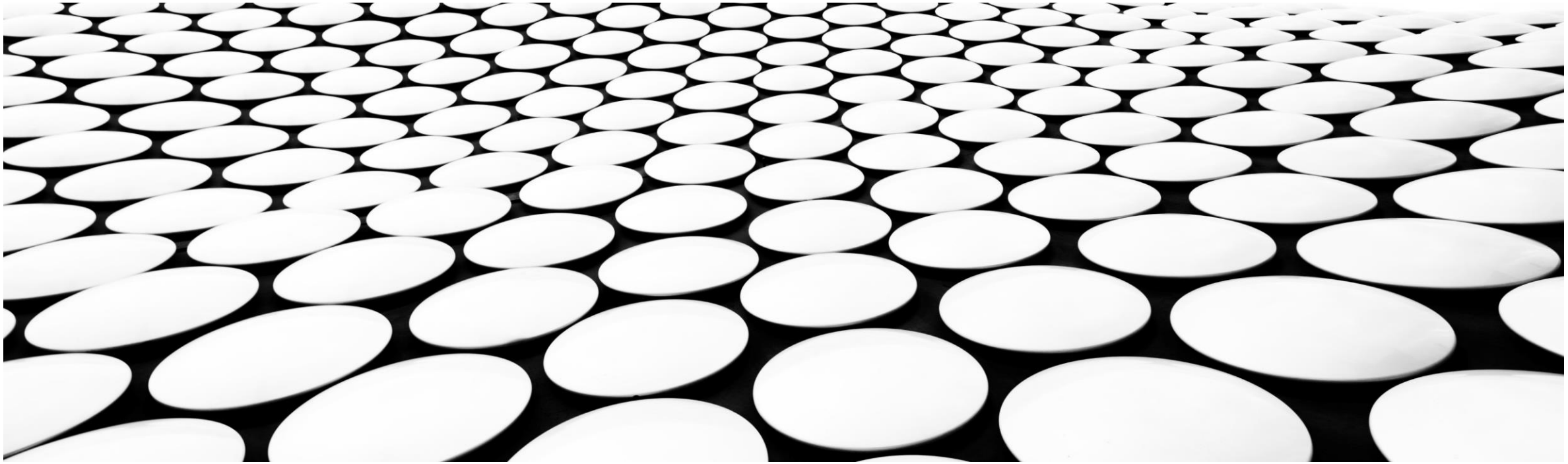
Python 基本語法 I

MARs教育團隊 – 孫汶琳 交大電機博士候選人



目錄

註解(Comments)

- **用於說明、解釋程式碼，並增添程式碼的可讀性**
- **在測試程式碼時，可避免執行特定的程式碼片段**
- **單行註解**
  - 使用一個米字符號: #
- **多行註解**
  - Python並未提供跨行的註解方式
  - 以下提供兩種替代方案:
    1. 為每一行開頭都加上 #
    2. 利用Python直譯器會忽略未指派給變數的字串之特性來做為註解使用，使用兩組三個的單引號(')或雙引號(")

```
# 這是一個單行註解
# 想要多行註解可以多用幾次
# 像是這樣

'''
利用Python直譯器的特色，
我們可以把未指派給變數的字串，
當作是多行註解來使用。
'''


"""
要用單引號或者雙引號都可以，
不過要記得不要混用唷!
"""
```

變數(Variables)與資料型態(Data Types)

- 變數
  - 用於「儲存資料」並「為資料命名」
  - 比較程式A和程式B:

```
# 程式A: 不使用變數
print('My BMI is ', 50 / (1.5 * 1.5))
```

```
# 程式B: 使用變數
weight = 50
height = 1.5
BMI = weight / ( height * height)
print('My BMI is ', BMI)
```

  - **變數除了可以幫助我們理解程式外，也方便我們修改資料數值!**

- 資料型態
  - 資料的種類，例如:
    - 布林(Boolean)
    - 數字(Numbers): 整數(Integer)、浮點數(Floating Point Number)、複數(Complex Number)
    - 字串(String)
  - **掌握變數資料型態，才能夠正確地處理資料唷!**

Python常用資料型態整理

項目	資料型態	範例
字串 String	**str**	x = "Hello world!" 或 x = 'Hello world!'
整數 Integer	**int**	x = 10
浮點數 Floating Point Number	**float**	x = 10.5
布林 Boolean	**bool**	x = True 或 x = False
串列 List	**list**	x = ['Alice', 'Bob', 'Cindy']
資料組 Tuple	**tuple**	x = ('Alice', 'Bob', 'Cindy‘)
範圍 Range	**range**	x = range(10)
字典 Dictionary	**dict**	x = {‘name’ : ‘Winnie’, ‘weight’ : 50, ‘height’ : 1.5}
集合 Set	**set**	x = {'Alice', 'Bob', 'Cindy‘}

