



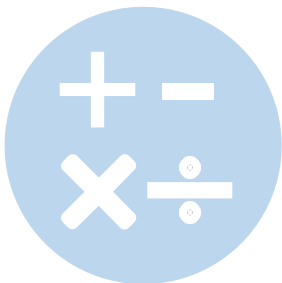
Evolve

Python Tutorial

Python Basics



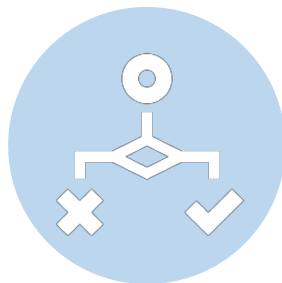
Hello Python



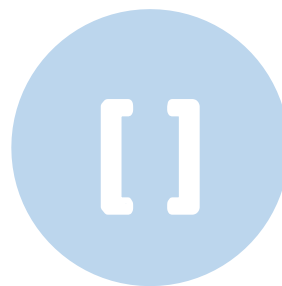
數值運算



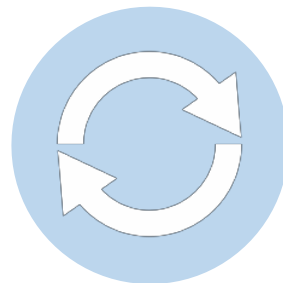
邏輯運算



if - else



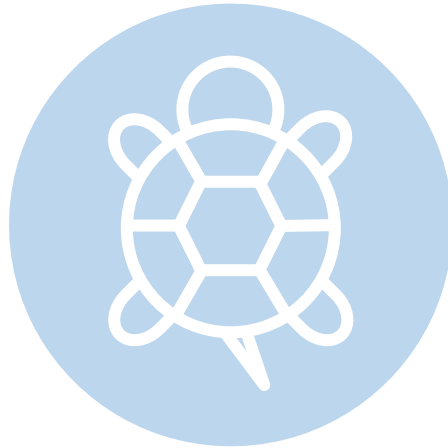
List



Loop



Function



Turtle

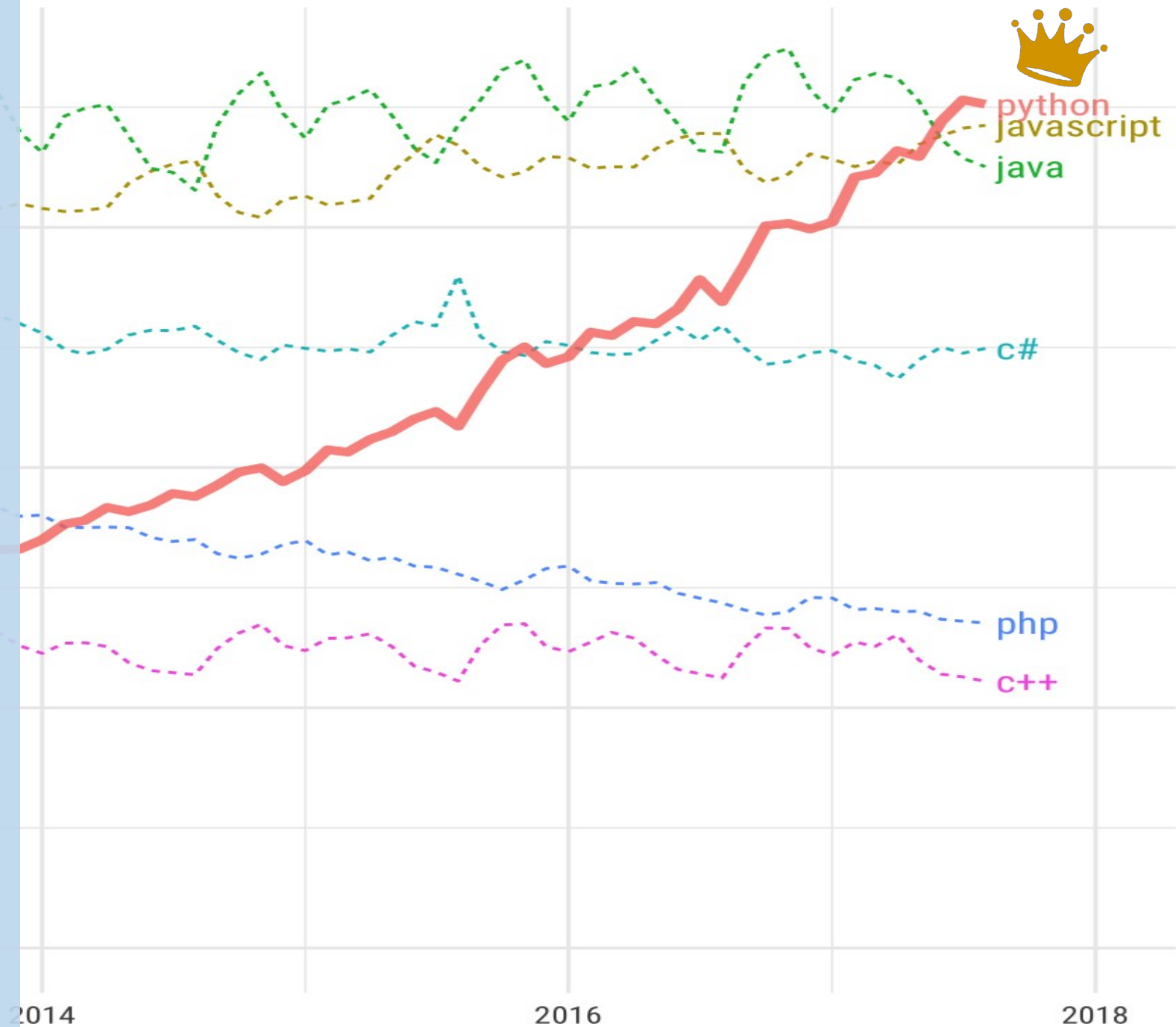
「Why Python ?」

Simple, intuitive
syntax

Amazing libraries

Machine Learning /
Data Visualization /
Data Analysis

Easy !



