

# Python\_03

## 程式碼的基礎

# 程式碼

- 語彙單元
- 語句敘述
- 註解

# 程式碼的編碼宣告

- Python 3 預設使用 UTF-8 編碼(UTF-8 Encoding)
- Python 2 預設是 ASCII 編碼
- 可透過編碼宣告，設定程式的編碼方式

`# -*- coding: <encoding name> -*-`

- 常見的編碼宣告

`# -*- coding: utf-8 -*-`

`# -*- coding: big5 -*-`

`# -*- coding: cp950 -*-`

```
1 # -*- coding: utf-8 -*-
2
3 print('Hello')
4
```

沒寫這行宣告就是用預設編碼，程式一樣可以執行！

有時Python程式碼第一行是UNIX的Shebang符號(#!)用來指定執行系統特定路徑的直譯器

```
1 #!/usr/bin/python3
2 # -*- coding: utf-8 -*-
3
4 print('Hello')
5
```

# 語彙單元(Token)

- Token指程式裡有意義的最小單位
- 直譯器透過Token來解析程式的語法
- Token之間用空白(white space)區隔
- Python 的Token有
  - 關鍵字 (keyword)
  - 識別字 (identifier)
  - 字面值 (literal)
  - 運算子(operator)

# 關鍵字 (keyword)

- 關鍵字為Python語法的保留字 (reserved word)

|        |          |         |          |        |
|--------|----------|---------|----------|--------|
| 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  |

```
Python 3.7 (32-bit)
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>>
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
```

# 識別字 (identifier)


- 識別字是使用者定義的名稱，例如：變數名稱、函數名稱、類別名稱、模組名稱...等。
- 識別字命名規則
  - 為英文字母(A-Z,a-z)、數字(0-9)、底線(\_)
  - 不可以數字開頭
  - 區分大小寫(case-sensitive)
  - 不限長度
  - 不可與Python保留字(關鍵字)相同
  - 一個完整的識別字，中間不能有空白

# 識別字命名與底線

- `_foo`  
模組的不公開成員名稱，當模組被import時，不會被載入。
- `__foo`  
Python 物件的私有成員名稱。
- `__foo__`  
Python 內建定義的名稱。

例如：

```
class Animal():  
    def __init__(self, name):  
        self.name = name
```

 Python內定的名稱，物件初始化函數名稱

```
a = Animal("dog")  
print(a.name)
```

# Python的命名方式

- 變數或函數名稱

variable\_name

\_code

CODE

\_1234

a1234

someThing

SomeThing

駝峰式(Camel-Case)



- 類別名稱

Class\_Name

ClassName

- 方法名稱

method\_name

methodName



# 字面值 (literal)

- 字面值是指字面上的意義也稱為字面常數，例如；  
**1200** 就代表整數數值一千二百的意義，也就是寫在  
Python程式碼中的數值。
- 依據資料型態分為：
  - 字串 字面值(string literal)
  - 位元組 字面值(bytes literal) `b'\x7f\x45\x4c\x46\x01\x01'`
  - 整數 字面值(integer literal)
  - 浮點數 字面值(floating-point literal) (例： 123.45 )
  - 複數 字面值(imaginary literal) (例： 3+4j )
  - Unicode字面值： `u'\u0041'`
  - 特殊字面值： `None`，表示這個變數沒有建立

# 字串字面值(string literal)

- 短字串：(用一對單引號或一對雙引號括起來)
- 長字串：(用一對三個單引號或三個雙引號括起來)
- 反斜線(`\`)：用來對特殊含義的字元進行轉義 (Escape Sequence)
- 加號(`+`)：用來串接字串
- 星號(`*`)：用來重複字串

```
print('短字串')
a = "abcdef"
print(a)
```

```
print('長字串 ')
b = """aaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbb
cccccccccccccccccc
dddddddddddddddddd"""
print(b)
```

```
print('測試 \\newline ')
d= "aaaaa\
bbbbbbb"
print(d)
```

```
print('串接與重複')
c = "0000" + 'KK'*2
print(c)
```

# 轉義字元 (Escape Sequence)

| Escape Sequence       | Meaning                               | Notes |
|-----------------------|---------------------------------------|-------|
| <code>\newline</code> | Backslash and newline ignored         |       |
| <code>\\</code>       | Backslash (\)                         |       |
| <code>\'</code>       | Single quote (')                      |       |
| <code>\"</code>       | Double quote (")                      |       |
| <code>\a</code>       | ASCII Bell (BEL)                      |       |
| <code>\b</code>       | ASCII Backspace (BS)                  |       |
| <code>\f</code>       | ASCII Formfeed (FF)                   |       |
| <code>\n</code>       | ASCII Linefeed (LF)                   |       |
| <code>\r</code>       | ASCII Carriage Return (CR)            |       |
| <code>\t</code>       | ASCII Horizontal Tab (TAB)            |       |
| <code>\v</code>       | ASCII Vertical Tab (VT)               |       |
| <code>\ooo</code>     | Character with octal value <i>ooo</i> | (1,3) |
| <code>\xhh</code>     | Character with hex value <i>hh</i>    | (2,3) |

