Python 講義

(一)安裝 python 及建立虛擬環境
python 下載
https://www.python.org/

安裝虛擬環境套件
pip install virtualenv

設定虛擬環境位置
virtualenv 名字

啟動虛擬環境
移置 Scripts 下 activate

測試 python 檔案
python 檔名.py

(一)安裝 visualstudio
https://code.visualstudio.com/

ctrl+shift+p
搜尋 lang
configure display language
install additional language

外掛
linting debugging

vs 啟動虛擬環境
get-executionpolicy Restricted
set-executionpolicy remotesigned
啟動.\activate
璃開 deactiate

Python 常數、變數、資料型態、運算式及流程控制

料型態名稱	值的範圍	說明
整數 int	正負整數和零皆可，而且最大值和最小值都沒有限制。	Python 的整數沒有最大值和最小值的限制，就算是三百位數的整數，也能夠儲存和運算。
浮點數 float	正負數和零皆可，最大值大約是 10308，最小值大約是-10308。	電腦計算浮點數會有誤差（不管是哪一種程式語言都一樣）。Python 計算浮點數只保證最前面 16 位數字是正確的。
浮點數 Decimal	正負數和零皆可，而且最大值和最小值都沒有限制。	為了提高浮點數計算的正確性，Python 特別設計了 Decimal 模組。它可以讓我們自己設定正確的位數，而且最大值和最小值都沒有限制。
複數 complex	複數是由實數和虛數部分組合而成	4+8j 3.5-2.8J
字串 str	字串長度決定於程式可用的記憶體容量。	字串是用單引號「'」或是雙引號「"」括起來的資料。如果用單引號或是雙引號把數字括起來，例如'123'，那麼這個數字就變成字串，而不是整數。字串長度可以一直增加，直到程式的記憶體用完為止。
布林值 bool	True 或 False	布林值是用來記錄某一個條件的判斷結果。如果條件成立（也就是「對」），那麼就是 True；如果條件不成立（也就是「錯」），那麼就是 False。
空值 None	None	None 表示什麼都沒有，也就是沒有任何東西。

```
student_name = '李大中'
ch_score = 98
en_score = 95
math_score = 90

# 計算總分和平均
total = ch_score + math_score + en_score
average = total / 3

# 顯示學生姓名和成績
print('學生姓名:', student_name)
print('國英數成績:', ch_score, en_score, math_score)
print('平均成績:', average)
學生姓名: 李大中
國英數成績: 98 95 90
平均成績: 94.33333333333333
```

```
score = input('請輸入成績：')
# 把 score 裏頭的「數字字串」轉換成整數，再存回 score
score = int(score)
```

```
score = int(input('請輸入成績：'))

name_score = input('請輸入姓名和成績，資料之間加入空格：')
name, score = name_score.split() # 這行程式碼是重點
score = int(score) # 把 score 從字串型態轉成整數型態
```

python 運算子

運算子功能	運算符號	範例
加法	+	3.29 + 5 value1 + 100 value1 + value2 + 1.23e5
減法	-	3.29 - 5 value1 - 100 value1 - value2 - 1.23e5
乘法	*	3.29 * 5 value1 * 100 value1 * value2 * 1.23e5
除法	/	3.29 / 5 value1 / 100 value1 / value2 / 1.23e5
求商	//	3.29 // 5 value1 // 100 value1 // value2 // 1.23e5
求餘數	%	3.29 % 5 value1 % 100 value1 % value2 % 1.23e5
次方	**	3.29 ** 2 value1 ** 0.5 value1 ** value2 ** 3

Python 的算術指定運算子

運算子功能	運算符號	範例
加法指定	+=	value1 += 3.29 value1 += value2 + 5
減法指定	-=	value1 -= 3.29 value1 -= value2 + 5
乘法指定	*=	value1 *= 3.29 value1 *= value2 + 5
除法指定	/=	value1 /= 3.29 value1 /= value2 + 5
求商指定	//=	value1 //= 3.29 value1 //= value2 + 5
求餘數指定	%=	value1 %= 3.29 value1 %= value2 + 5
次方指定	**=	value1 **= 3.29 value1 **= value2 + 5

變數 A = 變數 A + 某一個數值
變數 A += 某一個數值

x=x+1
x+=1

變數 A **= 變數 B + 5
變數 A = 變數 A ** (變數 B + 5)

