

ANN

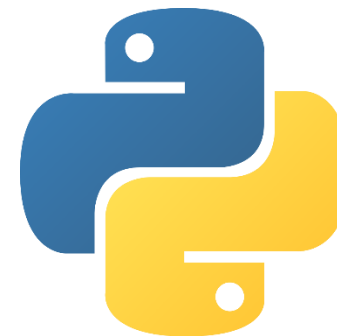
# CH05 Python 基本應用

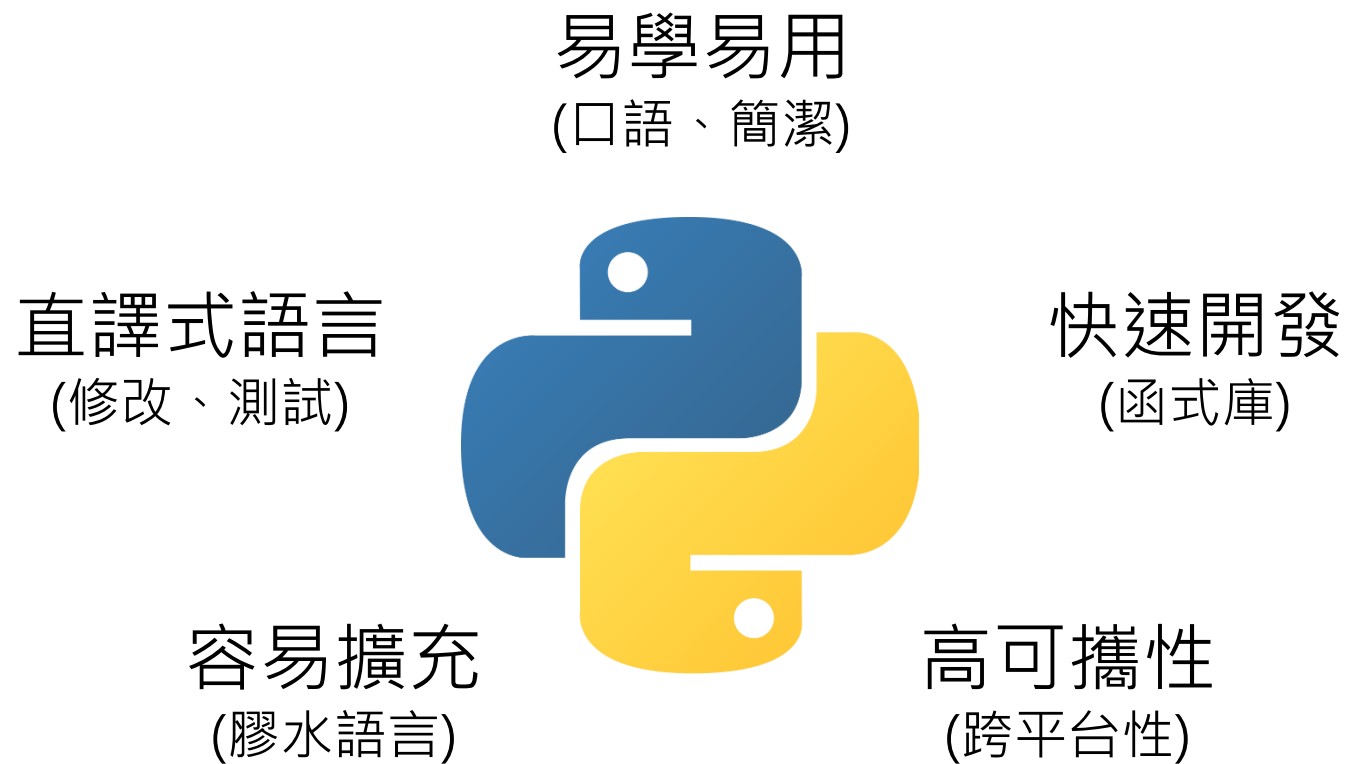
#1

---

## 5.1 What is Python ?

- Python 由荷蘭程式設計師 Guido van Rossum 於 1989 年創建
- 他是英國電視短劇 Monty Python's Flying Circus (蒙提派森的飛行馬戲團) 的愛好者, 因此用 Python 做為新語言的名稱
- 官網([www.python.org](http://www.python.org))也以蟒蛇圖案做為標誌

















