深入學習變數 – 產生變數

- 和C/C++、Java等高階程式語言不同，Python變數沒有所謂「宣告」的步驟
- Python變數在被賦予值的同時被創造，但是被使用前須要先被賦予值
- 命名規則:
  - 開頭必須是字母或底線，其餘可以是字母、數字、底線
  - 英文字母大小寫是有區別的
  - 保留字不可以被用作變數名稱

```
_____x = 10                    # 合法
my_name123 = 'Winnie'          # 合法
print(my_name123)              # 合法
123my_name = 'Wen-Lin'         # 不合法，因為開頭不得為數字
print(y)                       # 不合法，因為 y 還沒有被賦予值
int = 100                      # 不合法，因為int是保留字
```

深入學習變數 – 轉換變數型態

- Python提供簡便的方式來做型別的轉換

資料型態(欲轉換的對象)

```
x = str(10)     # x 會是 字串    '10'
y = int(10)     # y 會是 整數     10
z = float(10)   # z 會是 浮點數 10.0
```

深入學習變數 – 檢查變數型態

- 若想確認特定變數的資料型態，可以使用 **type()** 這個內建函式

```
x = 10
my_name123 = 'Winnie'
print(type(x))                  # 輸出結果: <class 'int' >
print(type(my_name123))         # 輸出結果: <class 'str' >
```

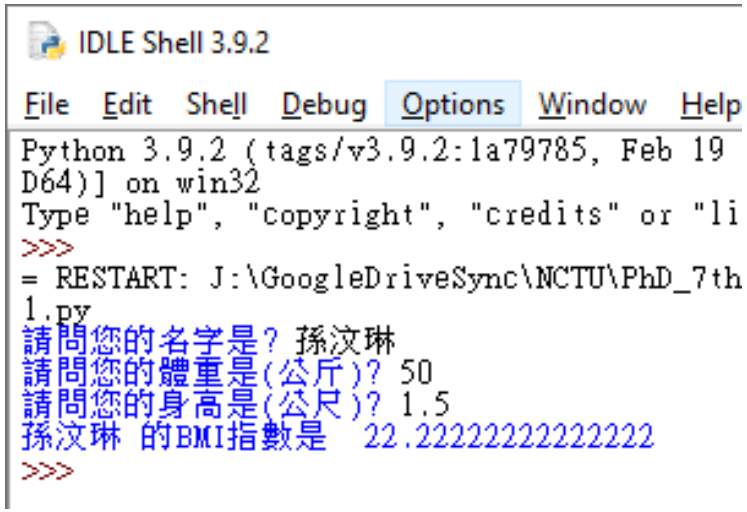
輸入與輸出(Input/Output)

- **輸入與輸出是最基本的互動式程式必備條件:**
  - 輸入是讓使用者提供資料的動作，常見的輸入裝置如: 鍵盤、滑鼠等。
  - 輸出是執行程式後結果的呈現，常見的輸出裝置如: 螢幕、印表機、耳機等。
- **標準的輸入和輸出裝置為鍵盤和螢幕，因此Python提供了對應的內建函式**
  - 輸入: **input()**
    - 由輸入得到的資料型態是字串，因此使用前需要透過適當的轉型再使用!
  - 輸出: **print()**

```
username = input("What's your name? ")  # input可以傳入提示字串
print(username)                          # output 可以直接輸出變數
print('Hello, ', username)               # output 可以使用逗點串接
age = input('How old are you? ')         # 從input獲得的資料型態是字串
age = int(age)                           # 先做資料型態轉換
print(age)                               # 再做使用
```

[課間練習] 互動式BMI計算程式

- 試著利用目前所學，改良右側之BMI計算程式
- 功能要求:
  - 詢問使用者名稱、身高、體重
  - 輸出使用者名稱以及其BMI指數
- 範例執行結果
  - 藍色字為程式輸出，黑色字為使用者輸入




```
# 這是一支計算BMI的程式
weight = 50  # 體重
height = 1.5  # 身高
BMI = weight / ( height * height)  # 計算BMI
print('My BMI is ', BMI)  # 輸出BMI計算結果
```

運算子 (Operators)

