UNIT 2

python™

Basic Python programs

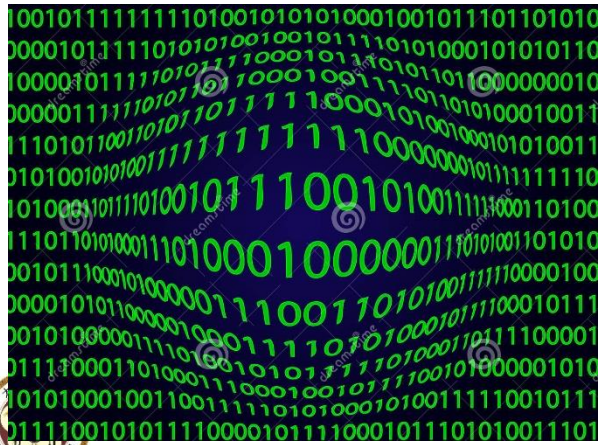
張傑帆 Chang, Jie-Fan

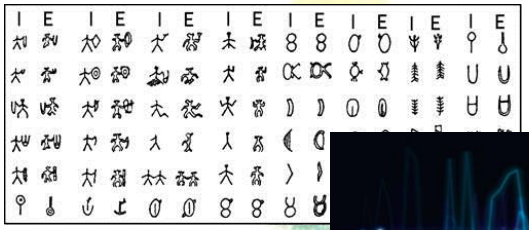
OUTLINE

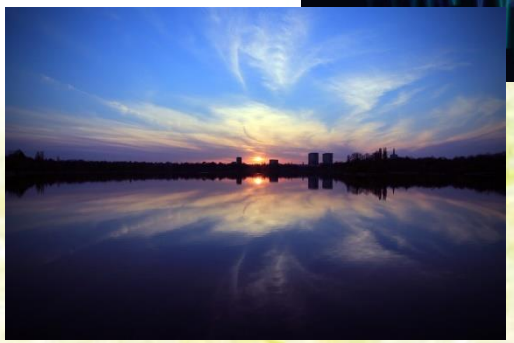
- 編碼
- 資料型態
- 輸出
- 輸入
- 註解
- 縮排

編碼

- 舉凡所有數位的資訊都是由0和1所組合而成的，包括音訊、影像、文字全都是。
- 那為什麼我們在電腦螢幕上看到的不是0跟1呢？
- 那是因為電腦經過「編碼」這個動作，把0跟1轉換成人類所看得懂的文字及前面所提及的聲音與影像等等。







編碼

- 在所有程式語言中，編碼都是相當重要的一環。
- 如果編碼錯誤：
  「鎻愪緵绲﹀湪灝埲キ绋嬪紡瑷悧■甯塲娇鐢ㄧ殑瑾炶■，鍚屾檪鎺〃鐪閬嶄鍪绛栫暐」
- 這樣的文字即為「亂碼」，就是因為編碼錯誤所造成的。

編碼

- 在電腦裡面，繁體中文編碼1 主要分成以下兩種：BIG-5 以及UTF-8。
- 在Python 的世界裡這兩種都支援，但是這裡比較推薦UTF-8。
  - BIG-5 是專門拿來作為繁體中文的編碼方式，對於其他語系的系統相容性較低；
  - UTF-8 是符合Unicode 格式的可變長度字元編碼，可以把它看成是電腦與各種不同語系之間的翻譯橋梁。

編碼

- 如果一支程式可以解讀UTF-8，那麼對於各種語系的解碼也就不成問題了。
- 在**Python 3.x** 版以後的版本也將**UTF-8** 設為**預設編碼**，因此即便不手動執行編碼的動作也可以使用中文；
- 但是如果您想要使用Python 2.x 的版本或者想要使用其他的編碼方式，那麼就必須要另外進行設定了。

編碼

#-*-coding:UTF-8-*-

- 上面看到的這個就是標準的Python 編碼方式，而其詳細說明則分述如下：「#」是開始註解的意思。
- 而註解是給程式撰寫者看的，電腦在執行程式時會自動跳過註解。
- 但是當「#」後面接「coding: 編碼方式」時，整句陳述就變成宣告編碼的意思，此時電腦便不會跳過，而是將其解讀為宣告編碼。

編碼

- 「coding:UTF-8」就是跟電腦說我要使用UTF-8的編碼方式。
  - 當然，如果想要換成其他編碼方式，只要把「UTF-8」換成您想要的編碼方式就可以了。
- 最初跟最後的「-*-」是跟電腦講說編碼開始與結束的意思。
  - 不過，隨著Python 版本的不斷演進，現在編碼連-*- 都可以省去了，直接#coding: 就可以了。