膠水語言：能夠調用其他語言編寫的程式或模組(例如：C++)進行整合，獲取其執行的結果

# 2018 Top Programming Languages

## 趨勢

Language Rank	Types	Trending Ranking
1. Python	  	100.0
2. C++	  	96.7
3. Java	  	94.6
4. C	  	93.7
5. Go	 	85.5
6. JavaScript	 	80.8
7. PHP		79.9
8. Scala	 	78.6
9. Ruby	 	77.2
10. HTML		75.5

## 就業

Language Rank	Types	Jobs Ranking
1. Python	  	100.0
2. Java	  	99.2
3. C	  	98.8
4. C++	  	94.6
5. C#	  	86.2
6. JavaScript	 	85.7
7. Assembly		83.4
8. PHP		83.1
9. HTML		81.3
10. Scala	 	76.5

## 測試 Spyder / Jupyter Notebook 執行 python 程式

- 進入 Jupyter Notebook / Spyder
- 輸入程式如下  
**`print('Hello python!')`**
- 按"Enter"鍵執行後出現  
Hello python !
- Jupyter Notebook 需同時按"Shift"或"Ctrl"+"Enter"才會執行程式

## 5.2 基本概念 – 算數運算

- 輸入程式如下

**4\*5**

**7/5**

**3\*\*2**

- 運算符號

\*：乘法

/：除法

\*\*：次方

- 科學表示：e 表示 10 的次方(也可以是 E)

## 基本概念 – 資料型態(data type)

- 資料的性質有：整數、小數、字串等類別
  - int：整數(integer)
  - float：浮點數(floating-point number)
  - str：字串(string)
- 利用 **type()** 函數，可以查詢資料類別
- 輸入程式如下

**type (10)** #整數

**type (2.718)** #浮點數

**type ('hello')** #字串

## 基本概念 – 資料型態 (data type)

- 資料型態之間可以相互轉換

**a=100 #整數**

**print(a)**

**print(type(a))**

**print('-'\*30)**

**b=float(a) #float()是內建函式，可將型態轉為浮點數**

**print(b)**

**print(type(b))**

**print('-'\*30)**



## 基本概念 – 字串(string)

- str 字串是用來表示文字資料
  - 使用單引號 ( ' ) 或雙引號 ( " ) 包起來的資料類型
  - \ : 跳脫(escape)
    - 使用'單引號'的字串，如果在字串中要使用'單引號'，必須使用符號 \ 來跳脫處理，避免被認為字串結束

```
x = 'I\'m Eric.'
```

```
print (x)
```

- 同理，字串中要使用雙引號或 \ 字元時，也必須以 \ 跳脫

```
file_path = 'C:\\pythowork\\test.py'
```

```
print (file_path)
```

# 基本概念 – 變數(variables)

- Python 是動態型態的程式語言
  - 使用變數不需要事先宣告，或是指定資料型態
  - 會視狀況自動決定變數型態
- 定義變數名稱規則
  - 可用英文字母(A-Z, a-z)、數字(0-9)、底線( \_ )組成
  - 英文字母大小寫有別 ( case-sensitive )
  - 變數名稱不得以數字開頭
    - \_Var、my\_X 是合法的變數名稱
    - abcd 與 Abcd 會被視為不同的變數
    - 6abc 不能使用!