- Python的運算子種類相較於其他程式語言來說非常多元
- 多元的運算子種類使其對於一件事情，能使用更精練的程式語句來陳述
- 運算子種類
  - 數學運算子
    - 例如: 加(**+**)、減(**-**)、乘(*****)、除(**/**)、取餘數(**%**)
  - 位元運算子: 以位元(bit)為單位的運算符號，一般情況較少使用，此處不特別作介紹
  - 指定運算子
    - 例如: 加等於(**+=**)、乘等於(***=**)
  - 比較運算子:
    - 例如: 大於(**>**)、小於等於(**<=**)
  - 邏輯運算子:
    - 例如: 且(**and**)、或(**or**)、否(**not**)
  - 身分運算子:
    - 例如: 是(**is**)、不是(**is not**)
  - 成員運算子:
    - 例如: 在(**in**)、不在(**not in**)

數學運算子 (Arithmetic Operators)

- 用於基本數學運算之運算子

```
x = 10
y = 12
print(x+y)  # 輸出x + y 的運算結果
z = x + y     # 將x+y的運算結果算出後存入z中
```

- 常用數學運算子整理

運算子	說明	用法	運算結果 (x = 4, y = 3)
+	加法	x + y	7
-	減法	x – y	1
*	乘法	x * y	12
/	除法	x / y	1.33333333…
%	取餘數	x % y	1
**	指數	x ** y	64
//	除法(忽略小數點)	x // y	1

指定運算子 (Assignment Operators)

- 指定符號(=)以及指定符號搭配數學或位元運算子
- 用來賦予變數數值的運算子
- 常用指定運算子整理

運算子	說明	用法	意義同於
=	指定	x = y	x = y
+=	加法後賦值	x += y	x = x + y
-=	減法後賦值	x –= y	x = x – y
*=	乘法後賦值	x *= y	x = x * y
/=	除法後賦值	x /= y	x = x / y
%=	取餘數後賦值	x %= y	x = x % y
**=	指數後賦值	x **= y	x = x ** y
//=	除法(忽略小數點)後賦值	x //= y	x = x // y

比較運算子 (Comparison Operators)

- 用來比較數值的運算子，比較結果為一布林數(即真True或假False)
- 常用來作為條件判斷使用
- 常用比較運算子整理

運算子	說明	用法	運算結果 (x = 4, y = 3)
==	等於	x == y	False
!=	不等於	x != y	True
>	大於	x > y	True
<	小於	x < y	False
>=	大於等於	x >= y	True
<=	小於等於	x <= y	False

邏輯運算子 (Logical Operators)

- 用來結合條件語句的運算子，運算結果為一布林數
- 常搭配比較運算子來作為條件判斷使用
- 邏輯運算子整理

運算子	說明	用法	運算結果 (x = 4, y = 3)
and	且 (若兩條件語句皆為真則為真)	x < 10 and y < 2	False
or	或 (若兩條件語句有一為真則為真)	x < 10 or y < 2	True
not	非 (反轉結果，語句為真則為假， 語句為假則為真)	not (x < 10 and y < 2)	True

身分運算子 (Identity Operators)

- 用來辨別兩變數是不是同一個物件
- 兩變數相等並不代表是同一個物件
  - 參考範例:

```
x = 10
y = 10
print(x == y)  # 輸出結果: True
print(x is y)   # 輸出結果: False
```

- 身分運算子整理

運算子	說明	用法	運算結果 (x = [1, 2], y = [1,2], z = x)
is	若兩變數為相同物件，則回傳真，反之回傳假	x is z	True
is not	若兩變數並非相同物件，則回傳真，反之回傳假	x is not y	True

成員運算子 (Membership Operators)

- 用來判斷特定序列是否在特定物件中
- 常搭配串列(List)等群集之資料型態使用
- 成員運算子整理

運算子	說明	用法	運算結果 (x = 1, y = [1,2])
in	若前者在後者中，則回傳真， 反之回傳假	x in y	True
not in	若前者不在後者中，則回傳真， 反之回傳假	x not in y	False

縮排 (Indentation)

- 縮排指的是在一行程式的最前面的空白，用來區分程式區段 (block)
- 良好的縮排習慣可以大大增加程式的可讀性
- Python規定必須要縮排，否則會有錯誤訊息!
- 同一份Python程式碼的縮排大小(一次縮排所使用的空格數量)必須一樣!
- 可以使用Tab鍵或者空白鍵來進行縮排，但是不可以混用!
  - 大部分編輯器都可以設定把Tab鍵替換為固定數量的空白鍵，常用數量是 2 和 4，可以依照個人喜好選擇

```
# 這是有縮排的程式
a = 10
b = 12
if a > b:
    print('a比較大')
```

```
# 這是沒有縮排的程式
a = 10
b = 12
if a > b:
print('a比較大')
```

流程控制: 條件判斷 if ... elif ... else

- if 常搭配邏輯敘述來控制程式流程，而 elif 和 else 必須搭配 if 來使用
- 使用方法:

```
if 條件1 :
    # 條件1符合時，欲執行之程式片段
elif 條件2 :
    # 條件1不符，但符合條件2時，欲執行之程式片段
else :
    # 條件1和2皆不符，欲執行之程式片段
```

elif 和 else 可視需求選擇都用、擇一使用或都不要用

- 中文邏輯翻譯:
  - 如果條件1符合就....，不然如果條件2符合就....，不然就....