編碼

- 從標準的編碼方式可以延伸出比較簡略的方式，以下分別列出並比對說明：

```
#-*-coding:UTF-8-*-    # 正式寫法，Python 編碼聲明格式
#coding:UTF-8          # 這樣也可以，可捨去-*-
#encoding:UTF-8        # 這樣也可以，coding 亦可使用encoding
#coding= UTF-8         # 這樣也可以，可用= 取代:
# coding: UTF-8        # 這樣也可以，# 後可容許空格。
                       # 注意: 2.7 版coding 與: 或= 間不容許空格。
                       # 3.3 版之後是允許可以有空格的。
```

資料型態

- 在Python 的世界裡面，資料會以各種不同的型別儲存在電腦裡，以供Python 程式進行存取

- 其中，資料型態主要可分成三種類型：數值型別、字串型別、容器型別。
  - 這些資料型態就像是魔法世界裡的元素屬性一般，可以描述物件的不同性質，彼此之間卻又可以被搭配使用來完成特定的任務。

數值型別
字串型別
容器型別

數值型別（NUMERIC TYPE）

- **int, float, bool, complex**
  - **int** ： integer 的縮寫，也就是整數的意思，負責儲存沒有小數點的整數數值。
  - **float**：浮點數，可以儲存具有小數點的數值。
  - **bool**：boolean 的縮寫， 中文通稱為「布林」， 只有三種資料值， 包括True、False 與None，分別代表真、假和空值三種意思。
  - **complex**：代表數學中的虛數，以complex(x, y) 的形式出現。其中，x 為實部的數字，而y 為虛部的數字。除此之外，也可以在數字後面直接加j 來代表虛數部，如：1+5j、6.7-4.2j 等。

資料型態

- **字串型別（String Type） - str**
  在Python 裡面，各位可以用「‘ ’」、「“ ”」、「‘‘‘ ’’’」把文字包住，而這些都代表字串的意思。
- 字串有可以被拆解使用，且具有前後順序的特性，例如在字串變數x=“Hello, World” 裡面，
- x 這個字串裡面有12 個字元。
- 依序是『H』、『e』、『l』、『l』、『o』、『,』、『 』、『W』、『o』、『r』、『l』、及『d』。

字串型別（STRING TYPE）

- 字串型別（String Type） - str
  如果想要取出x裡面第8個字元，就得下x[7] 這個指令。
- 為什麼要取出第8個字元，[ ] 裡面卻是7？
- 因為從0開始算的，所以從0算到7剛好是8 個

內容	H	e	l	l	o	,		W	o	r	l	d
index	0	1	2	3	4	5	6	7	8	9	10	11

資料型態

- 示範程式碼

```
01  x = ‘Hello, World’   # 將 ‘Hello, World' 指定到x
02  print(x[7])
```

- 執行結果

```
>>>
W
>>>
```

小練習

```
01  x = ‘Hello, World’   # 將 ‘Hello, World' 指定到x
```

- 請試著取出上面字串**x**中的「,」號

容器型別（CONTAINER TYPE）

- **tuple, list, set, dict**

  這個型別裡面的資料就像是箱子一樣，可以裝入各種不同型態的資料，也可以在箱子裡裝入其他的箱子喔。

  - **tuple**：序對，在( ) 裡面可以放置一個以上的資料值，有順序性，但是不能更改其內容。
  - **list**：串列，在[ ] 裡面可以放置一個以上的資料值，是一種有順序且可以更改其內容的型態。
  - **set**：集合，在{ } 裡面可以放置一個以上的資料值，類似數學裡面的集合概念，所以內容並無順序性。
  - **dict**：字典，是dictionary 的縮寫，以Key-Value 對應的型態在{ } 裡面放置一個以上的元素。字典是種配對型別，也可以放置一個以上的資料值，而key 就是用來存取每個 value 的索引值。

資料型態

- **容器型別（Container Type） - tuple, list, set, dict**

容器型別	tuple	list	set	dict
中文譯名	序對	串列	集合	字典
使用符號	( )	[ ]	{ }	{ }
具順序性?	有	有	無	無
更改內容?	不可以	可以	可以	可以

資料型態

- 在這裡，特別介紹函數type()
- 這個函數會回傳圓括號裡面的變數型態，可以利用這個函數來確認自己所設定的變數是否符合所想的型態，以免出現型態不符的錯誤。

資料型態

- 示範程式碼

```
01  x = 123                 # 將x 設定為整數int 型態
02  print(type(x))
03
04  x = '123'               # 將x 設定為字串str 型態
05  print(type(x))
06
07  x = 123.0               # 將x 設定為浮點數float 型態
08  print(type(x))
09
```

```
>>>
<class 'int'>
<class 'str'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'tuple'>
<class 'list'>
<class 'set'>
<class 'dict'>
>>>
```