## 基本概念 – 變數(variables)

- 井字號「#」做為註解符號，同一列#字號後的任何字不被執行

**a = 4** #指定變數，給予變數預設值

**b = a \* 4.5** #變數運算

**c = (a+b)/2**

**a = "Hello World"** #變數自動更新型態(整數→字串)

**type (a)** #檢查變數型態

**print (a)** #輸出結果

**print (b)**

**print (c)**

**type(b\*c)**

## 基本概念 – 變數(variables)

- 如果程式中需要讀取使用者所輸入的資料
- 可以使用 input 函式

**X = input('Please give me a number:')**

#要求使用者輸入一個數字，這個數字會儲存在 X 變數中

**print (X)**

## 基本概念 – 清單(list)

- 資料可整合變成清單(或叫陣列 array)
- 使用中括號 [ ] 來表示
  - a=[1,2,3,4,5]** #建立清單
  - print(a)** #輸出清單內容
  - len(a)** #取得清單的長度
  - a[0]** #取得最初(第一個)的元素
  - a[4]** #取得第五個元素
  - a[4]=99** #元素更新代入
- 元素的存取用 a[0] 的方式進行，[ ]裡面的元素稱為「索引值」(index)
- 索引值從『0』開始

## 基本概念 – 清單(list)

- List 裡的資料可以是任何資料型態，例如字串

**H = 'Hello World!'** #建立字串(清單)

**print (H)** #輸出清單內容

**len(H)** #取得清單的長度

**H[0]** #取得最初(第一個)的元素

**H[4:8]** #取得第五~第八個元素

- 試試看 **K = [1, 5, 'JQK', 9, "xyz"]**

- 索引值從『0』開始

## 基本概念 – 切片(slicing)

- 使用切片存取清單(陣列)中的部分資料

**print(a)** #輸出清單內容

**a[0:2]** #取得對應索引值第0~第2個元素(不包含第2個!)

**a[1:]** #取得對應索引值第1個到最後一個元素

**a[:3]=99** #取得最初到第三個元素(不包含第3個!)

**a[:-1]** #取得最初到最後元素的前一個元素

**a[:-2]** #取得最初到最後元素的前二個元素

- 範圍的索引值包括左值及右值，取出資料的範圍是從左值開始至右值的前一個元素
- 索引值 -1：對應最後一個元素(倒數第一個)
- 索引值 -2：對應最後的前一個元素(倒數第二個)

## 基本概念 – 字典(dictionary)型態

- 字典(dict)有"英文單字 – 中文翻譯"這種一對一的關係
  - 英文單字：key；中文翻譯：value
- dict 是一群 key 與 value 配對的集合
- 使用大括號 {} 來表示 dict
- key : value 之間以冒號：做為間隔
- 如果有好幾組 key : value，使用逗號，間隔

