

# Python程式設計

蘇柏原(teaching@bo-yuan.net)

# 軟體安裝

- 下載並安裝Python 3 :
  - <https://www.python.org/downloads/>
- 下載並安裝Sublime :
  - <https://www.sublimetext.com/3>

# 程式語言

# 程式語言

- 程式語言的執行是有順序性的，由上到下，由左到右
- 他有一些特性與數學類似，譬如：
  - 變數： $x$ 、 $y$ 、 $z$
  - 函數(函式)： $f(x)$ 、 $g(x)$
- 但它除了可以處理數字資料以外，還可以處理文字或者是執行電腦操作等等

# 程式語言

- 學習程式語言需要有良好的邏輯概念，譬如：
  - A等於B
  - B等於C
  - 所以A是否等於C？

# 程式語言

- 運算的邏輯：
  - 小王的體重是50公斤
  - 小明的體重比小王的體重多15公斤
  - 所以小明是幾公斤？
- Python程式語言表示法：
  - $x = 50$
  - $y = x + 15$
  - y等於？

# 程式語言

- 自己與自己的運算：
  - 小明1小時前身上有30元
  - 小明現在身上的錢比小明1小時前的錢少10元
  - 所以小明現在有多少錢？
- 在程式語言中每一列就是一個時間點，所以表示方式更為簡單：
  - $x = 30$
  - $x = x - 10$
  - x等於？

# 程式語言

- 字串的连接：
  - 1號是小明
  - 2號是小王
  - 1號與2號的姓名連在一起寫是「小明小王」
- 程式語言的表示方式（在表示字串時，必須將字串用「"」或「'」框著）：
  - `x="小明"`
  - `y="小王"`
  - 「`x+y`」等於「小明小王」

在Python中使用「+」作為文字串連的符號



# 程式語言

- 在過去我們學的數學中有所謂的函數：

- $f(x) = x + 10$

- $f(20) = ?$

使用

定義

- 而在Python程式語言中則有稱為函式的東西：

```
def f(x):
```

```
    return x+10
```

f(20)等於？

函式內容必須  
縮排

要傳回的值必須在前面  
加上return

# Python程式語言

變數

# Python程式 語言

- 在Python中，**變數**的型態有很多種，常見的有：
  - 數字
  - 文字
  - 串列(list)
    - 猶如數學的矩陣，一個變數中可以在存取多種資料，可使用位址索引來讀寫
  - 元組(tuple)
    - 不可修改的串列
  - 字典(dictionary)
    - 位址索引為文字的矩陣
  - 其它
    - 布林和其它物件等等

# Python程式 語言

- 變數在使用前需要先建立，而建立就是定義初始值：

`q=10`

如果要刪除：  
`del q`

- 而變數的型態取決於它的初值設定：

- `a=10` ← 數字

- `b="10"` ← 文字(只要被「"」或「'」框住就會被當文字)

- `c=[]` ← 串列

- `d=()` ← 元組

- `e={}` ← 字典

可以使用函式「**print**」  
來輸出文字或變數內容

# Python程式 語言

- **數字**是一個基本的變數型態，他可以進行的運算有：

運算子	說明	範例	簡寫
+	加	5+10、x+10、x+y	<b>a=a+x</b> 簡寫 <b>a+=x</b>
-	減	20-10、x-10、x-y	<b>a=a-x</b> 簡寫 <b>a-=x</b>
*	乘	3*5、x*5、x*y	<b>a=a*x</b> 簡寫 <b>a*=x</b>
/	除	20/4、x/4、x/y	<b>a=a/x</b> 簡寫 <b>a/=x</b>
%	取餘數	23%7、x%7、x%y	<b>a=a%x</b> 簡寫 <b>a%=x</b>

- 備註：運算元可以是**數字**或**變數**

# Python程式 語言

- 文字型態：
  - 設定時必須用「`"`」或「`'`」框著
  - 運算子只有「`+`」號，可被用來串聯兩個文字
    - `"abc"+"123"="abc123"`
  - 運算時的運算元可以是字串或變數
  - 字串的換行符號為「`\n`」，譬如：`"ABC\nDEF"`