αβγ

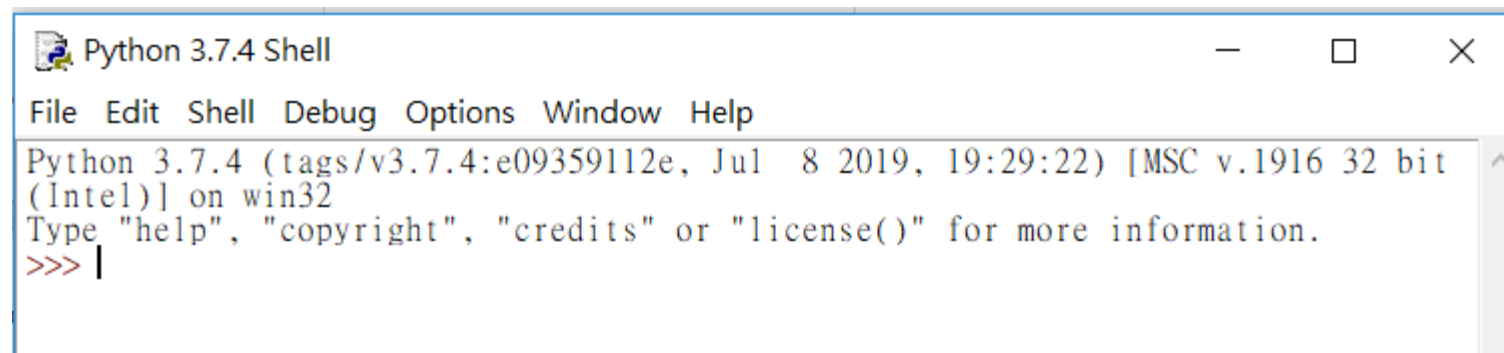


Hello Python

Hello Python

環境設置 - idle

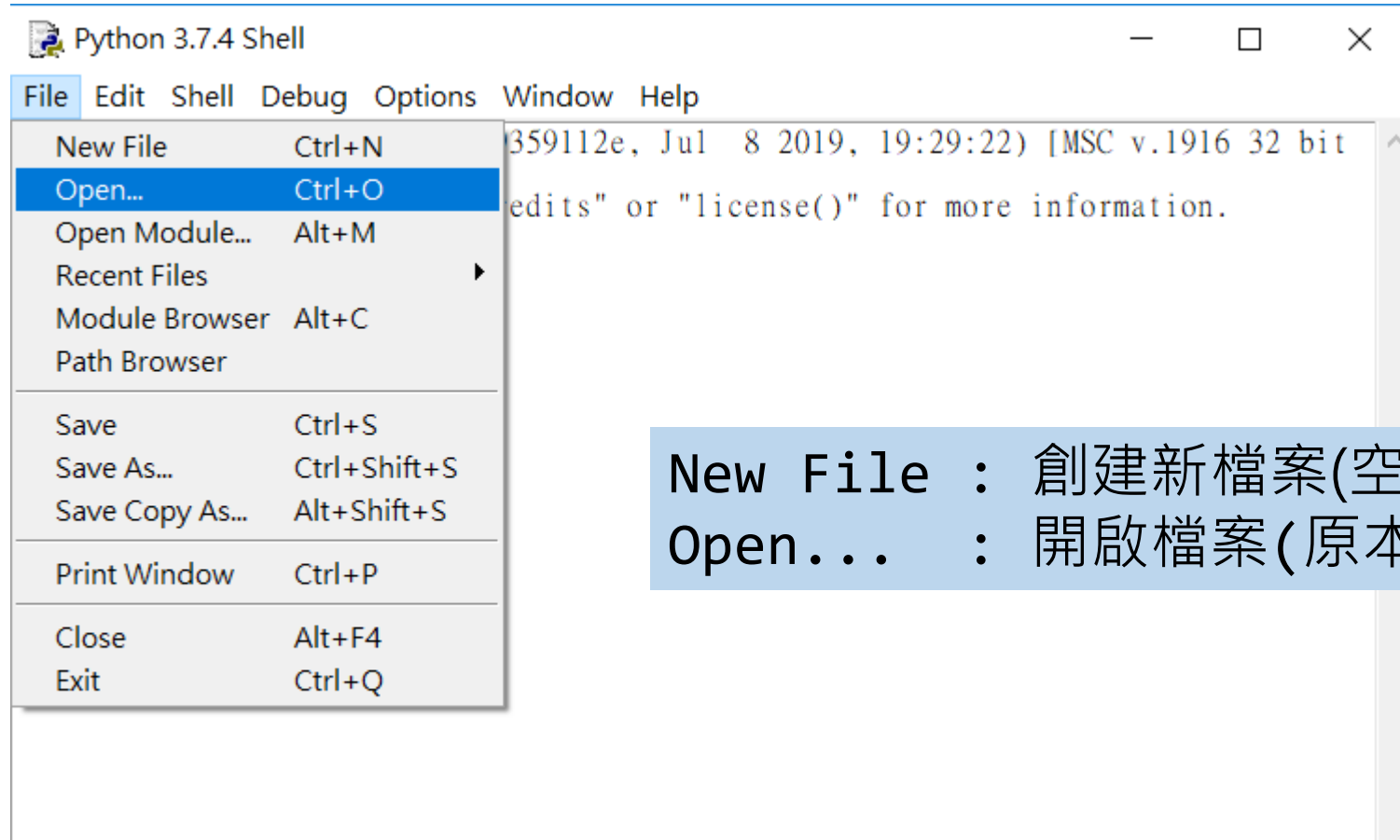
一點開會長這樣



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Hello Python

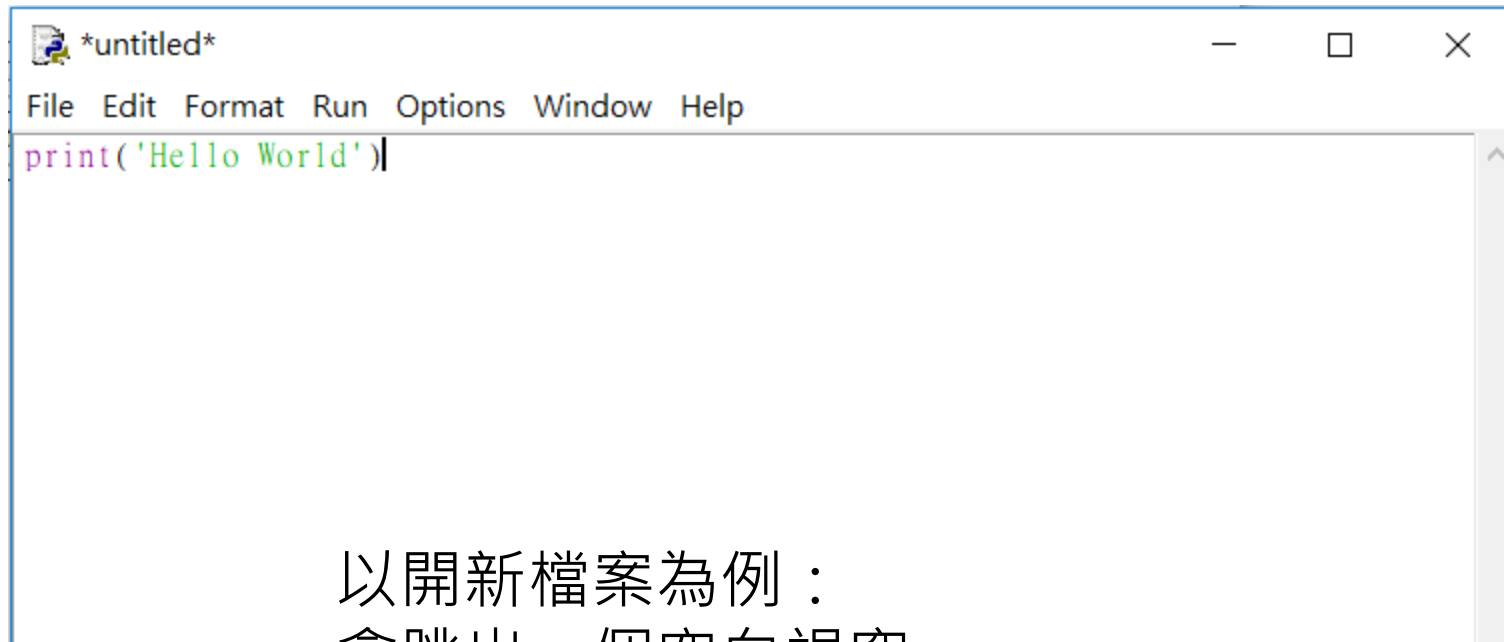
環境設置 - idle



New File : 創建新檔案(空的)
Open... : 開啟檔案(原本已存在)