# 位元組字面值(bytes literal)

- python3對文字和二進位制資料做了區分。
  - 文字是Unicode編碼，String型別，用於顯示。
  - 二進位制型別是bytes型別，用於儲存和傳輸，如： bytes literal 。
- string和bytes之間的轉換關係：

string → encode() → bytes → decode() → string

```
a = '你好'.encode('utf-8')  
print(a)
```

b'\xe4\xbd\xa0\xe5\xa5\xbd'

```
b = a.decode('utf-8')  
print(b)
```

你好

# 整數字面值 (integer literal)

- 十進位(decimal) : 1200
- 二進位(binary) : `0b1100100`, `0B1100100`
- 八進位(octal) : `0o123`, `0O123`
- 十六進位(hexadecimal) : `0x64`, `0X64`
- 也可以加底線( `_` )增加可讀性 :

```
100_000_000_000
0b_1110_0101
```

# 程式碼的敘述語句

- 每一行敘述(statement)的結束不用分號(;) )
- 同一行中使用多條語句，語句之間使用分號(;)分割
- **pass**代表此行為空敘述，不執行任何事情的敘述
- 透過空行區隔段落

```
1 a=10
2 b=15
3 print(_a + b_)
4
5 a=8 ; b=9; print(a-b)
6
7
```

# 註解(Comments)

- 單行註解

# 這是一行註解

# 這是一行註解

# 這是一行註解

print ( 'ABC ' ) # 印出ABC

- 多行註解

"""

這是三個雙引號的多行註解

這是三個雙引號的多行註解

"""

'''

這是用三個單引號的多行註解

這是用三個單引號的多行註解

'''

# 多行語句

- Python 通常是一行寫完一條語句，但如果語句很長，我們可以使用反斜線(\)來連接多行語句，例如：

```
if 1900 < year < 2100 and 1 <= month <= 12 \
    and 1 <= day <= 31 and 0 <= hour < 24 \
    and 0 <= minute < 60 and 0 <= second < 60:    # Looks like a valid date
    return 1
```

- 使用括號 ( )，方括號 [ ] 或大括號 { } 的多行語句，則不用反斜線

```
month_names = ['Januari', 'Februari', 'Maart',      # These are the
               'April',   'Mei',      'Juni',      # Dutch names
               'Juli',    'Augustus', 'September', # for the months
               'Oktober', 'November', 'December']  # of the year
```



# 程式碼區塊與縮排

- Python程式碼最特別的就是使用縮排(indentation)來表示程式碼區塊，不需要使用大括號{ }。有同樣縮排的程式碼會被視為同一個程式碼區塊。
- 縮排的空格數是可變的，但是同一個區塊的語句必須包含相同的空格數。
- 常用 Tab按鍵 或 4格空格 作為縮排

```
def perm(l):  
    # Compute the list of all permutations of l  
    if len(l) <= 1:  
        return [l]  
    r = []  
    for i in range(len(l)):  
        s = l[:i] + l[i+1:]  
        p = perm(s)  
        for x in p:  
            r.append(l[i:i+1] + x)  
    return r
```

# 標準輸出

- 語法： `print(*objects, sep=' ', end=' \n', file=sys.stdout)`
- 說明： `print()` 方法用於列印輸出。
- `print` 不換行 `print('Hello World', end='')`

```
print(1)
```

1

```
print("Hello World")
```

Hello World

```
a = 1235
```

1235 abcdef

```
b = 'abcdef'
```

aaabbb

```
print(a,b)
```

aaa bbb

```
print("aaa" "bbb")
```

```
print("aaa", "bbb")
```

[www.abcdef.com](http://www.abcdef.com)

```
print("www", "abcdef", "com", sep=".")
```

# 標準輸入

- 語法： `input([prompt])`
- 說明： `input()` 是Python3.x 內建函數，接受一個標準輸入(`std.in`)數據，回傳為`string` 類型。

```
a = input('請輸入：')  
print('您輸入的是：')  
print(a)
```