變數 A = 變數 A + (數值或運算式)
變數 A += 數值或運算式

Python 的關係運算子

運算子功能	運算符號	範例
大於	>	5 > 3.29 value1 > 100 value1 + value2 > 1.23e2
大於等於	>=	5 >= 3.29 value1 >= 100 value1 >= 1.23e2 + value2
小於	<	-5 < 3.29 value1 < 100 value1 + value2 < 1.23e2
小於等於	<=	-5 <= 3.29 value1 <= 100 value1 <= 1.23e2 * value2
等於	==	-5 == -5 value1 == 100 value1 == 1.23e2 * value2
不等於	!=	-5 != 3.29 value1 != 100 value1 != 1.23e2 * value2

3 < 5
score >= 60
ch_score < en_score
ch_score + en_score >= 180
math_score * 2 < ch_score + en_score

If 判斷式

```
if 關係運算式 1:
    # 如果關係運算式 1 為 True，就會執行這段程式碼
    # 這段程式碼必須內縮至少 1 個空格，建議內縮 4 個空格
    ...
elif 關係運算式 2:
    # 如果關係運算式 2 為 True，就會執行這段程式碼
    # 這段程式碼必須內縮至少 1 個空格，建議內縮 4 個空格
    ...
.
. # 其他 elif 程式碼
.
else:
    # 如果前面的關係運算式全部都是 False，就會執行這段程式碼
    # 這段程式碼必須內縮至少 1 個空格，建議內縮 4 個空格
    ...
```

```
if 90 <= score <= 100:
    score_level = 'A'
elif score >= 80:
    score_level = 'B'
elif score >= 70:
    score_level = 'C'
elif score >= 60:
    score_level = 'D'
else:
    score_level = 'E'
```

```
# gender 是使用者輸入的性別
# age 是使用者輸入的年齡

if gender == '男':
    if age >= 30:
        print('已經到達適婚年齡')
    else:
        print('還未達到適婚年齡')
else:   # 女生的情況
    if age >= 25:
        print('已經到達適婚年齡')
    else:
        print('還未達到適婚年齡')
```

邏輯運算子

邏輯運算子	功能說明
and	表示二個條件都要成立，等於中文「而且」的意思。
or	表示二個條件只要其中一個成立即可，等於中文「或」的意思。
not	把條件變成相反，等於中文「不是」的意思。

And 運算子的真值表

關係運算式 A	關係運算式 B	關係運算式 A and 關係運算式 B
True	True	True
False	True	False
True	False	False
False	False	False

Or 運算子的真值表

關係運算式 A	關係運算式 B	關係運算式 A or 關係運算式 B
True	True	True
False	True	True
True	False	True
False	False	False

Not 運算子的真值表

關係運算式	not 關係運算式
True	False
False	True

```
if score1 < 60 or score2 < 60 or score3 < 60:
    print("有成績不及格")
else:
    print("全部及格")
```

```
if not score1 < 60 and not score2 < 60 and not score3 < 60:
    print("全部及格")
else:
    print("有成績不及格")
```

```
if score1 >= 60 and score2 >= 60 and score3 >= 60:
    print("全部及格")
else:
    print("有成績不及格")
```

```
if (not score1 < 60) and (not score2 < 60) and (not score3 < 60):
    print("全部及格")
else:
    print("有成績不及格")
```

For 迴圈的語法

```
for 自訂變數 in range(起始值, 結束值, 遞增值):
    # 處理自訂變數中的資料
    …

    if 關係運算式:
        continue   # 跳過以下程式碼，立刻進入 range 數列的下一項
    …

    if 關係運算式:
        break   # 離開迴圈
    …
else:
    # 當沒有用 break 指令跳出迴圈時，會執行這個程式區塊
    …
```

```
sum = 0   # 累加變數
for i in range(1, 101, 1):
    sum += i

print(sum)
```

```
sum = 0   # 累加變數
for i in range(1, 101, 1):
    # 如果是偶數，直接跳到下一次迴圈
    if i % 2 == 0:
        continue

    sum += i

print(sum)
```

```
# 450 和 900 的公因數一定是 1 到 450 之間的整數
for i in range(1, 451, 1):
    # 檢查 i 是否可以整除 450 和 900
    if 900 % i == 0 and 450 % i == 0:
        print(i)
```

```
import random as rand   # 載入亂數套件
for i in range(0, 5, 1):   # 設定執行 5 次
    # 產生一個介於 1 和 100（含）之間的亂數
    rand_num = rand.randint(1, 100)

    # 根據亂數決定檢測結果
    if rand_num < 90:
        print("通過檢測")
    else:
        print("檢測失敗")
        break
else:
    # 當沒有用 break 指令跳出迴圈時執行
    print("通過品管檢測")
```