Hello Python

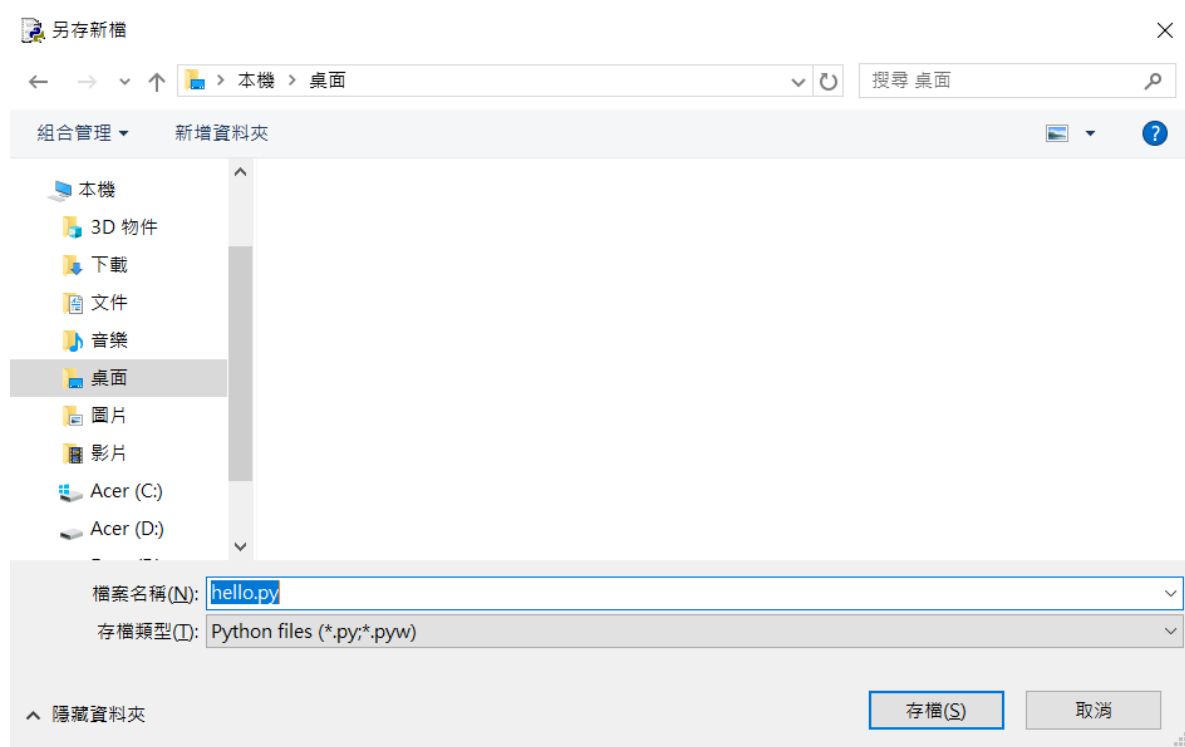
環境設置 - idle



以開新檔案為例：
會跳出一個空白視窗
裡面即是你打程式的地方

Hello Python

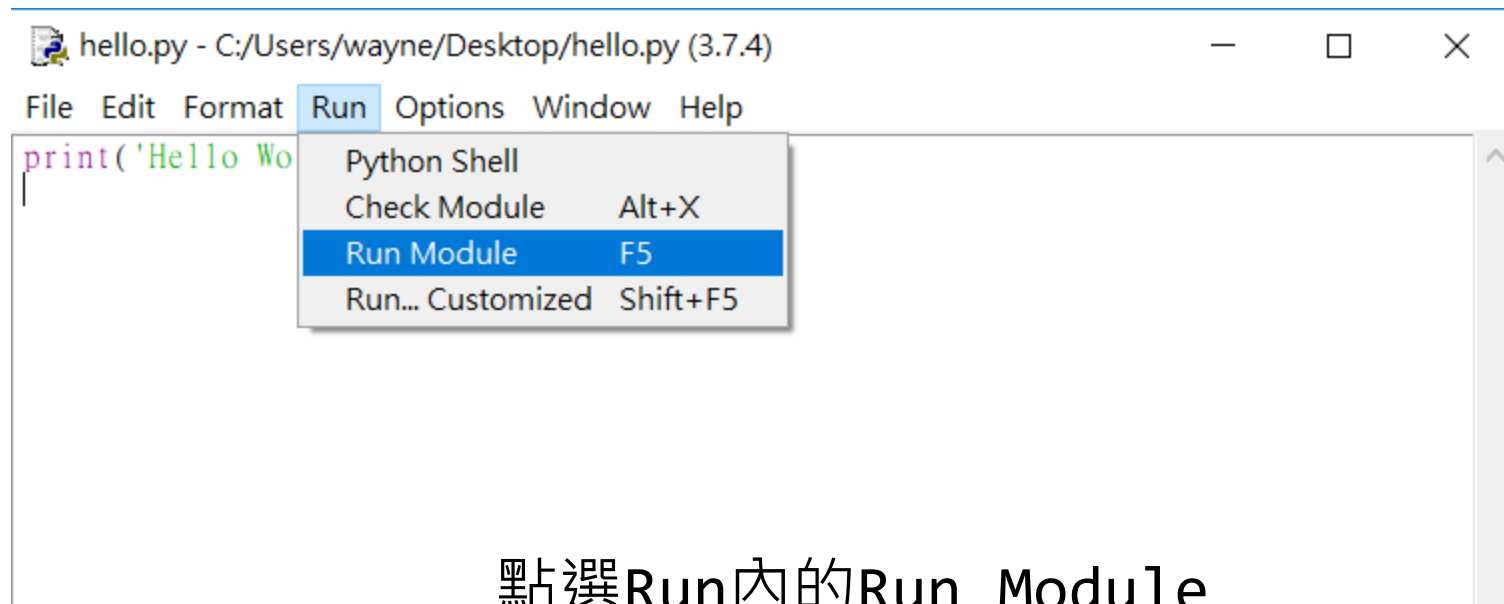
環境設置 - idle



打完程式後記得存檔

Hello Python

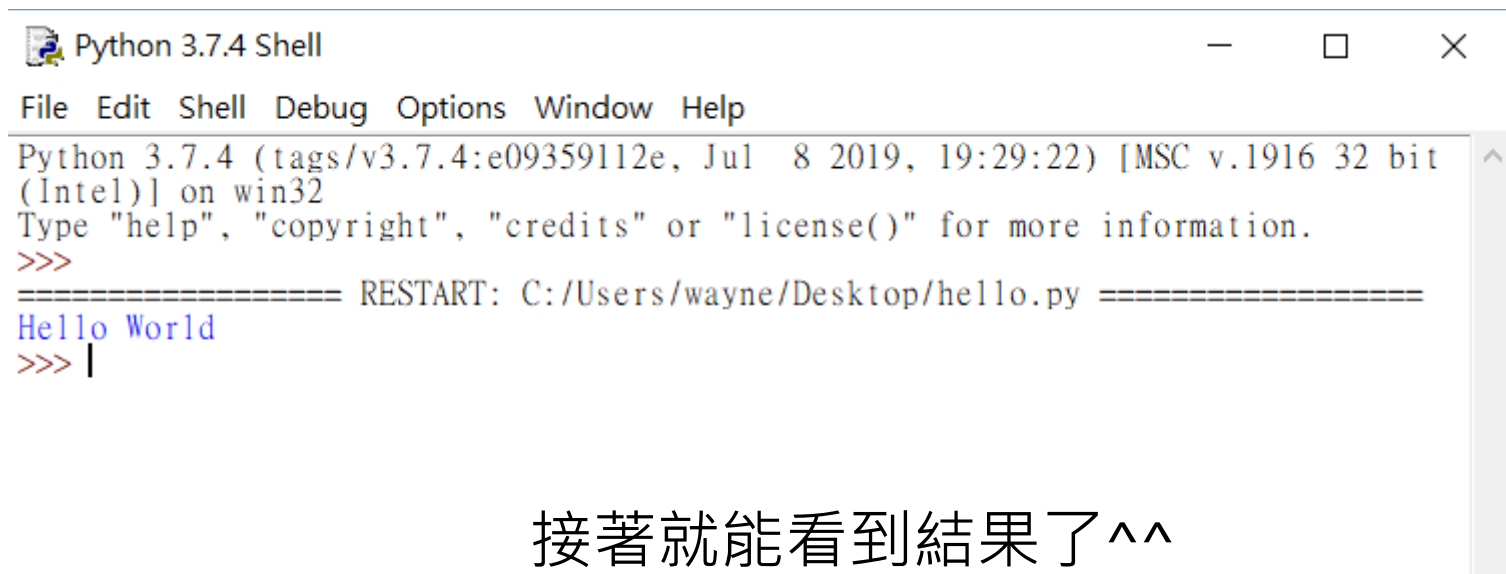
環境設置 - idle



點選Run內的Run Module
或按F5
即可跑出結果

Hello Python

環境設置 - idle



The screenshot shows a window titled "Python 3.7.4 Shell" with a standard menu bar (File, Edit, Shell, Debug, Options, Window, Help). The main text area displays the following content:

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit  
(Intel)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/wayne/Desktop/hello.py =====  
Hello World  
>>> |
```

接著就能看到結果了^^

Hello Python

輸出 print

```
1 print(87)
2 # output: 87
3 print("Hello World")
4 # output: Hello World
```

Hello Python

賦值 assignment

