



資訊科技延伸學習:Python

台北市立中崙高中 10908陳莉軒



摘要

在我們的日常生活中，往往充滿著資訊科技的身影。或許你不曾注意，但青少年的我們平時離不開的Instagram、不管男女老少都在使用的LINE，處處皆與資訊科技有著說不清的連結。

在這個學期接觸到Python語言後，我決定透過老師介紹的自學網站DataCamp進行延伸學習。在學習的過程中的確遇到了困難，但也使我收穫良多，對資訊科技有了更多的了解。

內文

◆ 動機

在這個充滿科技的世代，具備一定的資訊能力是我們的重要的課題。我在資訊課接觸了python這個語言，感覺十分有趣，因此希望能對這個語言有更深入的了解。

◆ 課程摘要

在這學期的資訊課中，老師帶領我們運用DataCamp這個網站來入門Python，也在課程中教導我們關於陣列及迴圈的應用

◆ 課堂學習成果

1.DataCamp

透過與小組成員互相討論及指導，讓我們透過DataCamp對Python進行初步了解。

Python Basics <small>FREE</small>		100%
An introduction to the basic concepts of Python. Learn how to use Python interactively and by using a script. Create your first variables and acquaint yourself with Python's basic data types.		
✓ Hello Python!		✓ 50 XP
• The Python Interface		✓ 100 XP
• When to use Python?		✓ 50 XP
• Any comments?		✓ 100 XP
• Python as a calculator		✓ 100 XP
• Variables and Types		✓ 50 XP
• Variable Assignment		✓ 100 XP
• Calculations with variables		✓ 100 XP
• Other variable types		70 XP
• Guess the type		✓ 50 XP
• Operations with other types		✓ 100 XP
• Type conversion		✓ 100 XP
• Can Python handle everything?		✓ 50 XP

這是對Python運算簡單的認識：

+、-、*、分別代表加、減、乘、除；而%則是取餘數；**是次方的意思。

```
1 # Addition, subtraction
2 print(5 + 5)
3 print(5 - 5)
4
5 # Multiplication, division, modulo, and exponentiation
6 print(3 * 5)
7 print(10 / 2)
8 print(18 % 7)
9 print(4 ** 2)
10
11
12
13
14
15
16
```

這是把字串帶入運算的練習：

我們先建立字串savings、growth_multiplier(類似利息)並定義它的值，接著建立一個字串result，這個字串如題目所述，是7年後總共會得到多少錢
`=savings*growth_multiplier**7`

最後再輸出result

- Create a variable `growth_multiplier`, equal to `1.1`.
- Create a variable, `result`, equal to the amount of money you saved after `7` years.
- Print out the value of `result`.

```
1 # Create a variable savings
2 savings = 100
3
4 # Create a variable growth_multiplier
5 growth_multiplier=1.1
6
7 # Calculate result
8 result=savings*growth_multiplier**7
9
10 # Print out result
11 print(result)
```

這是對`type()`指令的練習。

`type()`指令:是一個內建的函數，用它就能夠得到一個返回值，從而知道想要查詢的對象類型信息。如右圖的題目，要得出a、b、c分別為甚麼類型的信息，我們就可以利用`type()`指令來得到答案。

We already went ahead and created three variables: `a`, `b` and `c`. You can use the IPython shell to discover their type. Which of the following options is correct?

```
In [1]: print(type(a))
<class 'float'>

In [2]: print(type(b))
<class 'str'>

In [3]: print(type(c))
<class 'bool'>
```

2. Google Colab

透過習題的方式，老師帶領我們認識陣列、迴圈及其應用。

右圖是對**索引值**認識的練習，在每一個**陣列**中，由左到右的索引值=0.1.2... $(n-1)$ ，而由右到左的索引值=-1.-2... $-n$ 。可以發現，從不同的方向開始，索引值也會不同，其中由右邊開始數的索引值開頭為-1，是因為，若右邊的開頭也為0的話，則 $0=0$ ，會搞混。