巢狀迴圈

for i in range(1, 6, 1):
  for j in range(1, 3, 1):
  print(i, j)
	1 1
	1 2
	2 1
	2 2
	3 1
	3 2
	4 1
	4 2
	5 1
	5 2

for i in range(1, 10, 1):
  for j in range(1, 10, 1):
  print(i, 'x', j, '=', i*j)
	1 x 1 = 1
	1 x 2 = 2
	...
	1 x 9 = 9
	2 x 1 = 2
	2 x 2 = 4
	...
	2 x 9 = 18
	...
	9 x 9 = 81

for i in range(1, 10, 1):
  for j in range(0, i, 1):
  print(i, end='') # 設定 end 參數讓 print()不要換行
 	
  print() # 每印完一個數字要換行
	1
	22
	333
	4444
	55555
	666666
	7777777
	88888888
	999999999

While 迴圈

```
while 關係運算式:  # 這個關係運算式控制迴圈是否執行
    # 迴圈中的程式碼
    …

    if 關係運算式:
        continue
    …

    if 關係運算式:
        break
    …
else:
    # 第一行的關係運算式是 False 的情況下才會執行
    # 用 break 指令跳出迴圈時不會執行
    …
```

```
sum = 0  # 累加變數
i = 1  # 索引變數
while i <= 100:
    sum += i
    i += 1  # 改變索引變數的值
print(sum)
```

```
score = int(input("輸入學生成績："))

while score < 0 or score > 100:
    score = int(input("成績錯誤，請重新輸入："))
```

```
a, b = input('請輸入二個正整數：').split()
a = int(a)
b = int(b)
# 如果 a < b，把 a 和 b 對調
if a < b:
    a, b = b, a
r = a % b
# While 迴圈會計算到 r=0 才會停止
while r != 0:
    a = b
    b = r
    r = a % b

print('GCD 是 ' + str(b))
```

```
a, b = input('請輸入二個正整數：').split()
a = int(a)
b = int(b)
while a > 0:
    # 如果 a < b，把 a 和 b 對調
    if a < b:
        a, b = b, a
    r = a % b
    while r != 0:
        a = b
        b = r
        r = a % b
    a = input('請輸入下一個正整數（0 或負數表示結束）：')
    a = int(a)
print('GCD 是 ' + str(b))
```

List 資料組

串列、元組、字典、集合型態及函數認識

List 資料組和 Tuple 資料組的基本觀念很類似。

1. List 資料組建立之後，內部的資料可以變更，也可以增加和刪除。
2. List 資料組中的資料可以有不同的資料型態，甚至可以包含 Tuple 資料組或是 List 資料組。
3. List 資料組可以利用指定運算子把資料分配給多個變數。
4. List 資料組必須用方括弧把資料括起來。

