勞動部產業新尖兵計畫

人工智慧金融應用與實務培訓班

課程模組： AI 金融科技課程 - Python 程式設計

2. 程式語言的概念

葉建華 (Yeh, Jian-hua)

tdi.jhyeh@tdi.edu.tw
au4290@gmail.com

講次內容

- 程式的符號
- 變數設定與資料型別
- 運算符號
- 物件概念

程式語言就如同口說語言一樣，運用正確的「字彙」（關鍵字）和正確的文法（程式語法），產出合規的「語句」（敘述句）。也依照語言框架，將多個語句組合成一篇文章（程式）。只要都合乎語言規則，就應該可以產生唯一表達內容的文章，同理，只要利用標準程式語言規則編寫程式，電腦應該都能正確的執行，產出相同的結果。

程式語言不像自然語言 ...

「研表究明，漢字序順並不定一影閱響讀。比如當你看完這句話后，才發這現里的字全是都亂的。」

程式的符號

- 程式只是一堆文數字（關鍵字、敘述句）、標記組合而成的檔案（程式）
- 和一般文字檔案不同的是 : 程式中的文字的組成包含了特定的原則，就是程式撰寫的「語法 (Syntax) 」
- 依照正確語法排列而成的文字才能被 Python 直譯器所接受，正確執行

程式的符號

- 程式語言都會包含以下的組成元件
  - 識別字
  - 關鍵字（或保留字）
  - 資料常數
  - 符號 ... 等

識別字

- 識別字是指變數 (Variable)、函數 (Function)、類別 (Class) 的名稱
- 識別字命名的原則
  - 第一個字只能或是 _ 或是 字母 (Letter)
  - 第二個字起可以有數字
  - 識別字中不能有標點符號、空白 , 或是 -
  - 關鍵字不可以當作是識別字
- 那 ... 可不可以用中文當做是識別字的名稱？

關鍵字（或保留字）

- 程式語言本身定義的識別字即是「關鍵字」或「保留字」

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

不用特地去背，你會慢慢記得的

資料常數 (Literals)

- 本身的定義不會再更改的資料內容
  - 例如 : 數字「 1 」代表的意義不會再重新更改
- 字串常數
  - 單行字串： "Aman" , '12345'
  - 多行字串：使用倒斜線，或是三重引號 (''')

資料常數 (Literals)

- 數值常數
  - 整數 (Int)： 正負數，不帶指數標記部份
  - 浮點數 (Float)： 包含小數或是指數標記部份的數字
  - 複數 (Complex)： a+bj 形式， 5.67+3.14j

整數常數！！

- 0b10100 （二進位常數）
- 100 （十進位常數）
- 0o215 （八進位常數）
- 0x12d （十六進位常數）

不要太在意，真正用上了再記

符號

- 程式中的符號多半是運算符號
  - 例如：+ 、 - 、 * 、 / 、 = ... 等等
- 也可能是分隔符號
  - 例如「小括號（ ( ) 」、「中括號［ [ ] 」、「大括號｛ { } 」、「分號（ ;) 」、「句號（ .) 」

使用分號被視為「沒有 Python 風格」 (Not Pythonic)

運算符號

- 基本四則運算： + - * / %
- 運算特性：以資料型態 (?) 複雜者為主且運算精確性有限
  - 10/3 、 30*0.5 、 25+4j-3j
- 特殊運算：次方 (**)、整商除法 (//)

講次內容

- 程式的符號
- 變數設定與資料型別
- 運算符號
- 物件概念

變數的設定

- 變數 (Variable) 的主要目的是在記憶體中保留特定大小的空間，讓程式執行時可以暫時儲存資料
- 變數的資料型態 (?) 是動態的，會根據指定時的物件來決定型別
  - 物件才具有資料型態

表示式 (Expression)

- 4 + 9
- 3 + (9 * 4)
- 4**3
- ‘Hello’ + ‘World’

變數的指定（賦值運算）

- a = 4 + 9
- b = a * 3.6
- c = (a+b)/7.5
- s1 = “Hello World”
- 變數單純只是物件的名稱，所以 ...
  - Python 的變數是可以更換型態的

賦值運算 (Assignment)