這是對索引值進行改變的練習。

`x=score_array[0]`

這是在定義x的值=85。

`score_array[2]=100000`

則是在改變`score_array[2]`的值，由88改成了100000，而第12行也是同理。

由這個練習我們可以知道，陣列的值是可以修改的。

```
1 #課本程式碼
2 #----- P.61(Ch4-1.py) -----
3 score_array = [85,92,88,96]
4 animals = ['長頸鹿','獅子','老虎','河馬','熊','兔子','斑馬']
5
6 print(score_array)
7 print(animals)
8
9 print(score_array[3])
10 print(animals[5])
11 #-----
[85, 92, 88, 96]
['長頸鹿', '獅子', '老虎', '河馬', '熊', '兔子', '斑馬']
96
兔子
```

```
1 #課本程式碼
2 #----- P.61(Ch4-2.py) -----
3 score_array = [85,92,88,96]
4 x=score_array[0]
5 print(x)
6
7 score_array[2]=100000
8 x=score_array[2]
9 print(x)
10
11
12 score_array[2]=30000
13 print(score_array)
14
15 #-----
85
100000
[85, 92, 30000, 96]
```

這是對**二維陣列**認識的練習。

二維陣列，和上述的一維陣列不同，形式如右圖：

二維陣列=`[[],[],...[]]`

在中括號中還有中括號，其實可以理解成二維陣列是由許多一維陣列所組成的陣列。

而`print(animals[1][2])`

，而`[1][2]`都是索引值的順序去尋找，從而得出「紅蘿蔔」。

```
1 #課本程式碼
2 #----- P.63 -----
3 num_2d_array = [
4     [1,2,3],
5     [4,5,6],
6     [7,8,9]
7 ]
8 animals = [
9     0 ['長頸鹿','獅子','兔子'],
10    1 ['樹葉','肉','紅蘿蔔']
11 ]
12 print(num_2d_array)
13 print(num_2d_array[1][2])
14
15 #印出"紅蘿蔔"
16 print(animals)
17 print(animals[1][2])
18 #-----
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
6
[['長頸鹿', '獅子', '兔子'], ['樹葉', '肉', '紅蘿蔔']]
紅蘿蔔
```

這是對二維陣列加上for迴圈的練習。

for迴圈(以第6行做示範):

```
for i in range(2):  
    print(members[i])
```

(2)是指從0開始跑到1，跑兩個索引值，而[i]則代表索引值，可以看成:

```
print(members[0])
```

```
print(members[1])
```

這個練習可以和上面做比較:

雙層迴圈(以第6行做示範):

```
For i in range(2):  
    for j in range(4):  
        print(members[i][j])
```

雙層迴圈可以看成迴圈中還有迴圈，其實就是設定有i、j兩個索引值在跑。當i=0時，j會從0~3都跑一遍，才會又回到i=1，然後j繼續從0~3跑一遍，直到i=1,j=3,也就是‘Beth’，迴圈結束。

➤ 單層迴圈和雙層迴圈的比較:

名稱	單層迴圈	雙層迴圈
呈現效果	陣列單獨成行	陣列中的物件單獨成行
寫法	<pre>for i in range(2): print(members[i])</pre>	<pre>for i in range(2): for j in range(4): print(members[i][j])</pre>

```
1 #----- Q2:你的程式碼 -----  
2 members=[  
3     0 ["John", "Willson", "Alan", "Edward"],  
4     1 ["Shierly", "Rina", "Phoebe", "Beth"]  
5 ]  
6 for i in range(2):  
7     print(members[i])  
8  
9  
10 #-----  
["John", "Willson", "Alan", "Edward"]  
["Shierly", "Rina", "Phoebe", "Beth"]
```

```
1 #----- Q3:你的程式碼 -----  
2 members=[  
3     0 1 2 3 ["John", "Willson", "Alan", "Edward"],  
4     1 0 1 2 3 ["Shierly", "Rina", "Phoebe", "Beth"]  
5 ]  
6 for i in range(2):  
7     for j in range(4):  
8         print(members[i][j])  
9  
10  
11  
12 #-----  
John  
Willson  
Alan  
Edward  
Shierly  
Rina  
Phoebe  
Beth
```

◆ 課堂延伸學習成果

我利用課餘時間完成
DataCamp的Python剩下的
的入門課程，以下是我學習的
成果。

右圖是對list的初步認識。我們
先定義列表中物件的值，再將
列表輸出。從輸出結果我們可
以知道，當物件被()包起來的
時候，在list輸出時，會輸出它
被定義的值。

這個練習讓我們了解了不同類
型的list，從輸出結果可以看到
['hallway' ,11.25,' kitch
en' ,...,9.5]我們就可以觀察
到文字放入list並透過' ' 包起
來也可以被正常輸出。

下面的練習則關於子集的運算，
題目是讓我們先創造一個新的
值eat_sleep_area，然後這個
值是bedroom和kitchen的和。
所以可以列出：

```
eat_sleep_area=[areas[3]+  
area[-3]]
```

配合老師在堂上介紹過的索引
值，最後輸出的答案可以配合
上一題的數據確認是沒有錯的。

Python Lists FREE

100%

Learn to store, access, and manipulate data in lists: the first step toward efficiently working with huge amounts of data.