資料型態

- 示範程式碼

```
10  x = 123+0j          # 將x 設定為複數complex 型態
11  print(type(x))
12
13  x = True            # 將x 設定為布林bool 型態
14  print(type(x))
15
16  x = (123, 456)      # 將x 設定為序對tuple 型態
17  print(type(x))
18
```

```
>>>
<class 'int'>
<class 'str'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'tuple'>
<class 'list'>
<class 'set'>
<class 'dict'>
>>>
```

資料型態

- 示範程式碼

```
19  x = [123]          # 將x 設定為清單list 型態
20  print(type(x))
21
22  x = {123}          # 將x 設定為集合set 型態
23  print(type(x))
24
25  x = {1:123}        # 將x 設定為字典dict 型態
26  print(type(x))
```

```
>>>
<class 'int'>
<class 'str'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'tuple'>
<class 'list'>
<class 'set'>
<class 'dict'>
>>>
```

輸出

- 在3.4 版裡面，print 已經變成內建的函數。
  - **value** 就是想要輸出的資料。

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

  - **...** 代表可以用「,」隔開多個想輸出的值
  - **sep** 是將多個印出的值隔開的方式，預設為一個半形空白
  - **end** 是每一次print 輸出之後會在最後面加上的內容，預設為「**\n**」，也就是換行符號
  - **file** 為輸出串流裝置，預設為 **sys.stdout**，也就是標準輸出裝置（通常是螢幕）
  - **flush** 是3.3 版所增加的新功能，意思是在這行print指令執行後是否要強制將欲輸出之資訊輸出，而不是先放在緩衝區，等同一區塊的程式碼都執行完畢後才一次性輸出，通常，這個選項在迴圈執行時比較容易遇到，不過因為現在還沒有教到迴圈語法，因此先不用太在意，在迴圈教學中將會有詳細的介紹

輸出

- 示範程式碼

```
01  print('Hello, World')                  # 預設的print 方式
02  print('Hello, World', end=' ')         # 將預設的結尾換行符號替換成半型空白，
03                                         # 所以下一次輸出便會接在這一次輸出之後
04  print('Hello, ', 'World', sep='@')     # 用@ 替換半型空白當作隔開項目之間的字元
```

```
>>>
Hello, World
Hello, World Hello, @World
>>>
```


輸出- ESCAPE（逸出/脫逸 字元）

- 像「‘」、「’’」等不會被輸出至螢幕上
- 如果需要輸出這些字元時，就需要加上「\」。
  - 「\」後面接一些固定的字元時會有特別的用途所有跳脫序列的字元在輸出時看不見，由反斜線 \ 開始，後接一些特定字元，常見的有：

跳脫序列的字元	說明	跳脫序列的字元	說明
\a	響鈴	\v	垂直定位
\b	空白	\\	印出反斜線
\f	換頁	\t	tab鍵
\n	換行	\'	印出單引號
\r	歸位	\"	印出雙引號

輸出

- 示範程式碼

```
01  print('testing')                    # 一般輸出
02  print('print with reserved words including「 \'」「 \"」「 \\」「 \b」')
                                        # 加上保留字
03  print(' 姓名\t年齡')                 # 利用\t 實現對齊輸出
04  print(' 王小明\t16 歲')              # 即使字數不同也可以對齊
05  print(' 絕對無敵雷神王\t35 歲', end = '\n\n')
                                        # 若超出一個Tab 鍵距離就無法對齊？
06  print(' 姓名\t\t年齡')
07  print(' 王小明\t\t16 歲')
08  print(' 絕對無敵雷神王\t35 歲')
                                        # 可以利用兩組\t 來實現較長的對齊
```

輸出

- 執行結果

```
>>>
testing
print with reserved words including「 '」「 "」「 \」「 」
姓名          年齡
王小明        16 歲
絕對無敵雷神王    35 歲

姓名                    年齡
王小明                  16 歲
絕對無敵雷神王          35 歲
>>>
```

輸出

- **範例1**
  例如要印出下列的格式

```
>>>
張傑帆		Python
http://homepage.ntu.edu.tw/~d02922022/
>>>
```

- **程式碼**

```
01  print("張傑帆\t\tPython\nhttp://homepage.ntu.edu.tw/~d02922022")
                                        # 在視窗上印出文字
```

輸出

- **隨堂練習**
  請使用print 來印出如以下的內容：

```
>>>
姓名        座號        分數
王小明      20          40
鄭小勳      21          100
>>>
```