把以上特點和 Tuple 資料組的特點做比較，會發現以下二點差異：

List 資料組的內容可以改變，也可以新增和刪除。但是 Tuple 資料組中的資料不可以修改。再來就是 List 資料組是用方括弧括起來，Tuple 資料組是用圓括弧括起來。

```
x = [10, 20, 30]  # 一定要用方括弧括起來
y = ['Tom', 12]
print(x, y)

x1, x2, x3 = x
y1, y2 = y

*x1, x2 = x  # 如果 x 是[10, 20, 30]，x1 會收到 10 和 20，x2 則收到 30
x1, *x2 = x  # 如果 x 是[10, 20, 30]，x1 會收到 10，x2 則收到 20 和 30

x1 = x[0]
x2 = x[1]

x = [10, 20, 30]  # List 資料組
y = 'Tom', 12  # Tuple 資料組
z = [-1, x, y, 'end']

x = [10, 'a', 'b', 10, 10, 'a']
print(x.count('a'))  # 顯示 2，因為 x 中有 2 個'a'
print(x.index(10))  # 顯示 0，因為第一個 10 是在索引值 0 的位置
len(x)              # 顯示 6，因為 x 中有 6 筆資料

x = [10, 'a', 'b', 10, 10, 'a']
for item in x:
    print(item)



my_favorite_colors = ['blue', 'yellow', 'purple']
```

```
# color 物件中已經儲存一個顏色，例如 color = 'red'
if color in my_favorite_colors:
    print(color + '是我喜歡的顏色')
else:
    print(color + '不是我喜歡的顏色')


x = [10, 'a', 'b', 10, 10, 'a']
x[0] = 100   # x[0]的值從 10 變成 100
x[1] = 200   # x[1]的值從'a'變成 200
```

---

1. 利用 List 資料組的 append()方法
append()方法一次只能夠加入一項資料，而且是加在資料組的最後面。

```
x.append(300)
x.append('Tom')
x.append([50, 60])   # [50, 60]會當成一筆資料加入
```

2. 利用 List 資料組的 extend()方法
extend()方法可以一次把多項資料加在原來資料的最後，但是必須把這些資料放在一個 List 資料組，或是 Tuple 資料組裏頭。

```
x.extend([300])   # 把要加入的資料放在資料組裏頭
x.extend([50, 60])   # 50 和 60 會當成二筆資料加入
```

3. 利用 List 資料組的 insert()方法
insert()方法可以把一筆資料插入指定的位置，原來的資料會自動往後退。

```
x.insert(2, 300)   # 把資料插入索引 2 的位置，也就是第三個位置
```

4. 利用運算子「+」和「+=」
這二個運算子的功能就像是 extend()方法。

```
x += ['John']        # 等於執行 x.extend(['John'])
x += ['John', -10]   # 等於執行 x.extend(['John', -10])
y = x + ['John', -10]   # List 資料組相加就是把它們的資料合起來
```

如果要把資料從 List 資料組取出或刪除，有三種方法：

1. 利用 List 資料組的 remove()方法
remove()方法會從第一項資料開始，找到指定的資料，然後將它刪除。如果有多筆相同的資料，只會刪除第一個找到的資料。

```
x.remove('a')
```

2. 利用 List 資料組的 pop()方法
pop()方法可以從指定的位置取出資料，該資料會被移除。如果沒有指定位置，就取出最後一筆資料。

```
y = x.pop(3)  # 取出資料組 x 的索引 3 的資料（也就是第四筆資料）
y = x.pop()   # 取出資料組 x 的最後一筆資料
```

3. 利用 List 資料組的 clear()方法
clear()方法會清除所有資料。

```
x.clear()
```

```
# 提示使用者輸入學生人數
num_student = int(input("請輸入學生人數："))

score_list = []
for i in range(0, num_student):  # 利用學生人數決定迴圈執行的次數
    score = int(input("請輸入學生成績："))
    score_list += [score]

print("學生成績：", score_list)
```

Tuple 資料組

Tuple 資料組有以下特點:

1. Tuple 資料組建立之後，裏頭的資料就不可以再改變。
2. Tuple 資料組中的資料可以有不同的資料型態，甚至可以包含另一個資料組。
3. Tuple 資料組可以利用指定運算子把資料分配給多個變數。
4. Tuple 資料組顯示的時候，會用圓括弧把資料括起來。

```
x = 10, 20, 30   # 資料全部是數字
y = 'Tom', 12    # 資料包含字串和數字
print(x, y)

x1, x2, x3 = x
y1, y2 = y

*x1, x2 = x   # 如果 x 是(10, 20, 30)，x1 會收到 10 和 20，x2 則收到 30
x1, *x2 = x   # 如果 x 是(10, 20, 30)，x1 會收到 10，x2 則收到 20 和 30

x1 = x[0]
x2 = x[1]

x = 10, 20, 30
y = 'Tom', 12
z = -1, x, y, 'end'

x = 10, 20, 30, 10, 10, 30
print(x.count(10))   # 顯示 3，因為 x 中 10 出現 3 次
print(x.index(30))   # 顯示 2，因為 30 的索引值是 2（索引值從 0 開始）
len(x)               # 顯示 6，因為 x 中有 6 筆資料

for 自訂變數 in 資料組物件:
    # 處理自訂變數中的資料
    ...

x = 10, 20, 30, 10, 10, 30
for item in x:
    print(item)
```