VIEW CHAPTER DETAILS ▾

✓ Completed

Functions and Packages FREE

100%

You'll learn how to use functions, methods, and packages to efficiently leverage the code that brilliant Python developers have written. The goal is to reduce the amount of code you need to solve challenging problems!

VIEW CHAPTER DETAILS ▾

✓ Completed

NumPy FREE

100%

NumPy is a fundamental Python package to efficiently practice data science. Learn to work with powerful tools in the NumPy array, and get started with data exploration.

VIEW CHAPTER DETAILS ▾

✓ Completed

- Create a list, `areas`, that contains the area of the hallway (`hall`), kitchen (`kit`), living room (`liv`), bedroom (`bed`) and bathroom (`bath`), in this order. Use the predefined variables.
- Print `areas` with the `print()` function.

```
1 # area variables (in square meters)
2 hall = 11.25
3 kit = 18.0
4 liv = 20.0
5 bed = 10.75
6 bath = 9.50
7
8 # Create list areas
9 areas = [hall, kit, liv, bed, bath]
10
11 # Print areas
12 print(areas)
13
```

```
[11.25, 18.0, 20.0, 10.75, 9.5]
```

- Finish the code that creates the `areas` list. Build the list so that the list first contains the name of each room as a string and then its area. In other words, add the strings `"hallway"`, `"kitchen"` and `"bedroom"` at the appropriate locations.
- Print `areas` again; is the printout more informative this time?

```
1 # area variables (in square meters)
2 hall = 11.25
3 kit = 18.0
4 liv = 20.0
5 bed = 10.75
6 bath = 9.50
7
8 # Adapt list areas
9 areas = ["hallway", hall, "kitchen", kit, "living room", liv, "bedroom", bed, "bathroom", bath]
10
11 # Print areas
12 print(areas)
```

```
['hallway', 11.25, 'kitchen', 18.0, 'living room', 20.0, 'bedroom', 10.75, 'bathroom', 9.5]
```

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]
3
4 # Sum of kitchen and bedroom area: eat_sleep_area
5 eat_sleep_area = areas[3] + areas[-1]
6
7 # Print the variable eat_sleep_area
8 print(eat_sleep_area)
```

```
28.75
```

右圖是對list增加項目的練習，
可以觀察到輸出結果，由上而下分別是：

areas, areas_1, areas_2。

而從第7行和第9行就可以得知
要如何在list中增加項目：

list' =list+[要增加的項目]

這是刪除list中的項目，我們可
以看到，在第1行中，areas中
的第二項[1]是11.25，但在輸
出時的第二項卻是' kitchen'
，所以原本的第二項被刪除了。
因此我們可以得知要如何在list
中刪除項目：

del(list[項目的引索值])

接著是一些list的功能：

第7行的sorted：

sorted(____, reverse=____)

sorted就是對list作排列，而
reverse則是決定升序、降序
的排列，reverse = True 降序，
reverse = False 升序。

第11行的len()：

Len則是測量list的長度，就是
list包含了幾個項目的意思。

而從輸出結果來看，

var1_sorted確實是依照降序
排列，而var1也的確包含4個
項目。

```
1 # Create the areas list and make some changes
2 areas = ["hallway", 11.25, "kitchen", 18.0, "chill zone", 20.0,
3         "bedroom", 10.75, "bathroom", 10.50]
4
5 # Add poolhouse data to areas, new list is areas_1
6 areas_1=areas+['poolhouse',24.5]
7
8 # Add garage data to areas_1, new list is areas_2
9 areas_2=areas_1+['garage',15.45]
```