**me={'height' : 170}** #建立字典 dict

**me['height']** #輸出 key 對應的 value

**me['weight']=70** #新增一組 key : value

**Print (me)** #輸出結果

**{'height': 175, 'weight': 82}**



## 基本概念 – 字典(dictionary)型態

- dict 字典例子

**J = {'python': 'Large constricting snakes', 'Bach': 'German organist and composer, 1685–1750'} #建立字典 dict**

**J['python'] #輸出 key 對應的 value**

**J['Bach'] #輸出 key 對應的 value**

**len(J) #查詢 dict 物件中有幾組 key : value**

**J['organist']='A person who plays the organ' #新增 key : value**

**J['python'] = 'A powerful scripting language' #修改 key 對應的 value**

**del J['Bach'] #刪除一組 key : value**

**print (J) #輸出結果**

## 基本概念 – 布林(bool)型態

- 只有兩種值：True 與 False (字首一定要大寫!)
- Boolean 型態使用的運算子包括：and、or、not

**hungry= True** #肚子餓

**sleepy= False** #不想睡覺

**type(hungry)** #檢查 hungry 的資料型態

**not hungry** #肚子不餓?

**hungry and sleepy** #肚子餓且想睡覺?

**hungry or sleepy** #肚子餓或想睡覺?

## 基本概念 – if 陳述式

- 某些時候，需要設立條件判斷，以便分開執行不同程式碼
- 可以使用 if/else

**hungry= True** #肚子餓

**if hungry:** #使用 if 陳述式，以冒號「:」表示”區塊”敘述的開始

**print(“I’m hungry”)** #符合 if 條件時執行(then)

**else:** #不符合 if 條件時

**print(“I’m not hungry”)** #不符合 if 條件時執行

- Python 程式中的空格具有意義
- 使用四個空格(或 Tab 鍵)來縮排，代表區塊(block)敘述

## 基本概念 – if 陳述式

- 使用 if/elif/else

**G = input('Please enter an integer: ') #要求使用者輸入一個整數**

**G = int(G) #將資料型態由字串變成整數**

**if G > 0:**

**print("Positive") #符合 if 條件時執行**

**elif G==0: #判斷條件不只一個時，可以再加上 elif**

**print("Zero")**

**else: #不符合 if 條件時**

**print("Negative") #不符合 if 條件時執行**

- 條件判斷式會用 True 或 False 來判斷，只有結果為 True 時，才會執行內部區塊的程式碼

## 基本概念 – for 陳述式

- 如果要執行"迴圈"，可以使用 for  
**for i in [1,2,3]:** #使用 **for** 陳述式，依序存取(清單)資料集合的各元素  
**print(i)** #輸出 i
- 冒號「:」表示"區塊"敘述的開始
- 四個空格(或 Tab 鍵)縮排，表示"區塊"敘述

## 基本概念 – 函數

- 具有整合性的處理可以定義為"函數"(Function)

**def hello():** #定義 hello 函數

**print("Hello World!")** #函數內要執行的程式

**hello()** #執行 hello() 這個函數

- 函數也可以設定參數

**def hello(object):** #定義 hello 函數與設定參數

**print("Hello " + object + "!")** #使用「+」可以連接字串

**hello("cat")** #執行 hello() 函數，object 是 cat

## 5.3 Python Script 檔案

- Python 程式碼可以儲存成檔案，副檔名為：py
- 將之前的程式碼儲存
  - 使用 jupyter notebook 時，預設副檔名為：ipynb
- 在命令提示字元視窗，可利用下列程式碼執行該檔案  
**python xxxx.py**

## 類別(class)

- int或str等資料型態屬於「嵌入式」資料，是 Python 事先定義好的
- Python 也可以自行定義新的類別，建立專屬的資料型態
- 並自訂給類別使用的函數：方法(method)

- 使用關鍵字 class 定義類別(class)，class 格式如下：

**class 類別名稱：**

**def \_\_init\_\_(self, 參數, ...): #建構子(constructor)**

...

**def 方法名稱1(self,參數, ...): #方法1**

...

**def 方法名稱2(self,參數, ...): #方法2**



## 類別(class)

- `__init__` 是特別用來執行"初始化"的方法，也稱為建構子(constructor)
- 在建立 class 的實體(instance)時，只會呼叫一次

**class 類別名稱：**

**def `__init__`(self, 參數, ...): #建構子(constructor)**

- 在方法的第一參數之中，要清楚寫出代表本身(實體)的"self"

**def 方法名稱1(self,參數, ...): #方法1**

```
class Man : #定義 Man 這個新類別
    def __init__(self, name): #constructor，實體self，建立參數name
        self.name = name #使用參數name將實例變數self.name初始化
        print("Initialized!") #告知初始化成功
    def hello(self): #方法1：hello
        print("Hello " + self.name + "!") #帶入實例變數self.name後輸出
    def goodbye(self): #方法2：goodbye
        print("Good-bye " + self.name + "!")
m = Man("David") #使用類別Man與給定參數name為David，建立物件m
m.hello() #使用類別Man的方法hello
m.goodbye()
```

## 5.4 外部函式庫(需要 import 載入的步驟)

- Numpy
  - Numerical Python
  - 是用來快速計算數值的函式庫
  - 提供 N-dimensional array object (ndarray) 的資料結構
    - ndarray：同質且固定大小的多維度陣列物件
  - 提供高難度數學演算法、線性代數、矩陣運算及隨機數生成

**import numpy as np** #載入 numpy 之後簡稱為 np

## 外部函式庫 – Numpy

- 使用 `np.array()` 語法，可建立 Numpy N維陣列
  - 一維：向量；二維：矩陣；三維以上：張量
- `np.array()` 是從 Python 清單中取得參數，建立 Numpy 陣列 (`numpy.ndarray`)

**`na = np.array([1.0, 2.0, 3.0])`** #建立 numpy 陣列

**`print(na)`** #輸出陣列 `na` 結果

**`type(na)`** #輸出陣列 `na` 資料型態

## 外部函式庫 – Numpy

- 使用 `np.array()` 語法，建立 Numpy 一維陣列
  - 使用符號 ( `[]` )