# Python程式 語言

- 文字裡的值如果要搜尋**所在位址**：  
文字.find(要搜尋的值)
- 文字裡的值如果要統計**出現次數**：  
文字.count(要統計的值)
- 文字裡的值如果要**取代成新值**：  
文字.replace(要取代的值,新值)

# Python程式 語言

- 串列型態：
  - 可以在一個變數中存取多個資料(包含數字和文字)

設置範例1：`aa=[1,2,"A",4,"C"]`

使用範例：`aa[0]` `aa[1]` `aa[2]` `aa[3]` `aa[4]`

串列索引從 0 開始計算

這個數字稱為索引

`aa[1:4]`

可以在使用函式「**len**」  
來取得串列長度

只會取出索引1~3的值，不包含4



# Python程式 語言

- 串列可以進行的運算有：

運算子	說明	範例	簡寫
+	加	[5,2]+[10,7]、x+[10,7]、x+y	a=a+x簡寫a+=x
串列相加(串列元素會變多)，運算元可以是其他串列變數或串列值			

運算子	說明	範例	簡寫
*	乘	[3,2]*5、x*5、x*y	a=a*x簡寫a*=x
串列複製相加，運算元可以是其他數字變數或數值			

# Python程式 語言

- 串列裡的值如果要搜尋所在索引：  
串列.index(要搜尋的值)
- 串列裡的值如果要統計出現次數：  
串列.count(要統計的值)

# Python程式 語言

- 定義字典變數：

```
x={  
    "a":10,  
    "b":"abc",  
    "c":[10,20,30],  
    "e":{  
        "e1":"abc",  
        "e2":30  
    }  
}
```

變數名稱  $\swarrow$   $x$

$\leftarrow$  數字，使用： $x["a"]$

$\leftarrow$  文字，使用： $x["b"]$

$\leftarrow$  串列，  
使用： $x["c"][0]$ 、 $x["c"][1]$ 、 $x["c"][2]$

$\leftarrow$  子字典變數，  
使用： $x["e"]["e1"]$ 、 $x["e"]["e2"]$

也可以之後透過「 $x["索引"]=值$ 」來新增

# Python程式 語言

- 將字典的索引存入串列中：  
`list(字典.keys())`
- 將字典的值存入串列中：  
`list(字典.values())`

# Python程式語言

函式

# Python程式 語言

- 建立一個函式：  

```
def test(a, b):  
    return a+b
```

a與b是呼叫函式時  
可以傳進來的參數
- 建立一個函式，回傳兩個以上的變數：  

```
def test(a, b):  
    return a, b
```

用逗點隔開代表回傳兩個值
- 呼叫函式：  

```
test("X", "Y")
```

因應return的不同會回傳  
不同資料

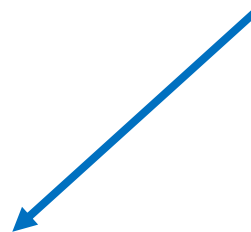
對應建立函式  
時的a與b參數

# Python程式 語言

- 建立一個函式：

```
def test(a="A", b="B"):  
    return a+b
```

參數可以指定初始值



- 呼叫函式：

```
test(b="Y")
```

有指定初始值的參數可以略過輸入，這個指令會回傳字串「AY」



# Python程式語言

判斷式



# 程式語言

- 程式語言擁有邏輯判斷，依據判斷的結果不同，可以有不同的操作
- 範例：

```
if x > 0:
    print("A")
elif x < 0:
    print("B")
else:
    print("C")
```

如果x大於0

否則如果x<0

否則

# Python程式 語言

- 在Python中可以設置**判斷式**，來決定要執行那些程式碼，而判斷式可設置的**條件**如下：

運算子	說明	範例
==	比對是否 <b>相等</b>	A==B
!=	比對是否 <b>不相等</b>	A!=B
>	比對是否 <b>左大於右</b>	A>B
<	比對是否 <b>左小於右</b>	A<B
>=	比對是否 <b>左大於等於右</b>	A>=B
<=	比對是否 <b>左小於等於右</b>	A<=B