```
['hallway', 11.25, 'kitchen', 18.0, 'chill zone', 20.0, 'bedroom', 10.75, 'bathroom', 10.5]
['hallway', 11.25, 'kitchen', 18.0, 'chill zone', 20.0, 'bedroom', 10.75, 'bathroom', 10.5, 'poolhouse', 24.5]
['hallway', 11.25, 'kitchen', 18.0, 'chill zone', 20.0, 'bedroom', 10.75, 'bathroom', 10.5, 'poolhouse', 24.5, 'garage', 15.45]
```

```
1 areas = ["hallway", 11.25, "kitchen", 18.0,
2         "chill zone", 20.0, "bedroom", 10.75,
3         "bathroom", 10.50, "poolhouse", 24.5,
4         "garage", 15.45]
5 del(areas[1])
6 print(areas)
```

```
['hallway', 'kitchen', 18.0, 'chill zone', 20.0, 'bedroom', 10.75, 'bathroom', 10.5, 'poolhouse', 24.5, 'garage', 15.45]
```

```
1 # Create variables var1 and
2 var1 = [1, 4, 3, 2]
3
4 # Sort var1 in descending order: var1_sorted
5 var1_sorted=sorted(var1, reverse=True)
6
7 # Print out full_sorted
8 print(var1_sorted)
9
10 # Print out length of var1
11 print(len(var1))
--
[4, 3, 2, 1]
4
```

右圖是Python的一些方法:

第2行的upper():

這個方法將字串裡的字符由小寫轉換成大寫。

可以觀察輸出的第2行，原本的poolhouse被轉換成POOLHOUSE。

第12行的count():

統計在字串中的某一個字符出現的字數。

如第12行就是在計算poolhouse中有幾個o，結果如輸出第三行=3。

count()也可以計算列表中的項目，如右圖的第8行:

print(areas.count(9.50))

就是在統計在areas這個list中有幾個9.50。

右圖的第5行index():

尋找在list中某一項目的索引值。

如第五行:

print(areas.index(20.0))

尋找20.0的索引值，而20.0的索引值為2=輸出第1行。

右圖的第5行append():

在list中增加項目。

如輸出第1行，我們可以觀察到，輸出的areas比原本的list多了24.5、15.45兩個項目。

第14行的reverse():

使list中的順序反轉。

如輸出第2行，可以和輸出第1行比較。

```
1 # string to experiment with: place
2 place = "poolhouse"
3
4 # Use upper() on place: place_up
5 place_up=place.upper()
6
7 # Print out place and place_up
8 print(place)
9 print(place_up)
10
11 # Print out the number of o's in place
12 print(place.count('o'))

poolhouse
POOLHOUSE
3
```

```
1 # Create list areas
2 areas = ['hallway', 'kitchen', 'living room', '
3
4 # Print out index of element living room
5 print(areas.index('living room'))
6
7 # Print out how often hallways appears in areas
8 print(areas.count('hallway'))
9
2
1
```

```
1 # Create list areas
2 areas = ['hallway', 'kitchen', 'living room', '
3
4 # Use append twice to add poolhouse and garage size
5 areas.append('poolhouse')
6 areas.append('garage')
7
8
9
10 # Print out areas
11 print(areas)
12
13 # Reverse the orders of the elements in areas
14 areas.reverse()
15
16 # Print out areas
17 print(areas)

['hallway', 'kitchen', 'living room', 'poolhouse', 'garage']
['garage', 'poolhouse', 'living room', 'kitchen', 'hallway']
```


接著是Python的package:

右圖是關於計算圓面積及周長的練習。我們先定義半徑 r ，接著匯入(import)套件(package)math，如右圖第5行:

`import math`

接著我們以 C 代表圓周長(Circumference)，先將 C 歸零，在定義為圓周長的公式:

`C=2*math.pi*r`

`math.pi`就是 π ，而`math`就是剛剛匯入的套件，而 π 則要透過匯入這個`math`套件來使用。而圓面積(A)和圓周長(C)的作法是一樣的，最後再將結果匯出即可。

右圖則是針對弧的長度:

弧長= r *弧度

而`radians()`的功能就是將角度轉換成弧度。所以要先匯入`math`這個套件中的`radians`:
`from math import radians`
這個匯入方法不一樣，所以使用時也不一樣，不需要`math.radians()`，而是直接輸入`radians()`即可。

我們定義`dist(distance)`作為弧長:

`dist=R*radians(12)`

最後再輸出即可。