```
1 x = 3
2 y = 2.5
3 print(x, y)
4 # output: 3 2.5
5 x = y
6 print(x, y)
7 # output: 2.5 2.5
```

Hello Python

Practice



將兩個變數 x, y 的值交換

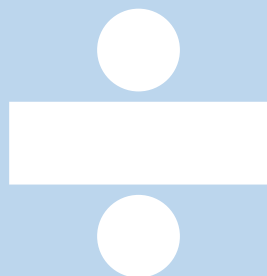
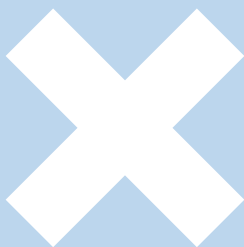
例如：

原本 $x = 3, y = 6$

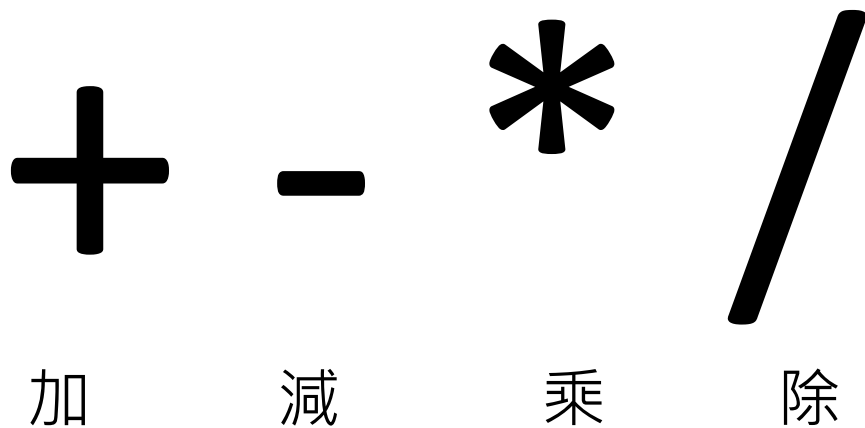
目標 $x = 6, y = 3$

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。以下是常用的資料型態：

```
1 a = 1
2 # a是int (整數 integer)
3 b = 2.5
4 # b是float (浮點數，就是小數)
5 c = True
6 # c是bool (True/False)
7 d = "Hello"
8 # d是str (字串 string)
9 e = [1,2,3]
10 # e是list (長大就學的到)
```



數值運算



跟你小學學的一樣，就是這麼簡單

```
a = 1 + 2
```

```
# a=3
```

```
b = 15 - 7
```

```
# b=8
```

```
c = 3 * 5
```

```
# c=15
```

```
d = 18 / 9
```

```
# d=2.0
```

有比小學還難了嗎

int int
↑ ↑
d = 18 / 9

d=2.0

↓
float????

e = 19 / 8

e=2.375

```
e = 19 / 8
```

```
# e=2.375
```

小學老師有云：
19除以8等於2，餘數是3

小學老師有云：
19除以8等於2，餘數是3

`f = 19 // 8`

`# f=2`

//

`g = 19 % 8`

`# g=3`

%

小學老師沒有云：
2的10次方是1024

```
h = 2**10
```

```
# h=1024
```

$+$ $-$ $*$ $/$ $//$ $\%$ $**$

數值運算

結合四則運算與賦值

四則運算

賦值

$+$ $-$ $*$ $/$ $//$ $\%$ $**$

$=$

$+=$

$-=$

$*=$

$/=$

$//=$

$\%=$

$**=$

數值運算

結合四則運算與賦值

```
1 x = 3
2 x += 2
3     # 等價於 x = x + 2
4 print(x)
5     # output: 5
```

Math module

```
1 import math
2 a = math.sqrt(25)
3 print(a)
4     # output: 5.0
5 b = math.log10(100)
6 print(b)
7     # output: 2.0
```



邏輯運算

True or False

```
1 print(3 > 5)
2     # output: False
3 print(2 < 4)
4     # output: True
```

邏輯運算

比較運算子

>

大於

<

小於

>=

大於等於

<=

小於等於

```
1 print(2 >= 2)
2     # output: True
3 print(2 > 2)
4     # output: False
```

邏輯運算

比較運算子

==

等於

!=

不等於

```
1 print(2 == 2)
2     # output: True
3 print(2 != 2)
4     # output: False
```

注意：= (賦值)和 == (比較左右兩邊變數的值)不同，要特別小心！

and

且

X	Y	X and Y
False	False	False
False	True	False
True	False	False
True	True	True

not

否

X	not X
True	False
False	True

or

或

X	Y	X or Y
False	False	False
False	True	True
True	False	True
True	True	True

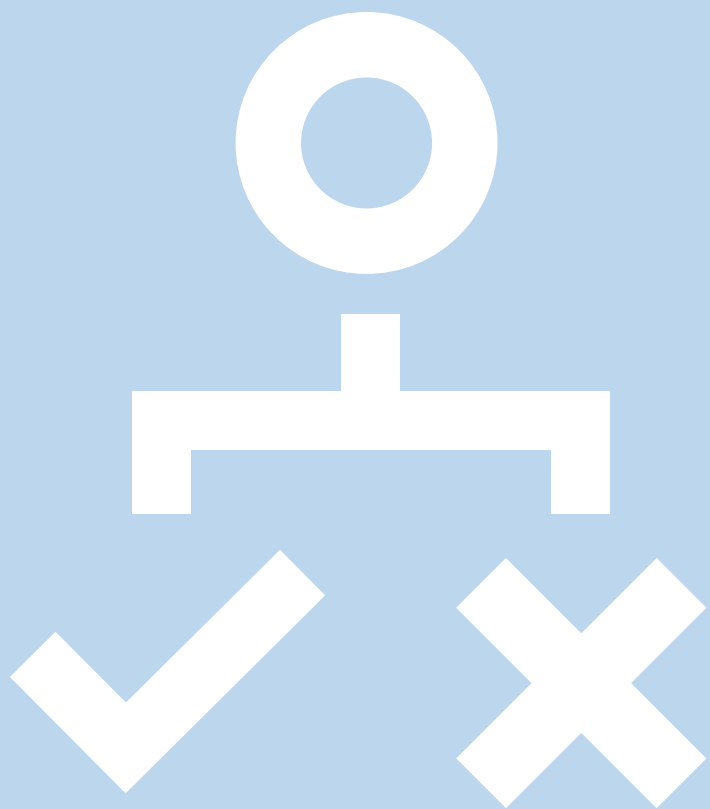
邏輯運算

布林運算子

```
1 print((5 > 3) and (7 > 8))
2     # output: False
3 print((5 > 3) or (7 > 8))
4     # output: True
5 print(not (5 > 3) )
6     # output: False
```


$>$ $<$ $>=$ $<=$
 $=$ \neq

and or not



if - else

if - else

就是 if - else

```
1 if 條件:  
2     條件成立時要做的事  
3 else:  
4     條件不成立時要做的事
```

Python利用縮排表示語句塊的範圍

```
1 people = "killed"  
2 if people == "killed":  
3     print("die")  
4     print("QQ")
```

冒號也記得要打

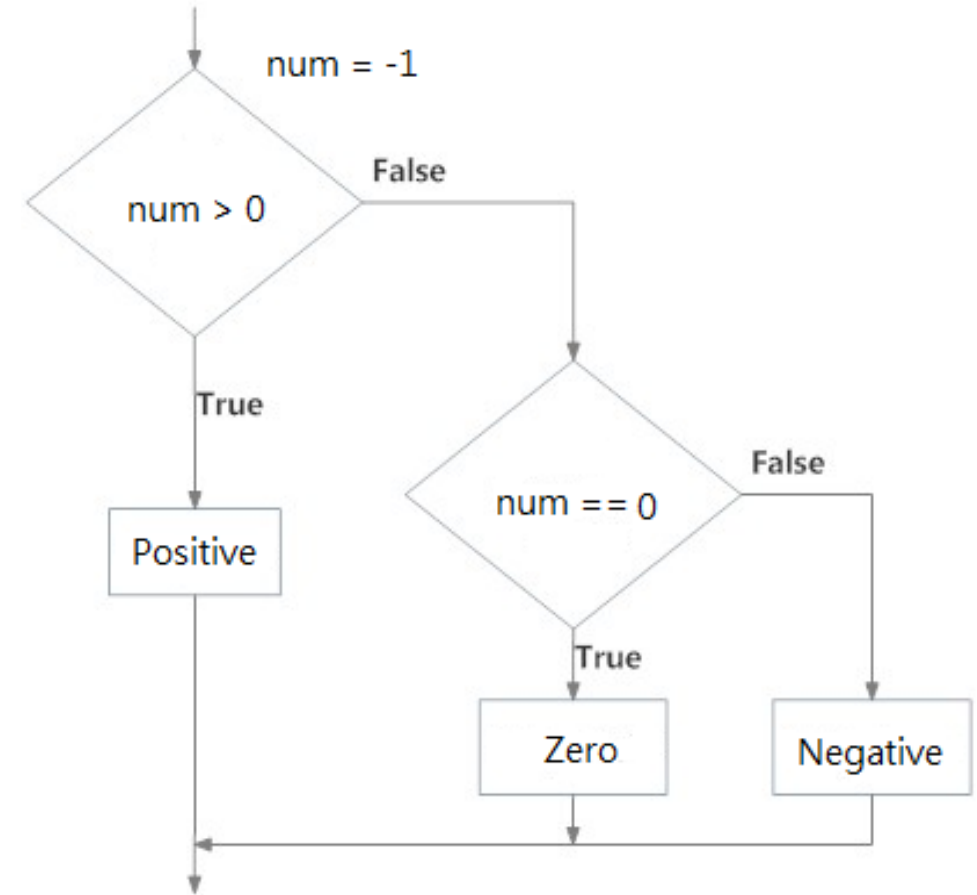


空4格 (或是按tab)

if - else

多層 if - else