- 試著提交上批改系統

輸入

- 示範程式碼

```
01  string = input(‘ 現在輸入的是字串: ’)
                                # 以input 函數取得使用者輸入字串
02  print(type(string))         # 利用type() 確定取得的資料型態是str
03
04  integer1 = input(‘ 現在我想取得兩個數字來相除，先取分子: ’)
05  integer2 = input(‘ 再取分母: ’)
06  print(type(integer1), type(integer2))
                                # 利用type() 確定取得的資料型態是str
07
```

輸入

- 示範程式碼

```
08  print(integer1/integer2)    # 這裡如果執行會產生錯誤，
09                              # 因input 取得的是字串，而字串不能相除
10  integer1 = eval(input(‘ 重新取兩個數字: ’))   # 重新取兩個數字，
11                              # 使用eval() 函數來轉變取得的資料型態
12  integer2 = eval(input(‘ 取分母: ’))
13  print(type(integer1), type(integer2))
                                # 用type() 確定取得的資料型態是數值型態
14  print(integer1/integer2)    # 兩個數值型態的資料相除就不會出錯
```

輸入

- 示範程式碼

```
>>> a = "455dd"
>>> a
'455dd'
>>> eval(a)
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    eval(a)
  File "<string>", line 1
    455dd
        ^
SyntaxError: unexpected EOF while parsing
>>>
```

```
15  integer = eval(input(‘ 測試eval 如果輸入不能轉變成數值的資料會如何: ’))
16
                    # 預設會出現error
```

- 執行結果

```
>>>
現在輸入的是字串: test
<class ‘str’>
現在我想取得兩個數字來相除，先取分子: 9
再取分母: 3
<class ‘str’> <class ‘str‘>
```

輸入

- 執行結果

```
重新取兩個數字: 9
取分母: 3
<class 'int'> <class 'int'>
3.0
測試eval 如果輸入不能轉變成數值的資料會如何: test
Traceback (most recent call last):
  File "C:\Users\John\Desktop\Google 雲端硬碟\Python Wizard\examples\Ch2\ex02_05.py", line 19, in <module>
  integer = eval(input(‘ 測試eval 如果輸入不能轉變成數值的資料會如何’)) # 預設會出現error
  File "<string>", line 1, in <module>
NameError: name 'test' is not defined
>>>
```

輸入

- 如果給予eval() 不能轉換為數值的資料，將會出現錯誤
- 例如，輸入「test」，而test 並不能被轉換為數值型態，在前面的程式碼也並沒有將test 當作變數來儲存數值型態的資料，因此出現「NameError: name 'test' is not defined」的錯誤訊息。

輸入

- **範例**
  在視窗上顯示所輸入的文字

```
01  name = input(“ 請輸入姓名：”)      # 取得使用者輸入的資料
02  print(name)                        # 在視窗上印出文字
```

輸入

- 執行結果

```
>>>
請輸入姓名：王大明
王大明
>>>
```

小練習

- 請輸入兩數字或字串
- 並將其交換順序輸出
- **Ex:**
- 輸入:

  123
  456
- 輸出：

  456
  123

註解

- 語法:

```
# 單行註解
""" 多行
註解"""
```

- 需要每行註解的情況，建議註解在每行的上面
- 有許多工程師會使用這種多行註解來加上自己的標記，以註明這支程式是誰撰寫的。

swallows2.py

```
1  # Suzy Student, CSE 142, Fall 2097
2  # This program prints important messages.
3  print("Hello, world!")
4  print()                          # blank line
5  print("Suppose two swallows \"carry\" it together.")
6  print('African or "European" swallows?')
```

註解

- 示範程式碼

```
01  # 單行註解是以「#」作為開頭，從每一行程式碼的第一個「#」
02  # 之後直到這一行結束都會被程式碼視為註解
03
04  ''' 三個單引號除了可以代表字串之外，也可以用在多行註解
05      如果以三個單引號括起來，自由的換行
06      也不會對程式碼的
07      執行造成錯誤'''
08
```

註解

- **示範程式碼**

```
""""""""""""""""""""""""""""""""""""""""""""""""""""
""" 三個雙引號跟三個單引號有一樣的功能      """
"""                                         """
""" 可以用一堆單引號或雙引號                """
""" 製造出撰寫程式的人想簽名、蓋印記        """
"""  或者記錄其他事項的空間                 """
"""                                         """
""""""""""""""""""""""""""""""""""""""""""""""""""""
print(‘這就是註解的使用方式’)
```

註解

- 執行結果

```
>>>
這就是註解的使用方式
>>>
```

REFERENCE 參考

- 官方網頁

http://www.python.org

- Codecademy-Python教學
- http://www.codecademy.com/learn