- 賦值運算是最常被使用的運算
- 將等號右邊的運算結果賦予（指定給）左邊的一個變數
  - 不是左右兩邊相等哦！

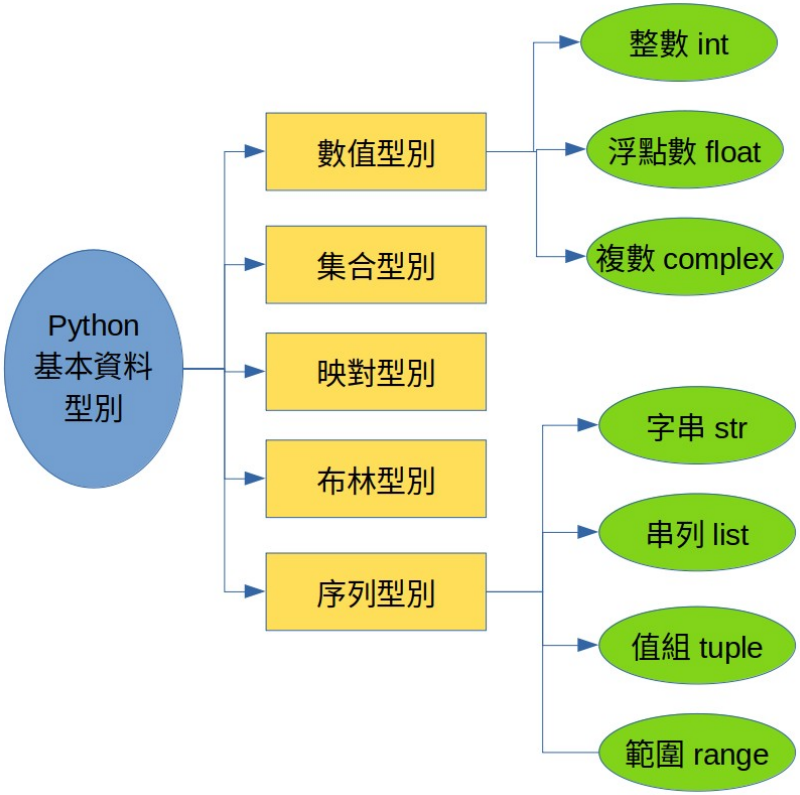
變數的有效範圍（Scope)

- 指在程式中可以使用某變數的地方
- 區域變數（Local Variable)：宣告在函數中，在執行區段中產生，離開區段後失效
- 內區域中可以使用外區段中的變數

資料型別

- 內建的基本資料型別
  - 數值型別： int, float, complex
  - 序列型別： list, tuple, range, str (文字型別)
  - 集合型別： set, frozenset
  - 映對型別： dict
  - 布林型別： bool

你可以使用 type() 函數來查驗資料的型別 !!



數值型別

- 整數、浮點數、複數
- type()、 isinstance()

```
a = 5
print("a的資料型態: ", type(a))

b = 40.5
print("b的資料型態: ", type(b))

c = 1+3j
print("c的資料型態: ", type(c))
print("c是否為複數? ", isinstance(1+3j,complex))
```

序列型別 - 字串

- 字串型態 str

```
str1 = "使用雙引號表示的字串"
print(str1)
str2 = '使用單引號表示的字串'
print(str2)
str3 = '''''使用三單引號表示的
多行字串'''
print(str3)

str1 = '哈囉大家好! '   # str1字串
str2 = ' 你好嗎? '      # str2字串
print(str1[0:2])     # 使用字元索引切分取出子字串:  [起點:終點(不計入)]
print(str2[3])       # 使用字元索引取出特定字元
print(str1*2)        # 字串印兩次!
print(str1 + str2)   # 字串加法? 就是字串合併
```

序列型別 - 字串

- 字串物件一旦建立，內容就無法再更改 (immutable) 了
- 即使將該字串變數重新指定或運算出新的內容，Python 的運作方式是建立一個新的字串物件，再將新的字串物件指定給原變數

序列型別 - 串列

- 串列型態 list

```
list1 = [1, "hi", "Python", 2]    # 串列內容可以是任何東西，包括串列自己
print("list1的資料型態: ", type(list1))

print(list1)    # 印出list1的內容
# 使用串列索引切分取出子串列:  [起點:終點(不計入)]，終點不寫? 那就是算到最後
print(list1[3:])
# 使用串列索引切分取出子串列:  [起點:終點(不計入)]
print(list1[0:2])
print(list1 + list1)     # 串列加法? 就是串列合併
print(list1 * 3)         # 串列內容印三次!
list1[2] = "Java"        # 設定list1特定位置的內容值
print(list1)             # 再印出list1的內容
```

序列型別 - 值組

- 值組型態 tuple： 類似 list，但內容不可改

```
tup1 = ("hi", "Python", 2)
print("tup1的資料型態: ", type(tup1))

print(tup1)     # 印出tup1的內容
# 使用值組索引切分取出子值組:  [起點:終點(不計入)]，終點不寫? 那就是算到最後
print(tup1[1:])
# 使用值組索引切分取出子值組:  [起點:終點(不計入)]
print(tup1[0:1])
print(tup1 + tup1)        # 值組加法? 就是值組合併
print(tup1 * 3)           # 值組內容印三次!
# 設定tup1特定位置的內容值，這會造成錯誤!!
tup1[2] = "hi"
```

