好 ']='Hi' print('lang1 爲 ', lang1) print('lang3 爲 ', lang3)	lang1 爲 {' 你好 ': 'Hi', ' 早安 ': 'Good Morning'} lang2 爲 {' 你好 ': 'Hi', ' 早安 ': 'Good Morning'} lang1 爲 {' 你好 ': 'Hello', ' 早安 ': 'Good Morning'} lang3 爲 {' 你好 ': 'Hi', ' 早安 ': 'Good Morning'}

Note:

- dict2=dict1.copy(), 會複製dict1 到dict2, dict1 與dict2 指向不同的字典物件, 若更改字典dict1 的元素, 並不會修改字典dict2
- 若使用「dict2=dict1」, 則dict1 與dict2 指向同一個字典物件, 修改字典dict1 的元素, 字典dict2 也會更改

ch3\3-3-5-dict5.py

程式	執行結果

lang={' 早安 ':'Good Morning',' 你好 ':'Hello'}
for ch, en in lang.items():
  print(' 中文爲 ', ch, ' 英文爲 ', en)
for ch in lang.keys():
  print(ch,lang[ch])
for en in lang.values():
  print(en)
	中文爲 你好 英文爲 Hello
	中文爲 早安 英文爲 Good Morning
	你好 Hello
	早安 Good Morning
	Hello
	Good Morning

3-3-5 使用「for」讀取字典每個元素

Note:

- 使用「for」讀取字典每個元素，配合字典的「函式items」會回傳「鍵」與「值」兩個元素
- 配合字典的「函式keys」會回傳「鍵」，而配合字典的「函式values」會回傳「值」

Programming codes 4

3-4 集合(set)

- 集合(set) 儲存沒有順序性的資料, 要找出資料是否存在, 集合內元素不能重複, 集合會自動刪除重複的元素

ch3\3-4-1-set1.py

- 使用「set()」或「{}」建立新的集合，集合會自動刪除重複的元素，「set()」只能使用一個參數，這個參數可以是字串、tuple、串列或字典都可以建立集合

程式	執行結果
s = {1,2,3,4} print(s) s = set(('a',1,'b',2)) print(s) s = set(['apple', 'banana', 'apple']) print(s) s = set({' 早安 ':'Good Morning', ' 你好 ':'Hello'}) print(s) s = set('racecar') print(s)	{1, 2, 3, 4} {1, 2, 'b', 'a'} {'apple', 'banana'} {' 早安 ', ' 你好 '} {'r', 'e', 'c', 'a'}

ch3\3-4-1-set1.py

- 使用「函式add」新增集合元素，使用「函式remove」刪除集合元素

程式	執行結果
s = set('tiger') print(s) s.add('z') print(s) s.remove('t') print(s)	{'g', 't', 'i', 'e', 'r'} {'g', 'i', 'z', 'e', 'r', 't'} {'g', 'i', 'z', 'e', 'r'}

ch3\3-4-2-set2.py

表 3-2　集合的運算說明

集合運算	說明	範例	結果
聯集 (|)	A|B 元素存在集合 A 或存在集合 B。	a = set('tiger') b = set('bear') print(a) print(b) a = set('tiger') b = set('bear') print(a | b)	{'r', 'i', 'g', 'e', 't'} {'r', 'a', 'e', 'b'} {'a', 'g', 'r', 'i', 'b', 'e', 't'}
交集 (&)	A&B 元素存在集合 A 且存在集合 B。	a = set('tiger') b = set('bear') print(a & b)	{'r', 'e'}

ch3\3-4-2-set2.py

集合運算	說明	範例	結果
差集 (-)	A-B 元素存在集合 A，但不存在集合 B。	a = set('tiger') b = set('bear') print(a - b)	{'t', 'i', 'g'}
互斥或 (^)	元素存在集合 A，但不存在集合 B， 或元素存在集合 B，但不存在集合 A。	a = set('tiger') b = set('bear') print(a ^ b)	{'i', 'a', 'b', 't', 'g'}

3-4-3 集合的比較

- ch3\3-4-3-set3.py
- 可以將任兩個集合進行下列比較運算
  - 子集合(<=)
  - 真子集合(<)
  - 超集合(>=)
  - 真超集合(>)