**`na1 = np.array([1.0, 2.0, 3.0])`**

**`na2 = np.array([2.0, 4.0, 6.0])`**

**`na1 + na2`** #陣列對應元素相加

**`na1 - na2`** #陣列對應元素相減

**`na1 * na2`** #陣列對應元素相乘，**element-wise product**

**`na1 / na2`** #陣列對應元素相除

**`na1/2.0`** #陣列元素與純量(**scalar**)相乘，稱作廣播(**broadcast**)功能

## 外部函式庫 – Numpy

- 使用 `np.array()` 語法，建立 Numpy 二維陣列
- 使用符號 `( [[ ]] )`

**`nb1 = np.array([[1.0, 2.0], [3.0, 4.0]])`**

**`nb2 = np.array([[3.0, 0], [0, 6.0]])`**

**`nb1.shape`** #輸出陣列的大小(形狀)

**`nb1.dtype`** #輸出陣列元素的資料型態

**`nb1 + nb2`** #陣列對應元素相減

**`nb1 * nb2`** #陣列對應元素相乘，element-wise product

**`nb1 / nb2`** #陣列對應元素相除

**`nb1*5.0`** #陣列元素與純量(scalar)相乘：廣播(broadcast)

## 外部函式庫 – Numpy

### ● 廣播範例

Diagram illustrating scalar broadcasting:

1	2
3	4

 \* 10 = 

1	2
3	4

 \* 

10	10
10	10

 = 

10	20
30	40

Diagram illustrating vector broadcasting:

1	2
3	4

 \* 

10	20
----	----

 = 

1	2
3	4

 \* 

10	20
10	20

 = 

10	40
30	80

## 外部函式庫 – Numpy

- 矩陣與矩陣相乘
  - 必須使用 ndarray 的 .dot( ) 方法或者 np.dot( ) 函數
- 前面矩陣的行數 = 後面矩陣的列數，兩矩陣才可以相乘

$$A_{m \times p} \times B_{p \times n} = (AB)_{m \times n} = C_{m \times n}$$

```
A1 = np.array([  
    [4, -3, -2],  
    [1, -1, 3],  
]) #設定矩陣A1為 2*3 矩陣
```

```
B1 = np.array([  
    [2, -5],  
    [3, 1],  
    [-1, 4],  
]) #設定矩陣B1為 3*2 矩陣
```

```
C1=A1.dot(B1) #A1與B1相乘，變成C1 2*2 的矩陣
```



## 外部函式庫 – Numpy

- Numpy 陣列存取元素使用符號 [ ]
- 索引值一樣從 0 開始

```
nc1 = np.array([[51, 55], [14, 19], [0, 4]])
```

```
print (nc1) #輸出陣列
```

```
nc1.shape #輸出陣列的大小(形狀)
```

```
nc1[0] #陣列第0列的對應元素
```

```
nc1[0][1] #陣列(0, 1)對應元素
```

```
nc1[1][1] #陣列(1, 1)對應元素
```

## 外部函式庫 – Numpy

- 可以使用 for 陳述式(迴圈)存取 Numpy 陣列元素

```
nc1 = np.array([[51, 55], [14, 19], [0, 4]])
```

```
for row in nc1:
```

```
    print(row) #輸出陣列 row 向量
```

- 直接使用陣列進行存取

```
nc1 = nc1.flatten() #把 nc1 轉換成輸出陣列 row 向量
```

```
nc1[np.array([0, 2, 4])] #取得 nc1 內索引值為第0、2、4個的元素值
```

## 外部函式庫 – Numpy

- 運用以下方法，可以單獨取出滿足條件的元素
- 例如只從 nc1 取出 15 以上的值

**nc1 > 15**

**array([ True, True, False, True, False, False])**

**nc1[nc1>15]**

**array([51, 55, 19])**

- 對 Numpy 陣列使用不等號之類的運算子，結果會變成布林陣列
- 利用布林陣列，可以取出對應為 True 的元素

## 外部函式庫 – Pandas

- 基於 NumPy，最初作為金融分析工具，為了解決資料分析任務而開發
- 提供高效操作大型資料集所需的函式和方法基礎圖等
- 為時間序列分析提供很好的支援
  - 一維的 Series、二維的 DataFrame、三維的 Panel
  - 名稱由來：Pan(el) + da(ta) + s