利用單行 For 迴圈產生 List 資料組

```
num_list = []
for i in range(1, 11):   # range()的第三個參數沒有設定，表示用預設值 1
    num_list += [i]   # 也可以寫成 num_list.extend([i])

num_list = [i for i in range(1, 11)]

資料組 = [自訂變數 for 自訂變數 in 資料組 if 關係運算式]

num_list = [i for i in range(1, 11) if i % 2 == 0]

# scores 是儲存考試成績的 List 資料組
scores_copy = [item for item in scores]          # 複製全部成績
fail = [item for item in scores if item < 60]   # 只複製不及格的成績

sum = [i + j for i in range(1, 4) for j in range(10, 14)]

sum = []
for i in range(1, 4):
    for j in range(10, 14):
        sum += [i + j]

[11, 12, 13, 14, 12, 13, 14, 15, 13, 14, 15, 16]

sum = [i + j for i in range(1, 4) if i % 2 == 1
        for j in range(10, 14) if j % 2 == 0]

x1 = 1, 'a', (1, 1)      # tuple 資料組
x2 = 2, 'b', (2, 2)      # tuple 資料組
y1 = [1, 'a', (1, 1)]    # list 資料組
y2 = [2, 'b', 2]         # list 資料組

[11, 13, 13, 15]
```

取得資料的索引值

```
# 產生數列 0, 100, 200, 300, 400, 500
num_list = [i for i in range(0, 501, 100)]

for item in enumerate(num_list):
    print(item)
(0, 0)
(1, 100)
(2, 200)
(3, 300)
(4, 400)
(5, 500)

for i, data in enumerate(num_list):
    print(i, data)

for i, data in enumerate(num_list):
    # 當索引值符合我們的條件時才執行，否則跳過
    if i % 2 == 0:  # 第 1 筆資料的索引值是 0、第 3 筆的索引值是 2、...
        print(data, end=' ')  # 指定用空格結束，不要換行
```

```
# 產生數列 0, 100, 200, 300, 400, 500
num_list = [i for i in range(0, 501, 100)]
for item in enumerate(num_list):
    print(item)
(0, 0)
(1, 100)
(2, 200)
(3, 300)
(4, 400)
(5, 500)
for i, data in enumerate(num_list):
    print(i, data)
for i, data in enumerate(num_list):
    # 當索引值符合我們的條件時才執行，否則跳過
    if i % 2 == 0:  # 第 1 筆資料的索引值是 0、第 3 筆的索引值是 2、...
        print(data, end=' ')  # 指定用空格結束，不要換行
```

索引區間的用法 – Slice

Slice 是「切片」的意思
第一種索引是以第一項資料為基準，它的索引值是 0，然後依序往後遞增。
第二種索引是以最後一項資料為基準，它的索引值是-1，然後依序往前遞減。
也就是說，這二種索引方式的使用時機是：

如果要以第一項資料為基準，就用第一種索引方式。如果要以最後一項資料為基準，就用第二種索引方式。

```
# num_list 是一個 List 資料組
num_list[0] = 100   # 把第一項設定成 100
num_list[1] = 200   # 把第一項設定成 200
num_list[-1] = 900   # 把最後一項設定成 900
num_list[-2] = 800   # 把倒數第二項設定成 800

num_list[len(num_list) - 1] = 900   # 把最後一項設定成 900
num_list[len(num_list) - 2] = 800   # 把倒數第二項設定成 800
```