表 3-3　集合的比較

集合運算	說明	範例	結果
子集合 (<=) issubset()	A<=B 相等於 A.issubset(B) 存在集合 A 的每個元素，也一定存在於集合 B，則回傳 True，否則回傳 False。	a = set('tiger') b = set('tigers') print(a<=b)	True
眞子集合 (<)	A<B 存在集合 A 的每個元素，也一定存在於集合 B，且集合 B 至少有一個元素不存在於集合 A，則回傳 True，否則回傳 False。	a = set('tiger') b = set('tigers') print(a<b)	True
超集合 (>=) issuperset()	A>=B 相等於 A.issuperset(B) 存在集合 B 的每個元素，也一定存在於集合 A，則回傳 True，否則回傳 False。	a = set('tiger') b = set('tigers') print(a>=b)	False
眞超集合 (>)	A>B 存在集合 B 的每個元素，也一定存在於集合 A，且集合 A 至少有一個元素不存在於集合 B，則回傳 True，否則回傳 False。	a = set('tiger') b = set('tigers') print(a>b)	False

Readings

3-2-4 使用「[開始: 結束: 間隔]」存取串列

- ch3\3-2-4-list4.py
- 使用「[開始: 結束: 間隔]」切割字串, 從「開始」到「結束」(不包含結束的字元) 每隔「間隔」個字元取一個字元出來

ch3\3-2-4-list4.py

- list[:] 表示取串列list 的每一個元素，若沒有指定結束元素，預設使用最後一個元素結束，若沒有指定開始元素，預設使用第一個元素開始。

程式	執行結果
a = list('abcdefghijk') print('a[:] 爲 ', a[:])	a[:] 爲 ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k']

ch3\3-2-4-list4.py

- list[開始:] 表示取串列list[開始] 到串列list 結束的所有元素, 若沒有指定結束元素, 預設使用串列list 最後一個元素, 包含最後一個元素。
- list[: 結束] 表示取串列list 第一個元素到串列list[結束] 所指定元素的前1 個元素為止的所有元素, 若沒有指定開始元素, 預設使用串列list 第一個元素開始。

ch3\3-2-4-list4.py

程式	執行結果
a = list('abcdefghijk') print('a[:5] 爲 ', a[:5]) print('a[5:] 爲 ', a[5:]) print('a[:-5] 爲 ', a[:-5]) print('a[-5:] 爲 ', a[-5:])	a[:5] 爲 ['a', 'b', 'c', 'd', 'e'] a[5:] 爲 ['f', 'g', 'h', 'i', 'j', 'k'] a[:-5] 爲 ['a', 'b', 'c', 'd', 'e', 'f'] a[-5:] 爲 ['g', 'h', 'i', 'j', 'k']

ch3\3-2-4-list4.py

- list[開始: 結束] 表示取串列list[開始] 元素到串列list[結束] 所指定元素的前1個元素為止的所有元素。

程式	執行結果
a = list('abcdefghijk') print('a[0:4] 爲 ', a[0:4]) print('a[-5:-3] 爲 ', a[-5:-3])	a[0:4] 爲 ['a', 'b', 'c', 'd'] a[-5:-3] 爲 ['g', 'h']

ch3\3-2-4-list4.py

- list[開始: 結束: 間隔] 表示取串列list [開始] 元素到串列list[結束] 所指定元素的前1 個元素為止的所有元素，每隔「間隔」個元素取一個元素。

程式	執行結果
a = list('abcdefghijk') print('a[1:10:3] 爲 ', a[1:10:3]) print('a[-1:-4:-1] 爲 ', a[-1:-4:-1])	a[1:10:3] 爲 ['b', 'e', 'h'] a[-1:-4:-1] 爲 ['k', 'j', 'i']

ch3\3-2-4-list4.py

- list[::-1] 表示反轉串列list, 反轉串列為串列中第1 個元素與最後1 個元素互換
- 第2 個元素與倒數第2 個元素互換
- 第3 個元素與倒數第3 個元素互換
- 一直到只剩下一個元素或沒有元素可以互換為止。

程式	執行結果
a = list('abcdefghijk') print('a[::-1] 爲 ', a[::-1])	a[::-1] 爲 ['k', 'j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']

3-2-5 拷貝串列

- ch3\3-2-5-list5.py
- 使用 [:] 與函式copy 拷貝串列, 會將串列複製一份與原來串列不同, 是兩個不同的物件, 佔有不同的記憶體空間, 而使用等號=, 只是貼上變數名稱的標籤

ch3\3-2-5-list5.py

- 使用等號「=」, 例如「list1 = list2」, 表示list1 與list2 指向相同的物件, 當串列list1 或串列list2 中元素有修改, list1 與list2 都會改變。