```
1 # Ex1
2 num = -1
3 if num > 0:
4     print("Positive")
5 else:
6     if num == 0:
7         print("Zero")
8     else:
9         print("Negative")
```



if - else

更多層 if - else

```
1 # Ex2
2 x = 0
3 if x < 3:
4     print("x is less than 3")
5     if x < 2:
6         print("x is less than 2")
7         if x < 1:
8             print("x is less than 1")
```

if - else

更多層 if - else

注意可讀性

```
function register()
{
    if (empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```



由於地球公轉一周大約是365.242199日，所以如果以一年365日來記算，每四年就會多出0.968796天，於是有人提出每四年要多出一天，這就是閏年。但是因為每四年多出來的並非完整的一天，所以累積到100年的時候，就不須要再多這一天了。而同樣的道理，到了400年，又會多出一天來。目前閏年的規則如下(西元紀年)：

以上廢話，總之遵循以下的規則

1. 如果這一年不是 4 的倍數，則它是平年
2. 如果這一年是 4 的倍數，但不是 100 的倍數，則它是閏年
3. 如果這一年是 100 的倍數，但不是 400 的倍數，則它是平年
4. 如果這一年是 400 的倍數，則它是閏年

現在給你一個年份，請你判斷它是不是閏年

[]

List

如果今天要存下全電機營所有小隊員的身高

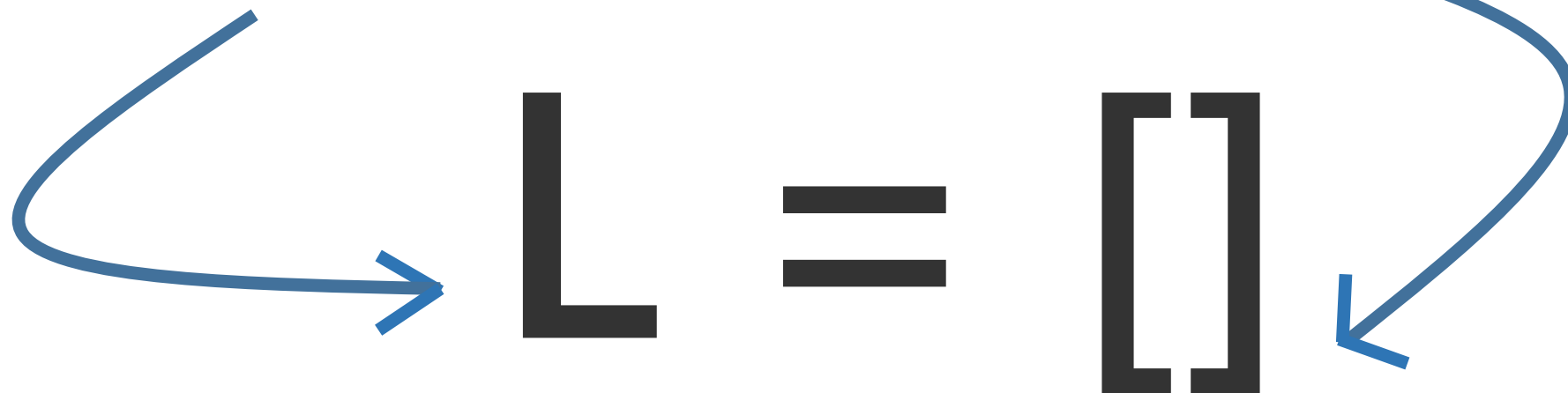
```
1 member_1 = 180
2 member_2 = 172
3 member_3 = 166
4 member_4 = 175
5 member_5 = 156
6 member_6 = 189
7 member_7 = 168
8 member_8 = 167
9 member_9 = 164
10 member_10 = 30
11 member_11 = 189
12 member_12 = 1450
13 member_13 = 152
... # 中間略
120 member_120 = 170
```

整整120個變數

List 讓你可以只用一個變數名稱，就儲存一群資料

變數名稱

中括號

A diagram illustrating the syntax for creating a list in Python. It shows the variable name 'L' followed by an equals sign and a pair of square brackets '[]'. A blue arrow originates from the text '變數名稱' (Variable Name) and points to the 'L'. Another blue arrow originates from the text '中括號' (Square Brackets) and points to the '[]'.

L = []

如果今天要存下全電機營所有小隊員的身高

```
1 member_1 = 180
2 member_2 = 172
3 member_3 = 166
4 member_4 = 175
5 member_5 = 156
6 member_6 = 189
7 member_7 = 168
8 member_8 = 167
9 member_9 = 164
10 member_10 = 30
11 member_11 = 189
12 member_12 = 1450
13 member_13 = 152
... # 中間略
120 member_120 = 170
```

```
1 member = [180, 172, 166 ..., 170]
2 # 表示略
```

List

List的用法

假如說要得到 list 中第 i 個元素，就用 `xxx[i - 1]`
其中 $i - 1$ 稱為 **index**

fruits[0] fruits[1] fruits[2]
↓ ↓ ↓
1 fruits = ["apple", "banana", "cherry"]

Lists start from 0

也就是說，如果想取得fruits裡的第3個元素，
是要用 fruits[2]

Lists start from 0

```
1 fruits = ["apple", "banana", "cherry"]
2 print(fruits[0])
3     # output: apple
4 print(fruits[2])
5     # output: cherry
6
7
8 print(fruits[3])
9 ## IndexError: list index out of range
```

fruits[3]不存在，所以會出錯

len() : 回傳 list 的長度

```
1 fruits = ["apple", "banana", "cherry"]  
2 print(len(fruits))  
3      # output: 3
```


append() : 從 list 後面新增一個元素

```
1 fruits = []    # 一個空的list
2 fruits.append("apple")
3 fruits.append("banana")
4 fruits.append("cherry")
5 print(fruits)
6 # output:
7 # ["apple", "banana", "cherry"]
```

Lists start from 0

要改變 list 中的第 i 個元素的話，就用
`xxx[i - 1] = ooo`

```
1 fruits = ["apple", "banana", "cherry"]
2
3 fruits[0] = "grape"
4 fruits[1] = "kiwi"
5 print(fruits)
6
7 # output: ["grape", "kiwi", "cherry"]
```

Lists start from 0

pop() : 要刪除 list 中第 **i** 個元素，可以用 `xxx.pop(i - 1)`
若括號中沒有放入任何數字，那**預設**是刪除**最後一個**元素

```
1 fruits = ["grape", "kiwi", "cherry"]
2
3 fruits.pop(2) # 刪掉index==2的元素
4 print(fruits)
5 # output: ["grape", "kiwi"]
6
7 fruits.pop() # 刪掉最後一個元素
8 print(fruits)
9 # output: ["grape"]
```