```
1 # Definition of radius
2 r = 0.43
3
4 # Import the math package
5 import math
6
7 # Calculate C
8 C = 0
9 C=2*math.pi*r
10 # Calculate A
11 A = 0
12 A=math.pi*r**2
13 # Build printout
14 print("Circumference: " + str(C))
15 print("Area: " + str(A))
```

Circumference: 2.701769682087222
Area: 0.5808804816487527

```
1 # Definition of radius
2 r = 192500
3
4 # Import radians function of math package
5 from math import radians
6
7 # Travel distance of Moon over 12 degrees. Store in dist.
8 dist=r*radians(12)
9
10 # Print out dist
11 print(dist)
```

40317.10572106901

最後是numpy array，他比list可以做更多的事。

但我們在使用numpy前，需先匯入，如第4行：

```
import numpy
```

numpy比list可以支援更多的數字類型。如第13行，numpy可將整個序列乘上float類型的數字，而list只能乘上整數(如右圖)。

```
1 # height_in is available as a regular list
2
3 # Import numpy
4 import numpy
5
6 # Create a numpy array from height_in: np_height_in
7 np_height_in=numpy.array(height_in)
8
9 # Print out np_height_in
10 print(np_height_in)
11
12 # Convert np_height_in to m: np_height_m
13 np_height_m=np_height_in*0.0254
14
15 # Print np_height_m
16 print(np_height_m)
```

[74 74 72 ... 75 75 73]
[1.8796 1.8796 1.8288 ... 1.905 1.905 1.8542]

```
12 # Convert np_height_in to m: np_height_m
13 np_height_m=np_height_in*0.0254
14 height_list=height_in*0.0254
15 # Print np_height_m
16 print(np_height_m)
17 print(height_list)
```

TypeError: can't multiply sequence by non-int of type 'float'

右圖是與我們日常生活十分貼近的bmi的練習：

這次的匯入又與前幾次不同，如第4行：

```
import numpy as np
```

這樣匯入讓我們之後不需要打numpy.array而是np.array即可。

第7、8、9行則是bmi的計算
第12行：

```
Light=bmi<21
```

這是一個判斷式，所以當我們輸出light時，會有[False,False...,False,False]這樣的結果，因為只要 ≥ 21 即會被輸出為False。

而當我們輸出的是bmi[light]，則會輸出 <21 的資料。

```
1 # height_in and weight_lb are available as a regular lists
2
3 # Import numpy
4 import numpy as np
5
6 # Calculate the BMI: bmi
7 np_height_m = np.array(height_in) * 0.0254
8 np_weight_kg = np.array(weight_lb) * 0.453592
9 bmi = np_weight_kg / np_height_m ** 2
10
11 # Create the light array
12 light=bmi<21
13
14 # Print out light
15 print(light)
16
17 # Print out BMIs of all baseball players whose BMI is below 21
18 print(bmi[light])
```

[False False False ... False False False]
[20.54255679 20.54255679 20.69282047 20.69282047 20.34343189 20.34343189
20.69282047 20.15883472 19.4984471 20.69282047 20.9205219]

這個是關於子集和元素的練習。

同樣的我們要先import numpy。

接著將兩個list轉換成np.array，

如第7、8行。

第11行:

`print(np_weight_lb[50])`

這是在輸出索引值為50的那個元素。

第14行:

`print(np_height_in[100:111])`

這個則是在輸出從索引值為

100~110的子集合。

我認為這個部分和list的做法差

不多，大同小異。

```
1 # height_in and weight_lb are available as a regular lists
2
3 # Import numpy
4 import numpy as np
5
6 # Store weight and height lists as numpy arrays
7 np_weight_lb = np.array(weight_lb)
8 np_height_in = np.array(height_in)
9
10 # Print out the weight at index 50
11 print(np_weight_lb[50])
12
13 # Print out sub-array of np_height_in: index 100 up to and including index 110
14 print(np_height_in[100:111])
```

208
[73 74 72 73 69 72 73 75 75 73 72]

右圖是numpy的二維陣列。同

樣的，要先匯入numpy，接著

一樣將list轉換成np.array。

第14行:

`print(type(np_baseball))`

輸出結果為:

numpy.ndarray(二維陣列)

第17行:

`print(np_baseball.shape)`

則是在描述這個陣列的形狀。

輸出結果(4,2)，4代表行數，

而2則代表列數。