程式	執行結果
list1 = [1, 2, 3, 4] list2 = list1 print('list1=', list1) print('list2=', list2) list1[2]=19 print('list1=', list1) print('list2=', list2) list2[2]=18 print('list1=', list1) print('list2=', list2)	 list1= [1, 2, 3, 4] list2= [1, 2, 3, 4] list1= [1, 2, 19, 4] list2= [1, 2, 19, 4] list1= [1, 2, 18, 4] list2= [1, 2, 18, 4]

ch3\3-2-5-list5.py

- 使用「[:]」與函式copy 拷貝串列，會將串列複製一份，是兩個不同的物件，占有不同的記憶體空間，修改時兩者不會互相影響。

程式	執行結果
list1 = [1, 2, 3, 4] list3 = list1[:] list3[2] = 19 print('list1=', list1) print('list3=', list3) list4 = list1.copy() list4[2] = 19 print('list1=', list1) print('list4=', list4)	list1= [1, 2, 3, 4] list3= [1, 2, 19, 4] list1= [1, 2, 3, 4] list4= [1, 2, 19, 4]

範例3-5-4 複雜的結構:ch3\3-5-4- 複雜的結構.py

tuple、串列、字典與字典都可以互相包含組成更大的結構，假設有以下兩個串列。

```
星期 = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
月份 = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September',
'October', 'November', 'December']
```

使用「week」對應到串列「星期」，使用「month」對應到串列「月份」，製作字典 dic，產生字典 dic 的程式，如下。

```
dic = {'week': 星期 , 'month': 月份 }
```

請寫出一個程式，從字典 dic 取出串列「月份」，從字典 dic 取出串列「月份」的「August」。

範例3-5-4 複雜的結構:ch3\3-5-4- 複雜的結構.py

- 解題想法
  - 綜合字典與串列的概念解題。
- 程式碼

```
1  星期 = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
2  月份 = ['January', 'February', 'March', 'April', 'May', 'June', \
3       'July', 'August', 'September', 'October', 'November', 'December']
4  dic = {'week': 星期 , 'month': 月份 }
5  print(dic['month'])
6  print(dic['month'][7])
```

範例3-5-3 找出一首詩的所有字:ch3\3-5-3- 找出一首詩的所有字.py

- 請設計一個程式找出一首詩的所有字, 本範例使用唐詩「春曉」, 作者「孟浩然」, 重複的字只顯示一個就可以
- 解題想法
  - 將詩儲存到集合(set) 結構內, 使用集合的功能完成程式。

範例3-5-3 找出一首詩的所有字:ch3\3-5-3- 找出一首詩的所有字.py

- 程式碼

```
1  詩 = '春眠不覺曉，處處聞啼鳥。夜來風雨聲，花落知多少。'
2  字 = set( 詩 )
3  字 .remove('，')
4  字 .remove('。')
5  print( 字 )
```

```
{'聲', '聞', '夜', '來', '不', '風', '知', '多', '少', '鳥', '眠',
'落', '曉', '啼', '雨', '花', '春', '處', '覺'}
```

範例3-5-4 複雜的結構:ch3\3-5-4- 複雜的結構.py

- 執行結果

```
['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December']
August
```