`xxx[i]`

`len(xxx)`

`xxx.append(ooo)`

`xxx[i] = www`

`xxx.pop(i)`

Lists start from 0

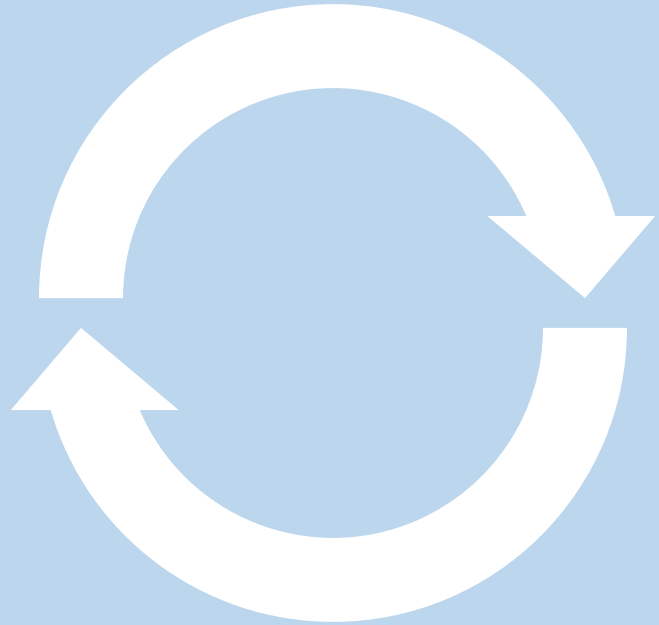
Lists start from 0

```
camp = ["N", "A", "T", "U", "R", "E"]
```

改成：

```
camp = ["N", "T", "U", "E", "E"]
```

```
1 camp = ["N", "A", "T", "U", "R", "E"]  
2  
3 # 目標: ["N", "T", "U", "E", "E"]  
4 # 應該跟你們一樣目標是NTUEE
```



Loop

- while loop
- for loop

用途：
重複執行相同的工作

Loop

while loop

while loop

當設定的條件被滿足時，程式會一直執行

```
1 while 判斷條件: 冒號也記得要打  
2     要執行的事
```

迴圈內要縮排
空4格 (或是按tab)

Loop

while loop

```
1 # EX1
2 x = 0
3 while x < 5: # 當x>=5時，會跳出迴圈
4     print(x) # 每次執行print這個動作
5     x += 1   # 並且x每次+1
```

output :
0
1
2
3
4

Loop

while loop

```
1 # EX2 你也可以這樣寫
2 x = 0
3 while True:    # 條件永遠成立，就是一直執行
4     print(x)   # 每次執行print這個動作
5     x += 1     # 並且x每次+1
6     if x >= 5: # 跳出條件加在這
7         break  # break表示跳出迴圈
```

output :
0
1
2
3
4

for loop

對一個範圍執行同一段程式。

這個範圍可以是list、string、或是一個數字區間

```
1 for xxx in 範圍: 冒號也記得要打  
2     要執行的事
```

迴圈內要縮排
空4格 (或是按tab)

Loop

for loop

```
1 # EX1 用list作範圍
2 A = [0, 1, 2, 3, 4] # A是一個list
3 for i in A:
4     print(i)
```

output :

0
1
2
3
4

Loop

for loop

```
1 # EX2 用一段數字區間作範圍
2 for i in range(5):
3     print(i)    # 一樣執行print的動作
4
5 # range(5)即從0數到5的範圍，但不包含5
6 # 即 0,1,2,3,4
```

output :

0
1
2
3
4

loop：用於重複執行相同的工作

- while loop：while 加條件
- for loop：for 用於範圍

但當然沒硬性規定你怎麼寫
想怎麼寫就怎麼寫

德國著名數學家高斯幼年時代聰明過人，上學時，有一天老師出了一道題讓同學們計算：

$$1 + 2 + 3 + 4 + \dots + 99 + 100 = ?$$

老師出完題後，全班同學都在埋頭計算，幸好小高斯是個Python天才，打了幾行程式，就能解出這道難題

總之就是叫你算 1 加到 100 啦

如果覺得太簡單，就用for跟while都寫寫看。
如果還是太簡單，就教教你隔壁的吧~

def:

Function

前面出現過的function：

- `print(x, y)`
- `math.sqrt(25)`
- `math.log10(100)`
- `fruits.append("apple")`

Function

呼叫 Function

特徵：

function名字(用逗號分隔的參數)

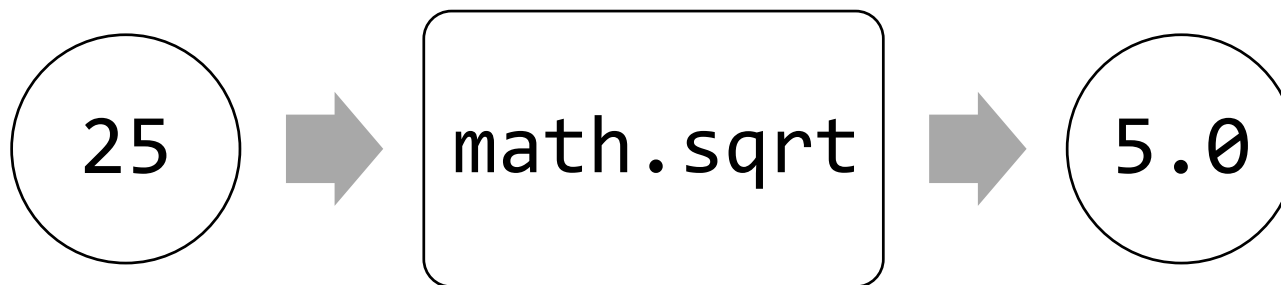
- print(x, y)
- math.sqrt(25)
- math.log10(100)
- fruits.append("apple")

Function

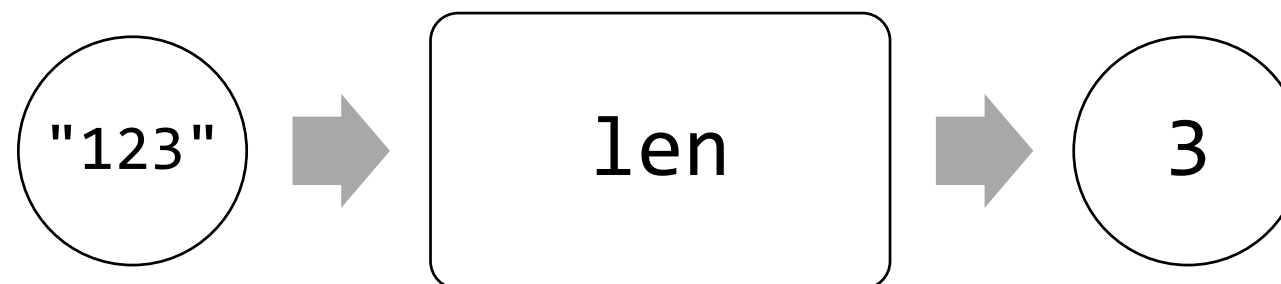
呼叫 Function

function根據傳入的資料做某些事，回傳某個東西

- `math.sqrt(25)`



- `len("123")`



先做完function要做的事，再把結果填到呼叫的位置

Function

呼叫 Function – 範例

→ `a = math.sqrt(25)`
`a = 5.0`

Function

呼叫 Function – 範例

```
→ a = math.sqrt(25) / math.sqrt(4)  
a = 5.0 / math.sqrt(4)  
a = 5.0 / 2.0  
a = 2.5
```

Function

呼叫 Function – 範例

複雜一點點的

➔ `a = [sum([1, 2]), len("1234")]`
`a = [3, len("1234")]`
`a = [3, 4]`

Function

定義 Function

```
1 def function名稱(參數1, 參數2): 冒號記得要打  
2     要執行的事  
3     return xxx
```

function內要縮排
空4格 (或是按tab)

※ 參數要幾個都可

Function

定義 Function

```
1 # EX
2 def summation(num_list):
3     result = 0
4     for number in num_list:
5         result += number
6     return result
7
8 a = summation([1, 2, 3])
9 print(a)
10 # output: 6
```