```
1 # Create base... a list of lists
2 baseball = [[188, 78.4],
3             [215, 182.7],
4             [210, 98.5],
5             [188, 75.2]]
6
7 # Import numpy
8 import numpy as np
9
10 # Create a 2D numpy array from baseball: np_baseball
11 np_baseball = np.array(baseball)
12
13 # Print out the type of np_baseball
14 print(type(np_baseball))
15
16 # Print out the shape of np_baseball
17 print(np_baseball.shape)
```

<class 'numpy.ndarray'>
(4, 2)

右圖是提取及索引子集和元素的練習。不可或缺的，一樣先匯入numpy，然後將baseball這個list轉換成np.array。

第10行:

```
print(np_baseball[49,:])
```

這個是在輸出np_baseball的第50行。[49,:]，49是第50行的索引值；而“:”則是用於切段，在這個範例中，他告訴Python這個包含所有列(灰色部分)。

第13行:

```
np_weight_lb=np_baseball[:,1]
```

定義一個新的值，包含

np_baseball的整個第2列。

[:,1]，“:”表示所有的行數，1是第2列的索引值。

第16行:

```
print(np_baseball[123,0])
```

這個則是要輸出第124位運動員的身高。

[123,0]123是第124行的索引值，0則是第1列(身高)的索引值。

右圖是numpy的其他功能。

np.median():取中位數

如右圖，因為我們要取的是身高的中位數，因此先定義一個新的值

```
np_height_in=np_baseball[:,0]
```

最後在輸出時(第10行):

```
print(np.mean(np_height_in))
```

就能得到中位數了。

```
1 # baseball is available as a regular list of lists
2
3 # Import numpy package
4 import numpy as np
5
6 # Create np_baseball (2 cols)
7 np_baseball = np.array(baseball)
8
9 # Print out the 50th row of np_baseball
10 print(np_baseball[49,:])
11
12 # Select the entire second column of np_baseball: np_weight_lb
13 np_weight_lb = np_baseball[:,1]
14 print(np_weight_lb)
15 # Print out height of 124th player
16 print(np_baseball[123,0])
17
18 [ 70 195]
19 [180 215 210 ... 205 190 195]
20
```

列 \ 行	1	2
50[49]	70	195
51[50]	73	200

列 \ 行	1	2
1[0]	74	180
...
1015 [1014]	73	195

```
1 # np_baseball is available
2
3 # Import numpy
4 import numpy as np
5
6 # Create np_height_in from np_baseball
7 np_height_in = np_baseball[:,0]
8
9 # Print out the median of np_height_in
10 print(np.median(np_height_in))
11
12 74.0
```

◆ 遇到的困難

我在完成DataCamp也遇到了一些困難:

1.全英文難理解

因為網站是全英文的，對於我而言，看英文是一件很吃力的事情。但也讓我意識到了自己在英文方面的不足，當我遇到不理解的難題時，我會先自行Google一些相關的資料，如果真的還是不懂或找不到資料時，我會向其他同學請教。

2.抽象難理解

我在學習numpy array那個部分的時候其實有點難理解它的運作規則是甚麼，後來透過自己找資料及圖表化的方式，才能比較清楚這個程式是如何運作的。

◆ 心得與反思

在接觸了Python的冰山一角後，我認為自己對這個語言的了解還不夠深入。學一門程式語言就像探索一大片海洋，而我現在只走到了淺海區，後面還有一片汪洋大海等待我去探索。

我這次學習的是Python語言的入門，學到了Python List、Function和Package的匯入和應用、Numpy的使用。我在學習的過程中遇到很多一時難以理解的程式，像是剛接觸Numpy的二維陣列時，我對於提出子集那個部分也是一頭霧水，後來經過自己查資料、測試和圖表整理，才釐清邏輯。我認為在學習程式時自己動手試試看，是一件很重要的事情。不用擔心做錯，如果錯了，就找出問題點，然後更正。

總而言之，透過這次學習Python語言，讓我對自己不足的部分有了加強的目標，也讓我增強了現代人需要具備的資訊能力。

◆ 資料來源

模板|slidesgo|<https://slidesgo.com/>

內容|DataCamp|<https://www.datacamp.com/>

|資訊老師的講義|

|菜鳥教程|<https://www.runoob.com/>