

UNIT 9



python™

class

張傑帆 Chang, Jie-Fan



OUTLINE

■ 類別



類別

- 類別 (class) 為物件 (object) 設計的模板
- Python 裡所有東西都是物件
- 凡是物件都有
 - 屬性 (attribute)
 - 方法 (method)
- 屬性雷同變數 (variable) ，專屬於物件
- 方法類似函數 (function) ，同樣專屬於物件



定義類別

- 定義類別使用關鍵字 (keyword) **class** ，形式如下

```
class Class_Name:  
    #內容
```



類別

```
class Demo:  
    i = 100  
    def hello(self):  
        print("hello")  
  
help(Demo)  
print(dir(Demo))
```

- 內建函數 **help()** 會顯示物件（包括類別）的資訊
- **dir()** 則會把物件（包括類別）的所有屬性與方法以串列 (**list**) 列出



建立物件

```
class Demo:  
    i = 100  
    def hello(self):  
        print("hello")
```

```
d = Demo()  
print(type(Demo))  
print(type(d))  
print(d.i)  
d.hello()
```

- Demo() 就是 Demo 類別建立物件的建構子 (constructor)
- 這裡利用指派運算子將建構子所建立的實體物件 (instance) 給變數 d，於是 d 就具有 Demo 型態的物件。



類別 `__INIT__()`

- 利用建構子 (constructor) 建立的物件被稱為實體 (instance)，實際上建立物件的整個過程是執行 `__init__()` 方法 (method)
- 自行定義的類別會有預先定義好的 `__init__()`，我們也可以改寫 (override) 成我們自己需要的
- 改寫方式就是再定義一次，方法的定義與函數 (function) 類似，兩者同樣使用關鍵字 (keyword) `def`



類別 `__INIT__()`

```
class Demo:
```

```
    def __init__(self):
```

```
        self.name = "Python"
```

```
    def hello(self):
```

```
        print("hello", self.name)
```

```
d = Demo()
```

```
print(type(Demo))
```

```
print(type(d))
```

```
print("d.name: %s"%d.name)
```

```
d.hello()
```

- 凡是實體的方法都需要一個特別的參數 -- **self**，**self** 是個預設的保留字 (reserved word)，所指的是實體本身
- 在 `__init__()` 所定義的實體屬性 (attribute) 都需要額外加 **self**，如第 3 行的 `self.name`。



設定類別 `__INIT__()` 參數

```
class Demo:
    def __init__(self,name):
        self.name = name
    def hello(self):
        print("hello",self.name)
```

```
d = Demo("Tom")
print(type(Demo))
print(type(d))
print("d.name: %s"%d.name)
d.hello()
```

- 帶入自訂的預設參數，加在**self**的後面



類別 `__DOC__`

- 類別 (class) 有 `__doc__` 屬性 (attribute) ，這是三引號字串定義的文字，屬於類別的說明文件

```
class Demo:
```

```
    """
```

```
        Demo Document:
```

```
        hello python
```

```
    """
```

```
    def __init__(self,name):
```

```
        self.name = name
```

```
    def hello(self):
```

```
        print("hello",self.name)
```

```
d = Demo("Tom")
```

```
print(help(d))
```



類別屬性與實體屬性

class Demo: **#x 為類別屬性， self.i 則為實體屬性**

```
    x=0
```

```
    '''
```

```
    Demo Document:
```

```
        hello python
```

```
    '''
```

```
    def __init__(self,i):
```

```
        self.i = i
```

```
        Demo.x += 1
```

```
    def hello(self):
```

```
        print("hello",self.i)
```




```
print("There are", Demo.x, "instances.")
a = Demo(1122)
a.hello()
print("a.x =", a.x)
b = Demo(3344)
b.hello()
print("b.x =", b.x)
c = Demo(5566)
c.hello()
print("c.x =", c.x)
d = Demo(7788)
d.hello()
print("d.x =", d.x)
e = Demo(9900)
e.hello()
print("e.x =", e.x)
print("After all, there are", Demo.x, "instances.")
```



類別屬性與實體屬性

class Demo:

i=0

'''

Demo Document:

hello python

'''

def __init__(self,i):

self.i = i

Demo.i += 1

def hello(self):

print("hello",self.i)

- 若是類別屬性與實體屬性的識別字相同
- 實體物件只能存取實體屬性



```
print("There are", Demo.i, "instances.")
```

```
a = Demo(1122)
```

```
a.hello()
```

```
print("a.i =", a.i)
```

```
b = Demo(3344)
```

```
b.hello()
```

```
print("b.i =", b.i)
```

```
c = Demo(5566)
```

```
c.hello()
```

```
print("c.i =", c.i)
```

```
d = Demo(7788)
```

```
d.hello()
```

```
print("d.i =", d.i)
```

```
e = Demo(9900)
```

```
e.hello()
```

```
print("e.i =", e.i)
```

```
print("After all, there are", Demo.i, "instances.")
```



類別 課堂練習

- 試寫一個名為**student**的類別
- 其中屬性包含:
 - **name, gender, grades**
- 函數包含:
 - **avg**: 回傳**grades list**的平均值
 - **add(grade)**: 新增成績到**grades list**中
 - **fcount**: 回傳不及格(<60)的總數
- 在建立實體實傳入 姓名及性別
 - **Hint**: 使用建構式**__init__**設定姓名，性別



回家作業

- 將右方程式碼加入
- 分別將每個學生的成績
平均、不及格的的數目印出
- 於類別外寫一個**top**的函數:
- 傳入值為學生物件的序列
- 將平均分數最高的學生回傳

student_test.py

```
1 s1 = student("Tom","M")
2 s2 = student("Jane","F")
3 s3 = student("John","M")
4 s4 = student("Ann","F")
5 s5 = student("Peter","M")
6 s1.add(80)
7 s1.add(90)
8 s1.add(55)
9 s1.add(77)
10 s1.add(40)
11 s2.add(58)
12 s2.add(87)
13 s3.add(100)
14 s3.add(80)
15 s4.add(40)
16 s4.add(55)
17 s5.add(60)
18 s5.add(60)
```