Exercise

3-5 範例練習

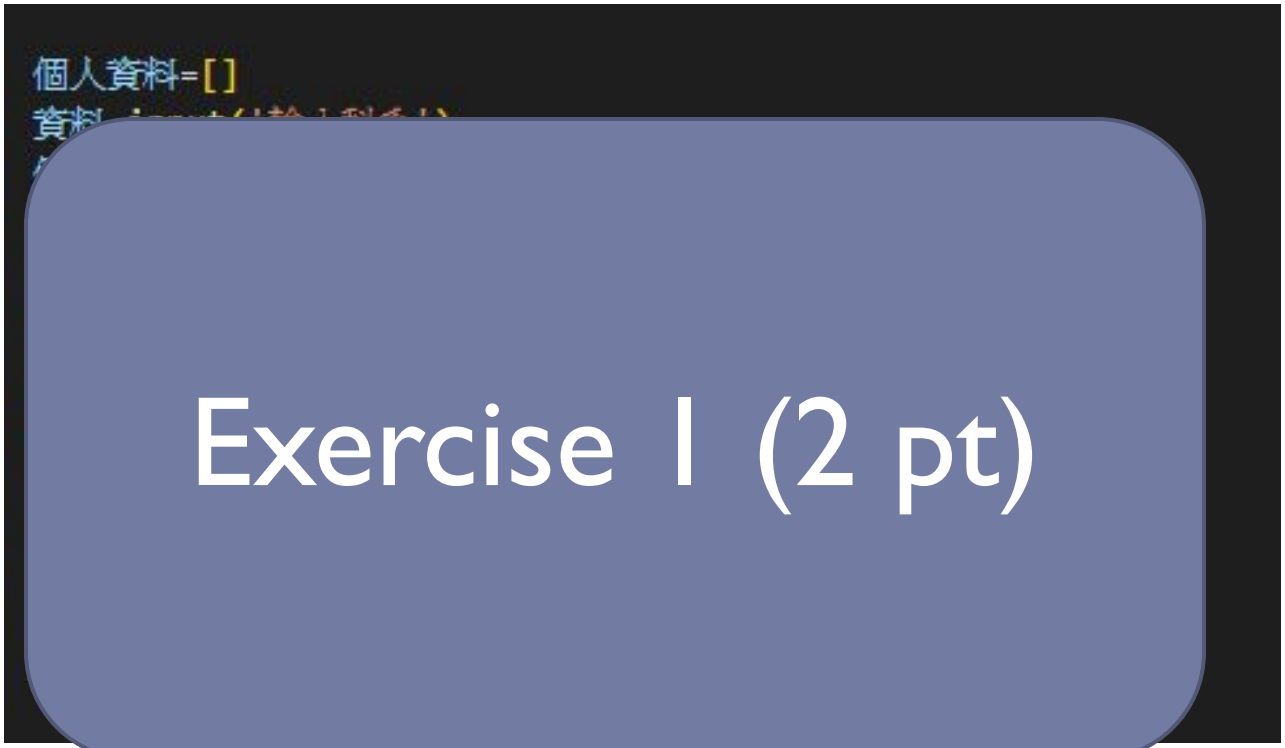
- **範例3-5-1**個人資料:ch3\3-5-1-個人資料.py
- 請設計一個程式將輸入的五項資料加入串列中, 取出最先加入的兩項工作, 顯示取出的資料與剩餘的資料, 接著取出最後加入的一項資料, 顯示取出的資料與剩餘的資料

- 請依個人的科系 班級 學號前三碼 學號後六碼 姓名 完成這次的五項資料

範例3-5-1個人資料:ch3\3-5-1-個人資料.py

- 解題想法
  - 利用串列紀錄個人資料, 使用函式append 將資料加入個人資料, 函式pop 取出個人資料, 使用函式print 顯示**個人資料**。

範例3-5-1 個人資料:ch3\3-5-1-個人資料.py



範例3-5-1 個人資料:ch3\3-5-1-個人資料.py

- 執行結果

```
電機 四乙 ['107', '650014', '盧奕帆']
盧奕帆 ['107', '650014']
```

範例練習2

- 請設計一個程式將學號轉換為姓名，輸入學號可以查詢到對應的姓名，顯示字典的學號有哪些，與顯示整個字典
- 解題想法
  - 學號與姓名對應的關係儲存到字典(dict) 結構內，使用字典的功能完成程式。

範例練習2

- 程式碼

```
1 people = {'01': '王大明', '02': '蔡曉慧', '110000001(本人學號)': '張帥哥(本人姓名)'}
2
3
4
5
6
```

Exercise 2 (2 pt)

範例練習2

- 執行結果

第一次執行輸入「02」，結果如下：

```
可輸入的學號： dict_keys(['01', '02', '110000001(本人學號)'])
{'01': '王大明', '02': '蔡曉慧', '110000001(本人學號)': '張帥哥(本人姓名)'}

請輸入查詢學號：02
對應姓名為： 蔡曉慧
```

第二次執行輸入「10」，結果如下：

```
可輸入的學號： dict_keys(['01', '02', '110000001(本人學號)'])
{'01': '王大明', '02': '蔡曉慧', '110000001(本人學號)': '張帥哥(本人姓名)'}

請輸入查詢學號：10
對應姓名為： 字典找不到該學號
```

To be continued……

Instructor: Cheng-Chun Chang (張正春)
Department of Electrical Engineering