[1,2,3]



summation



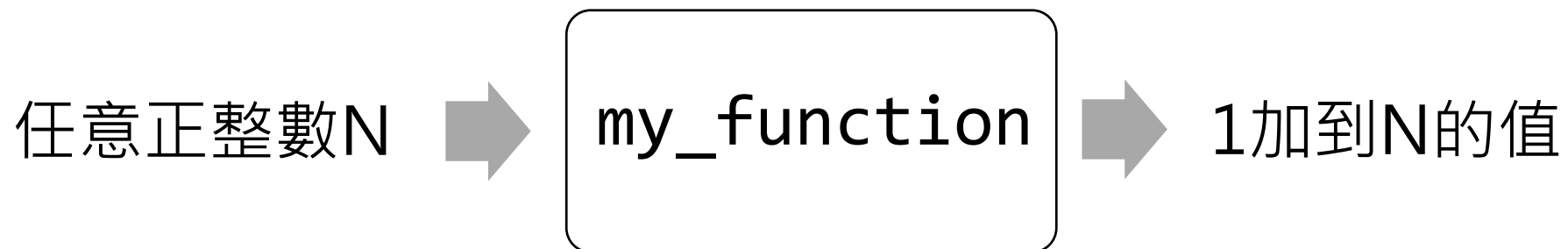
6

return: function呼叫完會被換成什麼

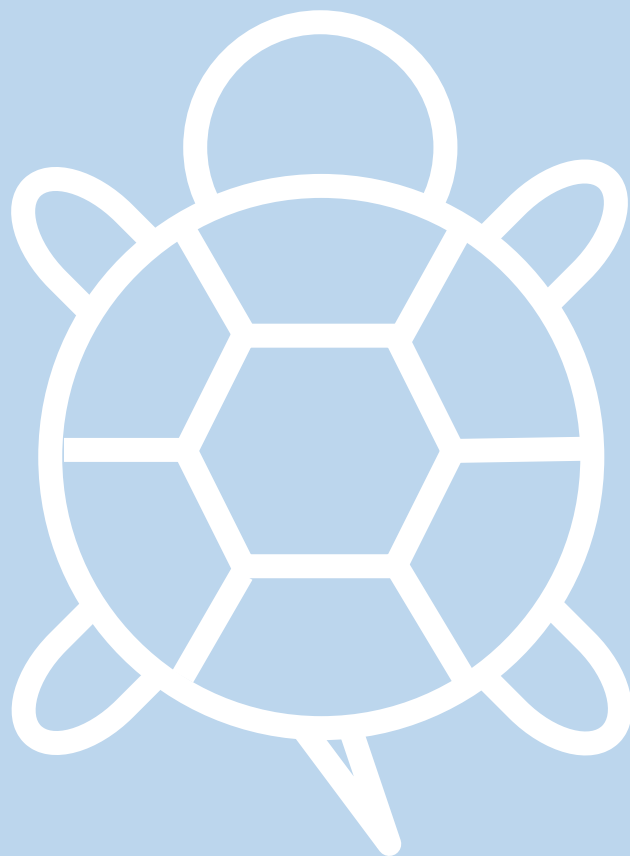
Function Practice

在loop的練習中，我們做過1加到100

現在我們來練習把loop的練習包進function裡



```
def my_function(N):  
    xxx  
    ooo  
    return www
```



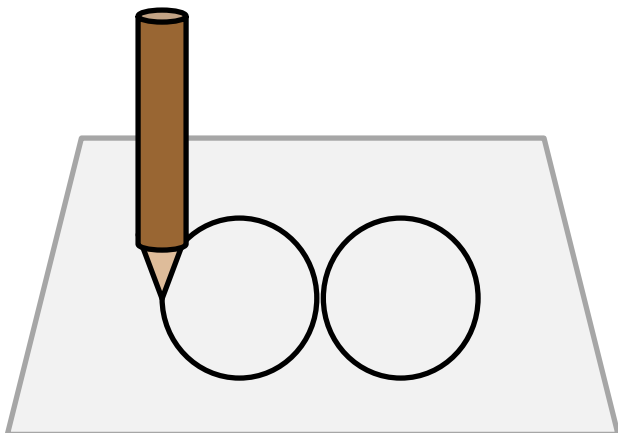
Turtle

- Python的繪圖模組
- 想像拿一支筆在螢幕上畫畫，那就對了！

```
1 import turtle as tt
2 # 使用別人寫好的turtle模組
3 # as tt 像是幫它取個名字
4
5 tt.xxx()
6 # 用tt.來開頭，以使用turtle內的function
7
8 tt.done()
9 # 一定要加這個結尾
```

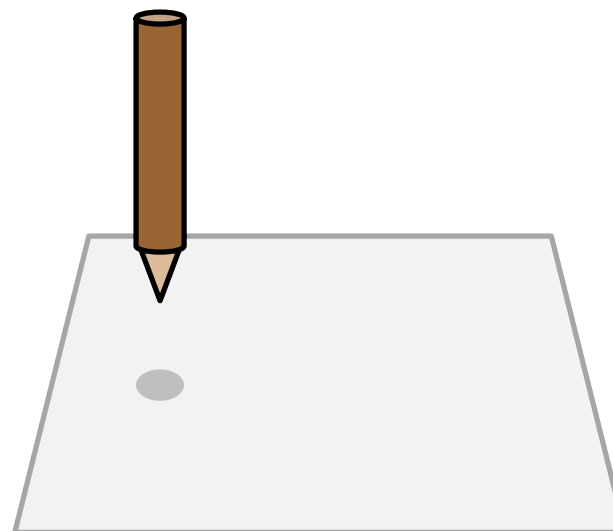
`tt.pendown()`

下筆



`tt.penup()`

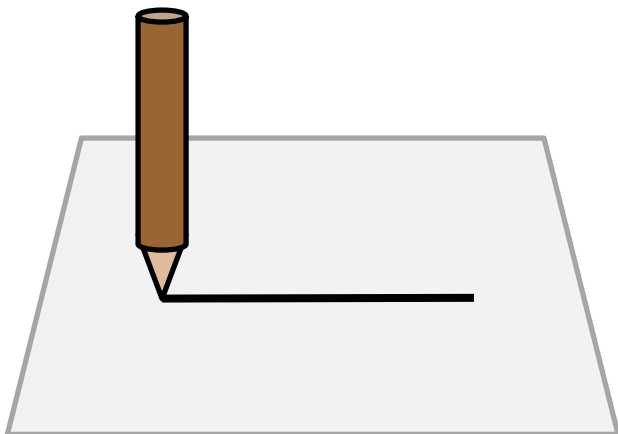
提筆



```
tt.forward(x)
```

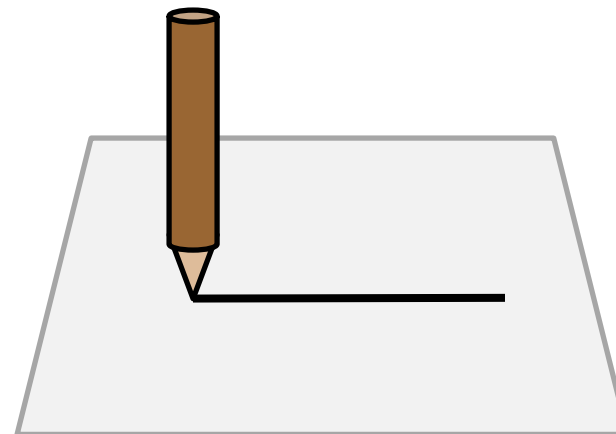
前進 x 像素

x要是正數



```
tt.goto((x,y))
```

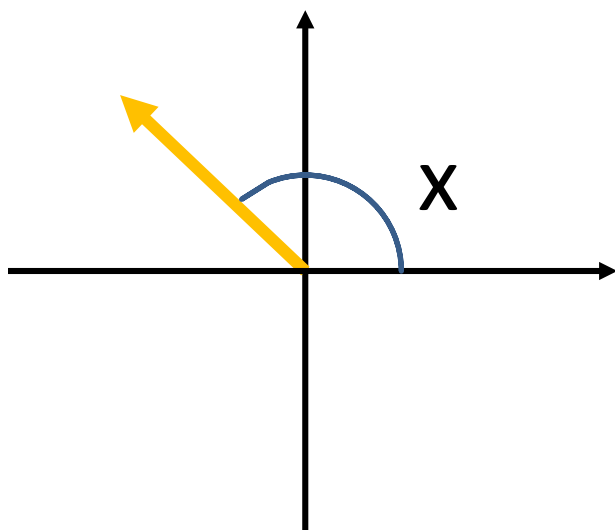
瞬移到座標(x,y)