- 串列**或**字典**的判斷：

運算子	說明	範例
in	比對是否 <b>存在</b>	A in B
not in	比對是否 <b>不存在</b>	A not in B

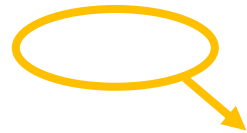
# Python程式語言

- 設置條件式(判斷式)：

```
a=10
```

```
b=20
```

```
if a == b:
```



程式碼寫在這

- 範例：

<pre>if a == b:     z = 1</pre>	如果a等於b則z設為1
<pre>elif a &lt; b:     z = 2</pre>	如果a小於b則z設為2
<pre>else:     z = 3</pre>	否則z設為3

```
if a == b:
```



a等於b時執行這段程式碼

```
elif a < b:
```



a小於b時執行這段程式碼

```
elif...
```

```
elif...
```

可以一直加不同的條件

```
else:
```



如果上面的條件都不成立則執行這段程式碼

# Python程式語言

迴圈

# Python程式 語言

- 當有一段程式碼要重複執行時，可以使用迴圈功能：
  - 譬如，我們想知道 $1+2+3+4+5+6+7+8+9+10...+1000$ 等於多少，若以人力的做法，我們必須在程式碼裡寫下所有的數字，來計算它們的結果，但這樣的做法是相當費時費力的，而迴圈可以解決這樣的問題

# Python程式 語言

- 使用For迴圈來計算：

`x=0`  
`for i in range( 0, 1001, 1 ):`  
`x = x + i`

迴圈開始時的數值  
迴圈結束的數值(小於該數)  
變數每次加的數值  
要重複執行的程式碼

- 上面的迴圈執行條件是i小於等於1000時執行，而i初始為1，每執行完一次迴圈都會將i加1，所以當i被加到超過1000時就會結束迴圈

# Python程式 語言

- 使用While迴圈來計算：

```
x = 0  
i = 1  
while i <= 1000:  
    x = x + i  
    i = i + 1
```

迴圈執行的條件

要重複執行的程式碼

- while只需設置迴圈的執行條件即可運行

# Python程式 語言

- 當迴圈仍然滿足執行條件但卻想提前中斷時：  
`break`
- 當迴圈目前輪迴尚未執行完，而要提前執行下個輪迴時：  
`continue`



# Python程式語言

例外狀況處理

# Python程式 語言

- 例外狀況處理，當程式碼A出現錯誤時不顯示錯誤訊息，而執行程式碼B：

**try:**

程式碼A...

**except:**

程式碼B...

**finally:**

程式碼C...

# Python程式 語言

- 作業：
  - 請用Python語言寫一個可以顯示出九九乘法表的簡易程式。
  - 請將程式碼與結果畫面繳交上來。

# Python程式語言

類別

# Python程式 語言

- 在Python中可以對變數與函式進行分類，這樣的功能可以讓程式更具模組化的特性
- 分類功能，可以讓同一個名稱的變數與函式同時存在不同類別中，而不會互相干擾

# Python程式 語言

- 建立一個類別：

`class` 類別名稱:

`self`代表當前所在的  
類別本身


變數 = 變數內容

`def __init__(self, 參數...):`  
`self.變數 = 變數內容`

一開始呼叫類別時會被  
執行的函式

`def 函式名稱(self, 參數...):`  
`print(self.變數)`

# Python程式 語言

- 呼叫類別：  
操作變數=類別名稱()  
 會呼叫類別中的「\_\_init\_\_」  
函式
- 呼叫類別內的變數：  
操作變數.變數
- 呼叫類別內的函式：  
操作變數.函式名稱(參數...)

# Python程式 語言

- 建立一個類別：

```
class test:  
    aa="1"  
    def __init__(self, a, b):  
        self.aa=a+b  
    def ff(self):  
        print(self.aa)
```

- 呼叫類別：

```
obj=test("A","B")  
obj.ff()
```

會print出「AB」





# Python程式 語言

- 如果不想建立類別操作變數就直接呼叫類別內的函式：

```
class 類別名稱:
```

```
    @staticmethod
```

```
    def 函式名稱(參數...):
```

```
        print("AB")
```

在函式宣告的上面加上  
這段可以讓該函式直接  
被呼叫

- 呼叫方式：  
類別名稱.函式名稱(參數...)