- 範例程式:

```
a = 10
b = 12
if a > b:
    print('a比較大')
elif a < b :
    print('b比較大')
else :
    print('a和b一樣大')
```

流程控制: 巢狀式條件判斷 Nested if

- 在 if 執行的程式片段中再加入 if 即為巢狀式條件判斷
- 觀察範例程式，可以發現縮排使程式結構更清楚了!
- 要幾層都是可以的，不限於兩層唷!

```
# 這是一支評估體脂率的程式
sex = input("請問您的性別是?(男/女) ")
BFP = input("請問您的體脂率是(%)? ")
BFP = float(BFP)
# 判斷體脂率範圍
if sex == '男':
    if BFP < 17:
        print('評估結果: 過低')
    elif BFP < 23:
        print('評估結果: 正常')
    else:
        print('評估結果: 過高')
else:
    if BFP < 20:
        print('評估結果: 過低')
    elif BFP < 27:
        print('評估結果: 正常')
    else:
        print('評估結果: 過高')
```

第一層

第二層

第二層

流程控制: while 迴圈

- 用來重複執行片段的程式，直到給予的條件不符合為止
- 使用方法:

```
while 條件 :
    # 條件符合時，欲執行之程式片段
```

- 檢查條件部分是真(True)還是假(False)
  - 若為真，則執行所包含的程式片段
  - 若為假，擇跳出迴圈
- 每執行完一次所包含的程式片段，則會再檢查條件部分，決定是否繼續執行
- 要記得在所包含的程式片段內更新條件相關的變數，不然可能會變成無窮迴圈唷!
- 利用縮排決定迴圈所包含的程式片段!

```
# 這支程式會輸出要求次數的文字
times = input("請問要輸出幾次? ")
times = int(times)
i = 0
while i < times :
    i = i + 1
    print('這是第', i, '次')
print('這行不算在迴圈內!')
```

流程控制: break 和 continue

- 在迴圈進行時，可能會有些情況下會想要:
  - 跳出迴圈 ➡ **break**
  - 中斷此次迴圈執行，繼續下次的迴圈運行 ➡ **continue**
- 一般都會搭配 **if** 來使用，使其在特定的條件下才發揮作用

```
# 這支程式使用break
times = input("請問要輸出幾次? ")
times = int(times)
i = 0
while i < times :
    i = i + 1
    if i == 4:
        break
    print('這是第', i, '次')
print('程式結束了!')
```

```
請問要輸出幾次? 5
這是第 1 次
這是第 2 次
這是第 3 次
程式結束了!
```

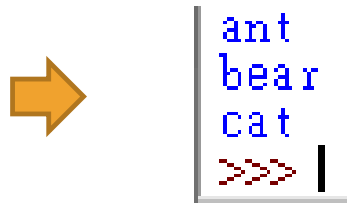
```
# 這支程式使用continue
times = input("請問要輸出幾次? ")
times = int(times)
i = 0
while i < times :
    i = i + 1
    if i == 4:
        continue
    print('這是第', i, '次')
print('程式結束了!')
```

```
請問要輸出幾次? 5
這是第 1 次
這是第 2 次
這是第 3 次
這是第 5 次
程式結束了!
```

流程控制: for 迴圈

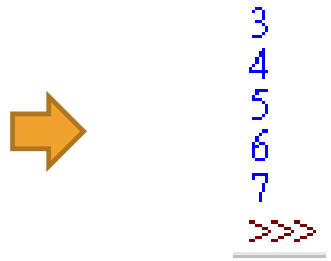
- 用來依序遍歷整個序列的迴圈，亦可搭配 **break** 和 **continue** 使用
  - 此處的序列是指 **list, tuple, dictionary, set, string** 等資料型態

```
animals = ["ant", "bear", "cat"]
for x in animals:
    print(x)
```



- 亦可搭配 **range()** 這個函式使用來控制迴圈
  - 跑特定區段

```
for x in range(3, 8):
    print(x)
```



  - 以特定間距跑特定區段

```
for x in range(3, 8, 2):
    print(x)
```