## 外部函式庫 – Pandas

- Series：一維資料結構，與 Numpy 的一維陣列 array 類似
  - 融合了字典/ndarray優點，可以運用字典/ndarray索引和函式
  - 與清單(list)的數據結構相近
    - list 的元素可以是不同的數據類型
    - Array 和 Series 只允許儲存相同的數據類型，才能提高運算效率
- DataFrame是表格型的資料結構，它含有一組有序的列，每列可以是不同的值型別(數值、字串、布林值等)
  - 有行索引也有列索引，可以被看做由Series組成的字典(共用同一個索引)

## 外部函式庫 – Pandas

**import pandas as pd** #載入 pandas 之後簡稱為 pd

**from pandas import Series, DataFrame**

**Series({'a':3, 'b':2})** #建立 Series

**d1 = {'col1': [1, 2], 'col2': [3, 4]}**

**df1 = pd.DataFrame(data=d)**

**df2 = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),  
                    columns=['a', 'b', 'c'])**

## 外部函式庫 – Matplotlib

- ◉ 是基礎繪圖函式庫，指令較為複雜
  - ◉ 其他可用的函式庫：seaborn、plotly、ggplot
- ◉ 可將資料或結果視覺化呈現
  - ◉ 散點圖、折線圖、長條圖、直方圖、圓餅圖、箱形圖等