# Python程式 語言

- 建立一個類別：

```
class test:  
    @staticmethod  
    def ff(a, b):  
        print(a+b)
```

- 呼叫類別：

```
test.ff("A","B")
```

← 會print出「AB」

# Python程式 語言

- 建立類別函式：

```
class 類別名稱:
```

```
    @classmethod
```

```
    def 函式名稱(cls, 參數...):
```

```
        print("AB")
```

加上這個，該函式不論  
怎麼被呼叫，第一個參  
數都會指向類別本身

- 呼叫方式：

1. 類別名稱.函式名稱(參數...)

2. 操作變數=類別名稱()  
 操作變數.函式名稱(參數...)

# Python程式 語言

- 建立一個類別：

```
class test:  
    b="A"  
    @classmethod  
    def ff(cls, a):  
        print(cls.b+a)
```

- 呼叫類別：

1. `test.ff("X")` ← 會print出「AX」

2. `obj=test()  
obj.ff("Y")` ← 會print出「AY」

# Python程式 語言

類別中的函式參數說明	透過建立的類別變數呼叫	直接呼叫
預設	第一個參數指向建立的類別變數	所有參數都是一般變數
staticmethod	所有參數都是一般變數	所有參數都是一般變數
classmethod	第一個參數是指向類別本身	第一個參數是指向類別本身
類別變數：類別存取成的變數 類別本身：直接呼叫類別		

# Python程式語言

常用函式庫

# Python程式語言

常用的函式

- 輸入文字：  
`input(提示文字)`

# Python程式語言

常用的函式

- 字串切割成串列：  
要切割的字串.split(作為切割依據的字串)
- 串列結合成字串：  
連接字串.join(要結合的串列)

串列中必須全是字串型態的資料





# Python程式語言

常用的函式

- Python除了一些基本函式以外，絕大多數的函式都需要透過載入模組進來才可以使用：

- `import` 模組名稱

- 使用模組中的函式或變數，範例(取得程式參數)：

`sys.argv`

↑  
模組名稱，需要載入

- 其他載入方式：

- `import` 模組名稱 `as` 新名子

← 使用模組底下的函式時，用新名子表示

- `from` 模組名稱 `import` 該模組底下的函式或變數等

← 可以讓使用時不必在加註模組名稱

# Python程式語言

常用的函式

- 執行指令，並將結果直接輸出至螢幕：  
`os.system(指令)`

- 執行指令，可將結果存入變數中：  
`程序變數=os.popen(指令)`

`程序變數.read()`

# Python程式語言

常用的函式

- 作業：
  - 用Python寫一個能夠自動呼叫預設瀏覽器開啟指定網址的程式。
  - 繳交程式碼檔案。

# Python程式語言

常用函式庫

# Python程式語言

常用函式庫

- 開啟檔案：  
檔案變數=codecs.open(檔名, 開啟類型, 編碼)
- 讀取檔案：  
檔案變數.read()
- 寫入檔案：  
檔案變數.write(寫入內容)
- 關閉檔案：  
檔案變數.close()

常用的類型有：

- w 覆寫
- a 附加
- r 讀取
- b 二進位



# Python程式語言

常用函式庫

- 讓資料在限定的範圍內使用：  
    with 指令 as 變數:  
        程式碼...
- 範例：  
    with codecs.open() as file:  
        print(file.read())

# Python程式語言

常用函式庫

- 刪除檔案：  
`os.remove(檔案名稱)`
- 建立資料夾：  
`os.mkdir(資料夾名稱和路徑)`
- 刪除資料夾：  
`os.rmdir(資料夾名稱和路徑)`
- 取得資料夾內的檔案列表：  
`os.listdir(資料夾名稱和路徑)`

# Python程式語言

常用函式庫

- 取得當前工作路徑：  
`os.getcwd()`
- 改變工作路徑：  
`os.chdir(新路徑)`



# Python程式語言

常用函式庫

- 判斷是否是目錄：  
`os.path.isdir(路徑與名稱)`
- 判斷是否是檔案：  
`os.path.isfile(路徑與名稱)`
- 判斷檔案是否存在：  
`os.path.exists(檔案名稱與路徑)`

# Python程式語言

常用函式庫

- 傳回路徑字串中的檔名部分：  
`os.path.basename(路徑字串)`
- 傳回路徑字串中的路徑部分：  
`os.path.dirname(路徑字串)`
- 取得檔案大小：  
`os.path.getsize(檔案名稱與路徑)`