資料組名稱[索引起始值 : 索引結束值 : 索引值改變量]
也就是說，我們可以在索引中指定起始值、結束值、和改變量，這三個值的功能如下：

1. 「索引起始值」是指定開始的位置，如果這部分空白，表示要從第一項或是最後一項開始，如果索引值改變量是正整數，就從第一項開始，如果是負整數，就從最後一項開始。
2. 「索引結束值」是指定結束的位置，但是要注意，這個位置的資料不包含在內，也就是只會取到它的前一項。如果這部分空白，代表最後一個索引值加 1，也就是一直取到最後一項資料為止。
3. 「索引值改變量」是設定資料的間隔，正負整數皆可，空白表示使用預設值 1。

```
# num_list 是一個 List 資料組
num_copy = num_list[:]   # 索引起始值、結束值、改變量都不指定，表示複製全部資料
num_copy = num_list[::2]   # 從第一項資料開始，間隔一個，取出下一個
num_copy = num_list[:3]   # 複製前 3 項資料，也就是索引值 0, 1, 2
num_copy = num_list[-3:]   # 複製最後 3 筆資料，也就是索引值-3, -2, -1
num_copy = num_list[1:-1]   # 從第二筆資料開始複製，直到倒數第二筆資料（含）
num_copy = num_list[::-1]   # 從最後一筆資料開始複製，直到第一筆資料
```

單行 For 迴圈可以搭配 If 判斷式來篩選資料，而 Slice 只能從指定的位置取得資料，無法根據資料的內容來篩選。

```
# 產生數列 0, 100, 200, 300, 400, 500
num_list = [i for i in range(0, 501, 100)]
for item in num_list[1:-1]:
    print(item)

for i, data in enumerate(num_list[1:-1]):
    print(i, data)
```

用巢狀 For 迴圈處理二維資料組

```
# 用來儲存所有成績的資料組
all_score_list = []
for i in range(0, 3):  # 設定第一層迴圈執行 3 次，每次處理一個班級
    print('第', i + 1, '班 ----------')

    # 用來儲存一個班級的成績
    score_list = []

    for j in range(0, 10):  # 第二層迴圈執行 10 次，每次輸入一筆成績
        score = int(input('請輸入學生成績：'))
        score_list += [score]

    # 存入這個班級的成績
    all_score_list += [score_list]

# 顯示每個班級的成績
for i, score_list in enumerate(all_score_list):
    print('第', i + 1, '班學生成績：', score_list)
```

```
# 詢問班級數
num_class = int(input('班級數：'))
# 用來儲存所有成績的資料組
all_score_list = []
for i in range(0, num_class):  # 設定第一層迴圈執行 num_class 次
    print('第', i + 1, '班 ----------')

    # 詢問學生人數
    num = int(input('學生人數：'))

    # 用來儲存一個班級的成績
    score_list = []

    for j in range(0, num):  # 第二層迴圈執行 num 次，每次輸入一筆成績
        score = int(input('請輸入學生成績：'))
        score_list += [score]

    # 存入這個班級的成績
    all_score_list += [score_list]

# 顯示每個班級的成績
for i, score_list in enumerate(all_score_list):
    print('第', i + 1, '班學生成績：', score_list)
```

```
# all_score_list 是儲存學生成績的二維資料組
pass_count_list = []  # 儲存及格人數的 List 資料組
fail_count_list = []  # 儲存不及格人數的 List 資料組

for score_list in all_score_list:
    pass_count = 0  # 累計目前班級的及格人數
    fail_count = 0  # 累計目前班級的不及格人數

    # 用 For 迴圈檢查目前班級的每一筆成績
    for score in score_list:
        if score >= 60:
            pass_count += 1
        else:
            fail_count += 1
```

```
    # 把累計結果存入資料組
    pass_count_list += [pass_count]
    fail_count_list += [fail_count]

for i in range(0, len(pass_count_list)):
    print('第', i + 1, '班及格人數', pass_count_list[i],
          '不及格人數', fail_count_list[i])
```

讀取和寫入資料組的不同作法

建立資料組

```
# 建立一個空的 List 資料組
data_list = []

for data in new_data:   # new_data 中是要加入資料組的資料
    data_list += [data]   # 把新資料加到資料組最後面
```

讀取資料組

```
# data_list 是一維資料組
for i, data in enumerate(data_list):
    # i 是資料的索引，data 是資料
    ...

# data_list 是一維資料組
for i, data in enumerate(data_list):
    # i 是資料的索引，data 是資料
    ...
```

修改資料組

```
資料組名稱[索引值] = 新值

# data_list 是一維資料組
# 我們可以利用 range()函式建立一個資料組索引數列
for i in range(0, len(data_list)):
    data_list[i] = -1   # 把目前這個位置的資料設定成-1
```