`tt.setheading(x)`

轉到 x 度

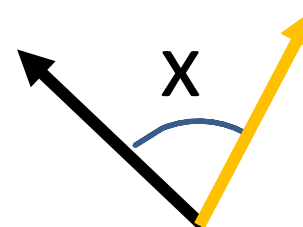
絕對



`tt.left(x) / tt.right(x)`

左右轉 x 度

相對



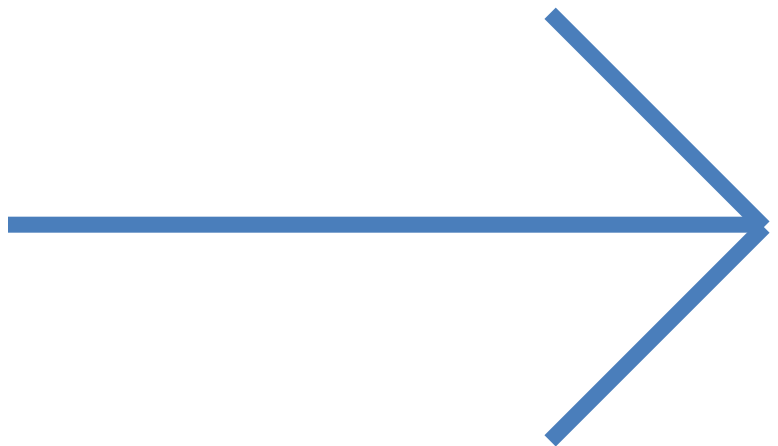
$x : 0 \sim 360$

函數名稱	函數功能
pendown()	下筆 (留下筆跡)
penup()	提筆 (不留筆跡)
forward(x)	前進 x 像素
goto((x,y))	順移到座標(x,y)
setheading(x)	設定箭頭方向到 x 度 (絕對)
right(x)	筆向右轉 x 度 (相對)
left(x)	筆向左轉 x 度 (相對)


```
1 import turtle as tt # 開頭必須
2 # tt.speed(1) # 若覺得太快，加上這行
3 tt.forward(100) # 前進100像素
4 tt.setheading(270) # 轉到270度
5 tt.forward(100) # 前進100像素
6 tt.setheading(0) # 轉到0度
7 tt.forward(100) # 前進100像素
8
9 tt.done() # 結尾必須
```

Try it!!

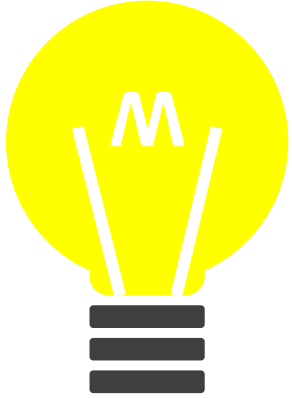
還有另外兩個階梯，可以跑跑看



開頭記得 `import turtle as tt`
結尾要寫 `tt.done()`

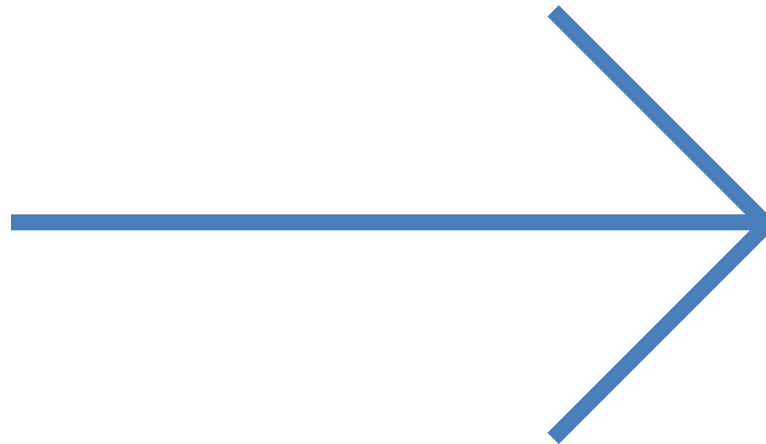
只要畫得出來就可以，沒有對與錯的差別！

如果還有時間，練習把程式改寫成function！



Hint :

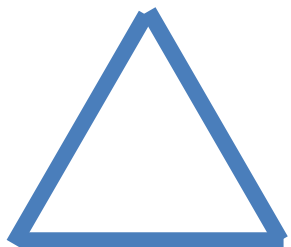
1. 由起點往前 100 像素
2. 旋轉到 135 度
3. 往前 40 像素
4. 倒退 40 像素
5. 旋轉到 225 度
6. 往前 40 像素



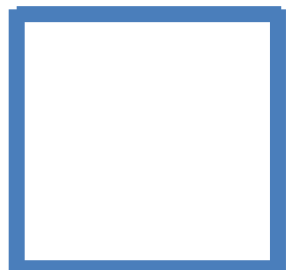
```
1 import turtle as tt # 開頭必須
2 tt.forward(100)
3 tt.setheading(135)
4 tt.forward(40)
5 tt.backward(40)
6 tt.setheading(225)
7 tt.forward(40)
8 tt.done() # 結尾必須
```

Turtle

Practice – 正多邊形



轉120度



轉90度

...



1. 畫一個三角形
2. 畫一個正方形
3. 畫一個正n邊形

試著寫成 function

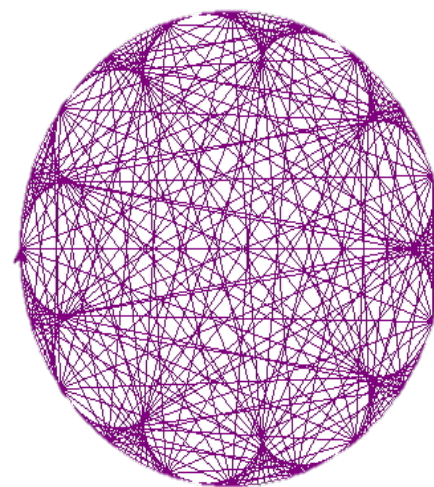
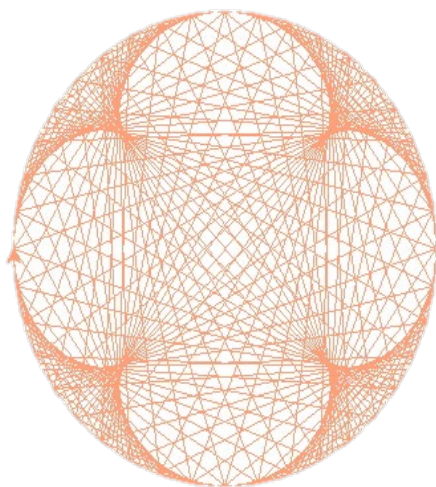
```
1 import turtle as tt
2 def polygon(x):
3     for i in range(x):
4         tt.forward(100)
5         tt.left(360/x)
6 polygon(3)
7 polygon(4)
8 polygon(10)
9 tt.done()
```

`color()`：更換筆的顏色

例如："red", "aqua", "light salmon"







可以調整的參數：
size 盡量不要超過300
multi 要是整數
color 隨你調

有興趣可以去看影片
[Times Tables, Mandelbrot
and the Heart of Mathematics](#)

Hello Python

Answer

```
1 x = 3
2 y = 6
3
4 temp = x
5 x = y
6 y = temp
7
8 print(x, y)
9 # output: 6 3
```

```
4
5 x, y = y, x
6
```

if - else

Answer

```
1 N = 2019
2 if N % 4 == 0:
3     if N % 100 == 0:
4         if N % 400 == 0:
5             print("閏年")
6         else:
7             print("平年")
8     else:
9         print("閏年")
10 else:
11     print("平年")
```

检查一下

2019: 平年
2020: 閏年
2100: 平年
2000: 閏年

List Answer

```
1 camp = ["N", "A", "T", "U", "R", "E"]
2
3 camp.pop(4)
4 camp.pop(1)
5 camp.append("E")
6
7 print(camp)
8 # output: ["N", "T", "U", "E", "E"]
```

Loop Answer

```
1 # while loop
2 a, i = 0, 0
3 while i <= 100:
4     a += i
5     i += 1
6 print(a)      # output: 5050
```

```
1 # for loop
2 a = 0
3 for i in range(100): # 或是寫range(101)
4     a += (i+1)
5 print(a)      # output: 5050
```

Function

Answer

```
1 def my_function(N):  
2     a = 0  
3     for i in range(N):  
4         a += (i+1)  
5     return a  
6 print(my_function(100))  
7 # output: 5050
```