# Python程式設計示範

檔案管理系統

# Python程式設計 示範

檔案管理系統

- 作業：
  - 完善檔案管理系統的功能。
  - 繳交程式碼檔案。

# Python程式語言

常用函式庫

# Python程式語言

常用函式庫

- 程式停頓：  
`time.sleep(秒數)`
- 取得當前時間的總毫秒數：  
`time.time()`
- 取得當前時間並格式化：  
`time.strftime(格式化方式)`

# Python程式語言

常用函式庫

格式代號	格式說明
%a	星期幾的簡寫
%A	星期幾的全稱
%b	月分的簡寫
%B	月份的全稱
%c	標準的日期的時間串
%d	每月的第幾天
%H	24小時制的小時
%I	12小時制的小時
%j	每年的第幾天
%m	兩位數的月份
%M	兩位數的分鐘數

# Python程式語言

常用函式庫

格式代號	格式說明
%p	本地的AM或PM的等價顯示
%S	兩位數的秒數
%U	第年的第幾周，把星期日做為第一天（值從0到53）
%w	表示星期幾（值從0到6，星期天為0）
%W	每年的第幾周，把星期一做為第一天（值從0到53）
%x	標準的日期串
%X	標準的時間串
%y	兩位數的西元年份（值從0到99）
%Y	四位數的西元年份
%Z	時區名稱，如果不能得到時區名稱則返回空字元
%%	百分號



# Python程式語言

函式庫安裝

# Python程式語言

函式庫安裝

- PIP是Python中專門用於安裝函式庫的程式：  
`pip install 函式庫名稱`
- 如果要移除函式庫：  
`pip uninstall 函式庫名稱`
- 列出當前已安裝的函式庫：  
`pip list`

# Python程式語言

PrettyTable函式庫

# Python程式語言

PrettyTable函式庫

- PrettyTable是一個Python函式庫，可以讓print指令輸出表格，他並不包含在python的預設函式庫中，需要使用PIP來安裝：

```
pip install prettytable
```

- 之後須將其載入：

```
import prettytable
```

# Python程式語言

PrettyTable函式庫

- 建立一個PrettyTable表格：

操作變數=prettytable.PrettyTable(表格標題, encoding=編碼)

list格式，list中的每個值代表一個欄位

注意大小寫

- 新增表格的一個列：

操作變數.add\_row(這個列的值)

list格式，list中的每個值代表一個欄位

# Python程式語言

PrettyTable函式庫

- 設定欄位的對齊方式：

操作變數.align[欄位名稱]="對齊方式"

↑  
「l」、「c」、「r」分別  
代表靠左、置中、靠右

- 把表格顯示出來：

print(操作變數)

# Python程式語言

Colorama函式庫

# Python程式語言

Colorama函式庫

- Colorama是一個可以更改python輸出文字樣式的函式庫，他並不包含在python的預設函式庫中，需要使用PIP來安裝：  
`pip install colorama`



# Python程式語言

Colorama函式庫

- 初始化Colorama函式庫設定：

`colorama.init(自動清除)`

可以設定True或False(預設)

- 樣式設定方式：

`print(樣式+文字+樣式...)`

詳見下幾頁介紹

# Python程式語言

Colorama函式庫

- 設定亮度樣式：  
`colorama.Style.樣式名稱`
- 樣式：
  - NORMAL
  - BRIGHT
  - RESET\_ALL => 清空樣式

# Python程式語言

Colorama函式庫

- 設定文字顏色：  
`colorama.Fore.顏色名稱`
- 設定背景顏色：  
`colorama.Back.顏色名稱`

- 可設定的顏色：
  - BLACK
  - RED
  - GREEN
  - YELLOW
  - BLUE
  - MAGENTA
  - CYAN
  - WHITE
- 原始顏色：
  - RESET

# Python程式語言

Colorama函式庫

- 作業：
  - 使用Colorama函式庫搭配時間函式，寫一個紅綠燈顯示程式。
  - 透過Colorama函式庫呈現紅黃綠三個色塊，並使用時間函式控制他們的停頓秒數如下：
    - 紅燈：5秒
    - 綠燈：4秒
    - 黃燈：1秒
  - 需在紅黃綠色塊下附上計時的秒數。
  - 繳交程式碼檔案。