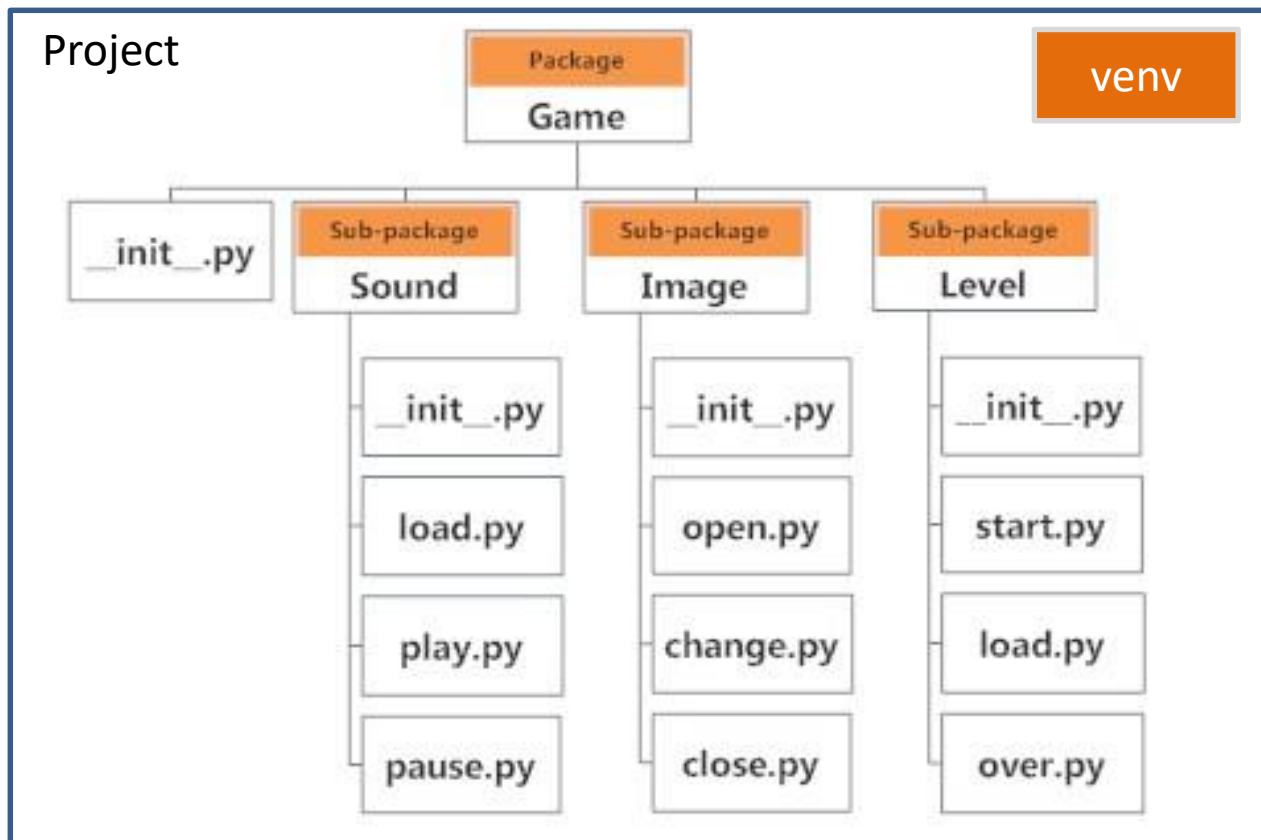
請輸入：*abcd12345*

您輸入的是

abcd12345

# 專案、套件、模組

- Python專案(Project)是透過套件(package)和模組(module)來組織
- 一個大型專案(Project)可以包含多個套件(package)
- 一個套件(資料夾)可以包含多個模組
- 一個.py檔案，可視為一個模組



# 載入模組

- Python用 **import** 或者 **from...import** 來 **載入模組**。
- 將某一個模組(somemodule)整個模組載入，格式為：  
**import somemodule**
- 從某個模組中載入某個函數(somefunction),格式為：  
**from somemodule import somefunction**
- 從某個模組中載入多個函數,格式為：  
**from somemodule import firstfunc, secondfunc, thirdfunc**
- 將某個模組中的全部函數載入，格式為：  
**from somemodule import \***

# import 與 from...import

- `sys` 是Python標準函式庫的模組
- `sys` 模組中主要的函式變數
  - `sys.stdin` 標準輸入
  - `sys.stdout` 標準輸出
  - `sys.stderr` 標準錯誤
  - `sys.path` 查詢模組所在目錄的目錄名列表
  - `sys.argv` 命令列的引數，包括指令碼名稱
  - `sys.platform` 返回當前系統平臺
  - `sys.exit` 返回異常以載入`sys` 模組為例

```
import sys
print('===== Python import =====');
print ('python 路徑為 \n',sys.path)
```

```
print('')
```

```
from sys import argv, path
print('===== Python from import =====')
print('python 路徑為 \n', path)
```

# 單元練習 03

- 變數名稱或檔案名稱，可否使用關鍵字？
- 變數名稱或檔案名稱，可否使用其他模組的識別保留字？
- Python程式碼的換行符號？
- Python程式碼的單行註解符號？
- Python的檔案名稱也代表什麼名稱？