序列型別 - 範圍

- 範圍型態 range： 表示不可變的數字序列
  - 常用來配合 for 迴圈

```
range1 = range(6)
for i in range1:
    print(i)

# 再來一個!
range2 = range(4, 7)
for i in range2:
    print(i)

# 再來一個!!
range3 = range(5, 10, 2)
for i in range3:
    print(i)
```

下一講次我們會提到

集合型別

- 集合型別 set： 儲存的元素不可以重複，且沒有順序性

```
set1 = set()      # 建立一個空集合
set2 = {'James', 2, 3,'Python'}
print("set1的資料型態: ", type(set1))

print(set2)       # 印出set2的內容
set2.add(10)      # set2加入元素10
print(set2)       # 再印出set2的內容(為什麼10不在尾端)
set2.add(2)       # set2加入元素2
print(set2)       # 再印出set2的內容(為什麼2無法加入?)

set2.remove(2)    # set2移除指定元素2
print(set2)       # 再印出set2的內容
```

映對型別

- 辭典型別 dict： 使用「鍵值 - 資料值」配對存取 (key, value)

```
dic1 = {1:'Jimmy', 2:'Alex', '3':'john', 4:'mike'}
print("dic1的資料型態: ", type(dic1))

print(dic1)              # 印出dic1的內容 => 「鍵值-資料」值配對
print("鍵值1對應的資料值: "+ dic1[1])     # 使用鍵值1來取出資料值
print("鍵值4對應的資料值: "+ dic1[4])     # 使用鍵值4來取出資料值
print("鍵值3對應的資料值: "+ dic1[3])     # 使用鍵值3來取出資料值...錯!！為什麼?
print("鍵值\'3\'對應的資料值: "+ dic1['3'])     # 使用鍵值'3'來取出資料值

print(dic1.keys())       # 取出所有的鍵值
print(dic1.values())     # 取出所有的資料值
```

映對型別

- 辭典型態 dict： 所有的鍵值形成集合 (set)，特性如同 set 一樣
  - 也就是說，鍵值不可以重複，且沒有順序性

布林型別

- 用來表示真假值
- 布林常數： True （真）、 False （假）

注意大小寫！

```
print(10 > 9)      # 天經地義
print(10 == 9)     # 怎麼可能
print(10 < 9)      # 亂來!

a = 200
b = 33

if b > a:          # 條件分支敘述
  print("b大於a")
else:
  print("a大於b")
```

特別的型別值： None

- None 用來代表：沒有、尚未定義、無效
  - 就是「空值」
  - 其他程式語言會稱為 Null, null, Nil, nil 等等
- 型別為 NoneType

型別轉換 (Casting)

- Python 是物件式的程式語言，型別轉換是使用重新建構物件的方式來完成

```
x = int(1)       # x值為1
y = int(2.8)     # y值為2
z = int("3")     # z值為3
```

```
x = str("s1")    # x值為's1'
y = str(2)       # x值為'2'
z = str(3.0)     # z值為'3.0'
```

```
x = float(1)     # x值為1.0
y = float(2.8)   # y值為2.8
z = float("3")   # z值為3.0
q = float("4.2") # w值為4.2
```

複雜的資料型別會有專門的段落介紹

講次內容

- 程式的符號
- 變數設定與資料型別
- 運算符號
- 物件概念

運算符號

- 運算式：「運算子 (Operator)」和「運算元 (Operand)」組合而成
  - 「運算元」代表資料的表示方式
  - 「運算子」決定資料的運算方式，又稱為「運算符號」
  - c = a + b => 「a」、「b」、「c」代表運算元，「+」、「=」是運算子

運算符號

- 算術運算子
- 指定運算子
- 關係運算子
- 邏輯運算子
- 位元運算子
- 成員運算子
- 身份運算子

算術運算子

運算子	說明
+	加法運算。 x+y (3+2 得 5)
-	減法運算，或是負號。 x-y (3-2 得 1) 或是 -x
*	乘法運算。 x*y (3*2 得 6)
/	除法運算。 x/y (3/2 得 1.5)
%	餘數運算。 x%y (3%2 得 1)
**	指數乘方運算。 x**y (3**2 得 9)
//	整商運算。 x//y (3//2 得 1)

關係運算子

運算子	說明
==	如果兩邊運算元相等，則狀態為 True。 3==3 真， 3==2 為假
!=	如果兩邊運算元不等，則狀態為 True。 3!=2 真， 3!=3 為假
<=	如果第一運算元小於或等於第二運算元，則狀態為 True。
>=	如果第一運算元大於或等於第二運算元，則狀態為 True。
<	如果第一運算元小於第二運算元，則狀態為 True。
>	如果第一運算元大於第二運算元，則狀態為 True。