**import matplotlib.pyplot as plt**

**X = np.arange(0, 6, 0.1)** #從 0~6，以 0.1 為單位產生資料

**Y = np.sin(X)** #套用 Numpy 的 sin 函數，產生資料 Y = sin(X)

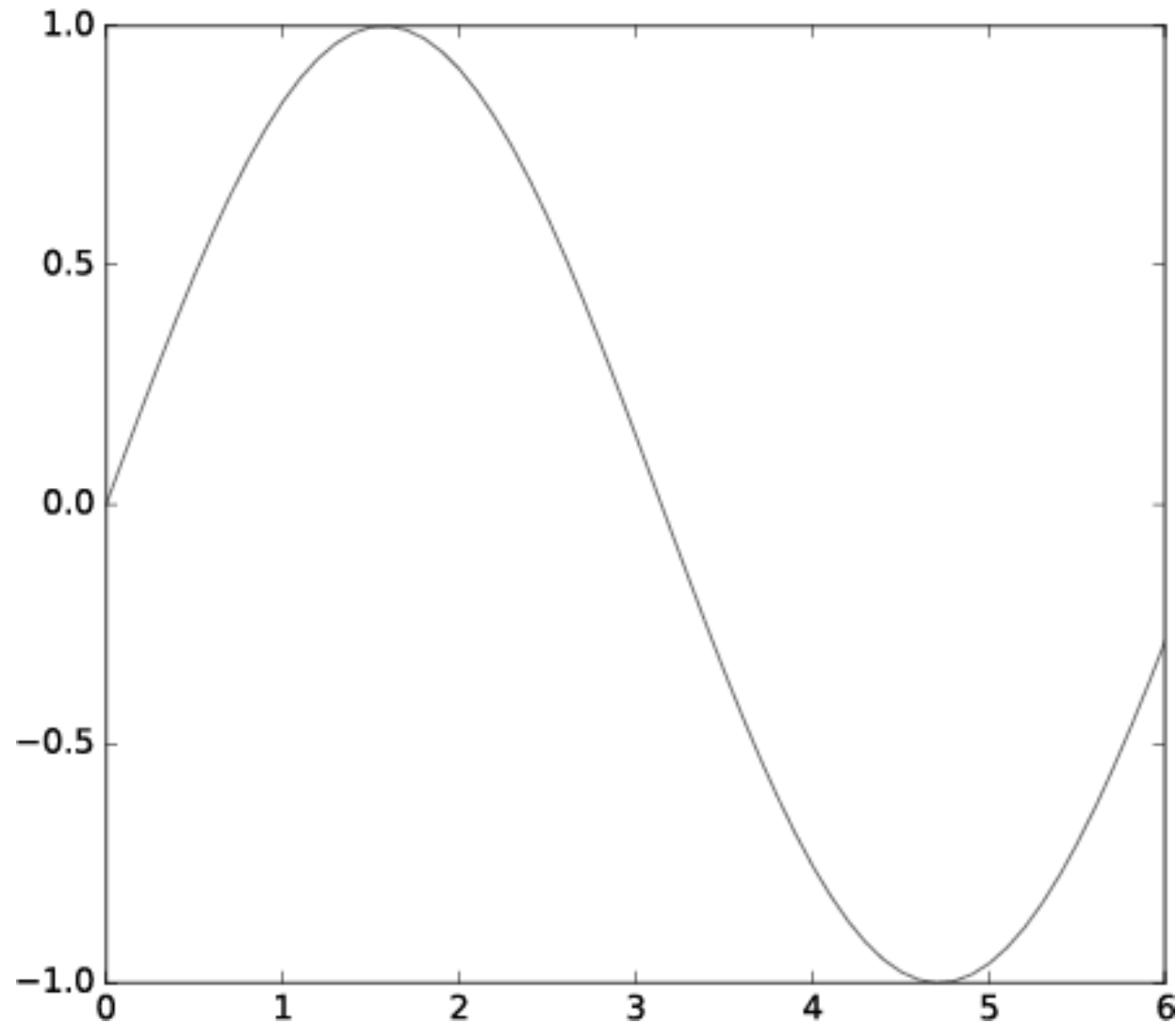
**plt.plot(X, Y)** #繪製圖表

**plt.show()** #顯示圖表

**$Y = \text{np.sin}(X)$**

#套用 Numpy 的 sin 函數

#產生資料  $Y = \sin(X)$



**$X = \text{np.arange}(0, 6, 0.1)$**  #從 0~6，以 0.1 為單位產生資料



## 外部函式庫 – Matplotlib

- 練習

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
#建立資料
```

```
X = np.arange(0, 6, 0.1) #從 0~6，以 0.1 為單位產生資料
```

```
Y1 = np.sin(X) #套用 sin 函數，產生資料 Y1 = sin(X)
```

```
Y2 = np.cos(X) #套用 cos 函數，產生資料 Y2 = cos(X)
```

```
plt.plot(X, Y1) #繪製圖表
```

```
plt.show() #顯示圖表
```

## 外部函式庫 – Matplotlib

- 練習-續

### #繪製圖表細節

`plt.plot(X, Y1, label="sin")` #繪製圖表

`plt.plot(X, Y2, linestyle="--",label="cos")` #用虛線繪圖

`plt.xlabel("X")` # x 軸標籤

`plt.ylabel("Y")` # y 軸標籤

`plt.title("sin & cos")` #圖表標題

`plt.legend()` #顯示圖例

`plt.show()` #顯示圖表

## 外部函式庫 – Matplotlib

- 如何用來載入圖檔，顯示影像

```
import matplotlib.pyplot as plt
```

```
from matplotlib.image import imread #使用 image 模組內的 imread
```

```
img = imread('lena.png') #載入影像(需設定適當路徑)
```

```
plt.imshow(img) #顯示影像
```

```
plt.show() #顯示影像
```

這裡假設圖檔儲存在目前的工作目錄中

應根據使用環境，更改成適當的檔案名稱或檔案路徑

路徑設定範例：C:/Users/pc12/lena.png (**Note**：要用"/"，而不是"\")

## 外部函式庫 – 其他

- SciPy
  - Scientific Python
  - 能與 NumPy 陣列一起工作，處理插值、積分、優化等工作
- Scikit-learn
  - Machine Learning 常用函式庫
  - 包括完整的資料處理流程：
    - Preprocessing
    - Dimensionality reduction
    - Model selection
    - Mining/Learning
    - Experiment