Set 和 FrozenSet 資料組

Set 資料組的特性：

1. Set 資料組中的資料不可以重複。也就是說，不可以出現一樣的資料。
2. Set 資料組中的每一項資料都必須是固定的內容，不可以變更。因此像是 List 這種可以變更內容的物件，就不可以放入 Set 資料組。但是 Tuple 資料組可以放入 Set 資料組，因為 Tuple 資料組的內容不會變更。
3. Set 資料組中的資料沒有先後順序，所以無法用索引值取出 Set 資料組裏頭的資料。

Set 資料組和 FrozenSet 資料組的差別只有一點，就是 Set 資料組的內容可以隨意修改（包括加入資料和刪除資料），但是 FrozenSet 資料組一旦建立之後，它的內容就固定了，不可以再更動。所以 Set 資料組裏頭可以儲存 FrozenSet 資料組，但是不能夠儲存 Set 資料組。

建立 Set 資料組和 FrozenSet 資料組

```
s = set()  # 建立空的 set 資料組
t = {1, 2, 'abc', (10, 20)}  # 有初始資料的 set 資料組
u = set({1, 2, 'abc', (10, 20)})  # 用內建函式 set()來建立
v = set([1, 2, 'abc', (10, 20)])  # 根據 list 資料組來建立
frozen_set = frozenset({1, 2, 'abc', (10, 20)}) # 建立 frozenset

# 建立 A 到 Z 的字母
letters = {chr(c) for c in range(65, 91)}

# 建立 A 到 Z 的字母和 ASCII 碼對應
letters = {(c, chr(c)) for c in range(65, 91)}

# 只建立奇數 ASCII 碼的字母
letters = {(c, chr(c)) for c in range(65, 91) if c % 2 != 0}
```

加入或是刪除資料

```
s = {1, 2, 'abc', (10, 20)}
len(s)  # 得到的結果是 4
s.add(3)  # 把 3 加入 s 資料組

# 加入 100 和 200 二項資料，資料必須先放在資料組裏頭
s.update([100, 200])

s.remove((10, 20))  # 從 s 中刪除 Tuple 資料組(10, 20)
s.clear()  # 刪除全部資料
```

集合運算子

集合 運算	集合 運算子	對應的集合物件的方法	範例
聯集	|	union()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s | t s.union(t)
交集	&	intersection()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s & t s.intersection(t)
差集	-	difference()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s - t s.difference(t)
對稱差集	^	symmetric_difference()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s ^ t s.symmetric_difference(t)
聯集指定	|=	update()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s |= t s.update(t)
交集指定	&=	intersection_update()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s &= t s.intersection_update(t)
差集指定	-=	difference_update()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s -= t s.difference_update(t)
對稱差集指定	^=	symmetric_difference_update()	# s 和 t 是二個 set 資料組 # 以下二個運算式的結果完全相同 s ^= t s.symmetric_difference_update(t)

```
s = {1, 2, 'abc', (10, 20)}
t = {1, (10, 20, 30)}
u = s | t   # 結果為{1, 2, 'abc', (10, 20), (10, 20, 30)}
u = s & t   # 結果為{1}
u = s - t   # 結果為{2, 'abc', (10, 20))}
u = t - s   # 結果為{(10, 20, 30)}
u = s ^ t   # 結果為{2, 'abc', (10, 20), (10, 20, 30)}
```

Dict 資料組

Dict 資料組就是利用 key 和 value 的對應關係，來儲存和取得資料。而且它的 key 不一定要使用字串，還可以用數值，或是 Tuple 資料組。

```
d1 = {'id':5, 'name':'john', 'age':18, 8:'a number', (1, 2):100}
d2 = {}  # 建立空的 dict 資料組

# 利用 dict()內建函式建立 dict 資料組，請留意參數的格式
d3 = dict(id=5, name='john', age=18)

print(d1['id'])
print(d1[8])
print(d1[(1, 2)])
print(d3['name'])
print(d3['age'])
```

加入、修改和刪除資料

```
# d 是已經建立的 dict 資料組
len(d)  # 計算資料組中有幾筆資料

# 如果 100 這個 key 已經存在，就修改它對應的資料，否則就加入這筆資料
d[100] = 'score'

# 如果'Peter'這個 key 已經存在，就修改它對應的資料，否則就加入這筆資料
d['Peter'] = 'my friend'

# 刪除 100 這個 key 和它對應的資料
del d[100]

# 取出 d['Peter']。等於先執行 d['Peter']，再執行 del d['Peter']
data = d.pop('Peter')

d.clear()  # 刪除全部資料
```