邏輯運算子

運算子	說明
and	且 (AND) 運算，依據真值表運作
or	或 (OR) 運算，依據真值表運作
not	否 (NOT) 運算，依據真值表運作

真值表在此!
0: 假， 1: 真

&AND

A AND B	A=0	A=1
B=0	0	0
B=1	0	1

|OR

A OR B	A=0	A=1
B=0	0	1
B=1	1	1

^ XOR (excusive OR)

A XOR B	A=0	A=1
B=0	0	1
B=1	1	0

!NOT

A	A=0	A=1
NOT A	1	0

那 ... 有沒有人知道 xor 要怎麼辦?

位元運算子

運算子	說明
&	輸入的兩個位元都是「1」時，會產生「1」的輸出位元
|	輸入的兩個位元都是「0」時，會產生「0」的輸出位元，其餘的情形都會輸出「1」
^	輸入的兩個位元不相同時，會產生「1」的輸出位元，位元相同的情形會輸出「0」
~	將數值的位元反向
<<	算數位元左平移
>>	算數位元右平移

指定運算子

運算子	說明
=	將右邊的運算結果，指定給左邊的變數。 x=10
+=	將左邊變數的值的加上右邊的值，再指定給左邊的變數。 x+=10
-=	將左邊變數的值的減去右邊的值，再指定給左邊的變數。 x-=10
*=	將左邊變數的值的乘以右邊的值，再指定給左邊的變數。 x*=10
/=	將左邊變數的值的除以右邊的值，再指定給左邊的變數。 x/=10
%=	將左邊變數的值的除以右邊的值，再指定餘數給左邊的變數
**=	將左邊變數的值以右邊的值做指數乘方，再指定給左邊的變數
//=	將左邊變數的值以右邊的值做整商除法，再指定給左邊的變數

指定運算子

運算子	說明
&=	將左邊變數的值和右邊的值進行 & 運算，再指定給左邊的變數
|=	將左邊變數的值和右邊的值進行 | 運算，再指定給左邊的變數
^=	將左邊變數的值和右邊的值進行 ^ 運算，再指定給左邊的變數
>>=	將左邊變數的值以右邊的值做位元右平移，再指定給左邊的變數
<<=	將左邊變數的值以右邊的值做位元左平移，再指定給左邊的變數

成員運算子

運算子	說明
in	第一運算元是第二運算元的元素時為真 (list, tuple, dict)
not in	第一運算元不是第二運算元的元素時為真 (list, tuple, dict)

這不就是 in 套用邏輯運算子 not 嗎？！

身份運算子

運算子	說明
is	如果第一運算子和第二運算子指稱的物件相同，則為真
is not	如果第一運算子和第二運算子指稱的物件不同，則為真

這不就是 is 套用邏輯運算子 not 嗎？！

運算子是有執行順序的!

1. **	8. <= < > >=
2. ~ + - (是正負號哦)	9. <> == !=
3. * / % //	10. = %= /= //= -= += *= **=
4. + -	11. is is not
5. >> <<	12. in not in
6. &	13. not or and
7. ^ |	

很難記? 不用特地去記，寫多就記得了。不然 ... 就用括號區分先後吧!

講次內容

- 程式的符號
- 變數設定與資料型別
- 運算符號
- 物件概念

何謂物件?

Python 是以物件為根基的物件導向程式語言，而物件使用的基本概念是：「物件使用前，必需先要實體化」。變數僅只是配置了存取物件的「參考(reference)」，並未配置該物件的實際儲存空間，物件必需「實體化」後才會有記憶體空間

何謂物件?

- 物件也具有生命週期，不再使用時，它原先所佔用的系統資源必需回收
- Python 有資源回收機制 GC (Garbage Collection)

萬物都是物件

- 還記得基本資料型別嗎？它們都是物件

```
x = int(1)          # x值為1
y = int(2.8)        # y值為2
z = int("3")        # z值為3

x = float(1)        # x值為1.0
y = float(2.8)      # y值為2.8
z = float("3")      # z值為3.0
q = float("4.2")    # w值為4.2

x = str("s1")       # x值為's1'
y = str(2)          # x值為'2'
z = str(3.0)        # z值為'3.0'
```

```
list1 = list()
list1.append('A')
list1.append(2)
list1.append(False)
print(list1)

tup1 = tuple((1,2,3))
print(tup1)

dic1 = dict()
dic1['A'] = 'apple'
dic1['B'] = 'banana'
print(dic1)
```

都有實體化哦!

這個講次中，你應該學到了 ...

- 程式的符號
- 變數表示和操作
- 資料型別和操作
- 運算式的表示和操作
- 物件的基本概念