使用字串和相關內建函式

```
# 建立三個字串 s1、s2 和 s3
s1 = '123'
s2 = 'Peter'
s3 = '王大一'

# 使用內建函式 str()和 chr()建立字串
num = 90
s4 = str(num)        # '90'
s5 = str(3.14159)   # '3.14159'
s6 = str(5.5e2)      # '550.0'
s7 = str('Mary Taylor')
s8 = chr(65)         # 'A', '
```

以下函式也可以用來處理字串：

1. len(字串物件)
   計算字串中有幾個字元。
2. max(字串物件)
   找出字串中 ASCII 字元碼最大的字元。
3. min(字串物件)
   找出字串中 ASCII 字元碼最小的字元。

請參考以下程式碼範例：

```
# s7 是前面程式碼範例建立的字串物件
len(s7)   # 結果是 11
max(s7)   # 結果是'y'，因為'Mary Taylor '中，'y'的 ASCII 字元碼最大
min(s7)   # 結果是'M'，因為'Mary Taylor '中，'M'的 ASCII 字元碼最小
```

函式的格式和運作原理

```
def 函式名稱(參數 1, 參數 2, 參數 3, ...):
    # 以下是函式中的程式碼
    # 程式碼會處理參數 1, 參數 2, 參數 3, ...中的資料
    .
    .
    .

    # 最後傳回結果
    return 傳回值   # 如果不需要傳回值，就直接使用 return 指令即可
```

```
# 建立 print_hello()函式
def print_hello():
    print('Hello')

# 呼叫 print_hello()
print_hello()
```

```
# 建立 print_hello()函式
def print_hello():
    print('Hello')
    print_name('John')   # 呼叫 print_name()

# 建立 print_name()函式
def print_name(name):
    print(name)

# 呼叫 print_hello()
print_hello()
```

```
# 建立 sum_scores()函式
def sum_scores(scores):
    sum = 0
    for n in scores:
        sum += n;

    # 傳回加總結果
    return sum

scores = [70, 80, 90, 95, 100]
# 呼叫 sum_scores()，傳入資料組，並且把傳回值設定給 sum 物件
sum = sum_scores(scores)
print('總分', sum)
```

全域變數和區域變數

「全域變數」就是有效範圍比較大的變數，「區域變數」就是有效範圍比較小的變數

```
a = 1  # 建立全域變數 a

def fun():
    b = a  # 建立區域變數 b，將它設定為全域變數 a 的值
    a = 10  # 改變全域變數 a 的值
    b = c  # 讀取全域變數 c 的值，將它設定給變數 b
    print(b)  # 顯示變數 b 的值

c = 3  # 建立全域變數 c

fun()  # 呼叫函式

print(a)  # 顯示變數 a 的值
print(b)  # 顯示變數 b 的值
print(c)  # 顯示變數 c 的值
```

```
a = 1  # 建立全域變數 a

def fun():
    global a, c  # 宣告函式中會用到全域變數 a 和 c
    a = 10  # 改變全域變數 a 的值
    c = 30  # 改變全域變數 c 的值

c = 3  # 建立全域變數 c

fun()  # 呼叫函式

print(a)  # 顯示變數 a 的值
print(c)  # 顯示變數 c 的值
```

函式的屬性與靜態變數

```
def fun():
    fun.count += 1  # 使用函式的屬性
    print(fun.count)

fun.count = 0  # 必須在函式程式碼後面建立函式的屬性
```

```
def fun():
    # 先嘗試修改函式屬性的值，如果出現 AttributeError
    # 表示該屬性不存在，這時候就建立該屬性
    try:
        fun.count += 1  # 修改函式屬性的值
    except AttributeError:
        fun.count = 1  # 建立函式的屬性

    print(fun.count)
```

相關資料參考

https://sites.google.com/site/ezpythoncolorcourse/

https://www.w3schools.com/python/default.asp

https://www.runoob.com/python3/python3-tutorial.html

http://tw.gitbook.net/python/index.html