Python Modules

- A module allows you to logically organize your Python code.
  - Grouping related codes into a module.
- A module is a Python object with arbitrarily named attributes that you can bind and reference.
  - Simply, a module is a file consisting of Python codes.
- A module can define functions, classes and variables.
  - Also includes runnable code.

The *import* Statement

- You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax:

```
import module1[, module2[,... moduleN]
```

- When the interpreter encounters an import statement, it imports the module in the search path.
- A search path is a list of directories that the interpreter searches before importing a module.

```
In [3]: import sys

In [4]: sys.modules.keys()
```

```
Out[4]: dict_keys(['sys', 'builtins', '_frozen_importlib', '_imp', '_thread', '_warnings', '_weakref', 'zipimport',
'_frozen_importlib_external', '_io', 'marshal', 'nt', 'winreg', 'encodings', 'codecs', '_codecs', 'encodings.aliases',
'encodings.utf_8', '_signal', '__main__', 'encodings.latin_1', 'io', 'abc', '_abc', '_bootlocale', '_locale',
```

Example

- To import the module *samply_module.py*, you need to put the following command at the top of the script:

samply_module.py | untitled7.py | untitled0.py | untitled2.py

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 18 16:19:33 2020

@author: user
"""

def sample_func():
    print('Hello!')
    print("__name__", __name__)


sample_func()
```

```
Hello!
__name__ __main__
```

temp.py | samply_module.py | untitled1.py*

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 18 16:21:50 202

@author: user
"""


import samply_module
if __name__ == '__main__':
    samply_module.sample_func()
```

```
Hello!
__name__ samply_module
Hello!
__name__ samply_module
```

Example

- xmath.py

```python
def max(a, b):
    return a if a > b else b
def min(a, b):
    return a if a < b else b

def sum(*numbers): # numbers 接受可變長度引數
    total = 0
    for number in numbers:
        total += number
    return total

pi = 3.141592653589793
e = 2.718281828459045
```

```
# import xmath
3.14159265359
10
15
# import xmath as math
2.71828182846
# from xmath import min
5
```

```python
import xmath
print '# import xmath'
print xmath.pi
print xmath.max(10, 5)
print xmath.sum(1, 2, 3, 4, 5)

print '# import xmath as math'
import xmath as math # 為 xmath 模組取別名為 math
print math.e

print '# from xmath import min'
from xmath import min  # 將 min 複製至目前模組，不建議 from modu import *，易造
print min(10, 5)
```

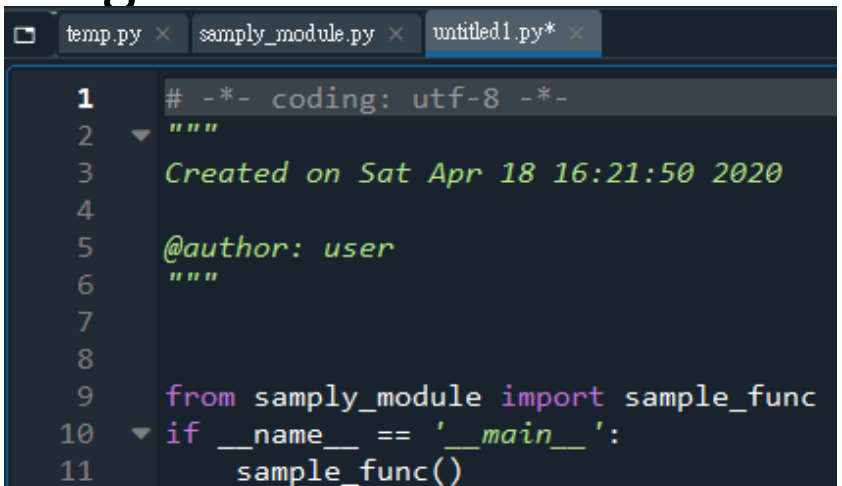
The *from...import* Statement

- Python's *from* statement lets you import specific attributes from a module into the current namespace.
- The *from...import* has the following syntax:

```
from modname import name1[, name2[, ... nameN]]
```

- For example, to import the function fibonacci from the module fib, use the following statement:

```
from fib import fibonacci
```



The *from...import* * Statement:

- It is also possible to import all names from a module into the current namespace by using the following import statement:

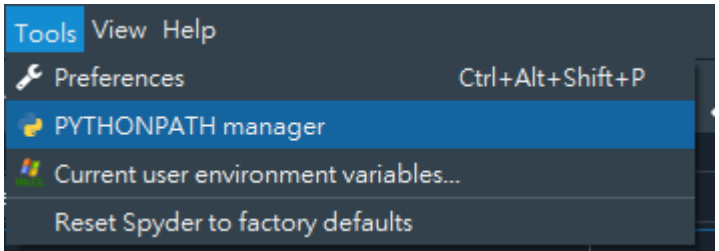
```
from modname import *
```

Locating Modules:

- When you import a module, the Python interpreter searches for the module in the following sequences:
  - The current directory.
  - If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
  - If all else fails, Python checks the default path.
    - On UNIX, this default path is normally /usr/local/lib/python/.

The *PYTHONPATH* Variable:

- The PYTHONPATH is an environment variable, consisting of a list of directories.
- The syntax of PYTHONPATH is the same as that of the shell variable PATH.
- Here is a typical PYTHONPATH from a Windows system:
  - set PYTHONPATH=C:\Users\user;
- Here is a typical PYTHONPATH from a UNIX system:
  - set PYTHONPATH=/usr/local/lib/python
- Ubuntu
  - /usr/lib/python3.8



Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Preferences Ctrl+Alt+Shift+P
PYTHONPATH manager
Current user environment variables...
Reset Spyder to factory defaults

C:\Users\user

C:\Users\user\.spyder-py3\temp.py

temp.py

```
123
124     def hello(self):
125         print("hello", self.i)
126
127     @staticmethod
128     def statictest():
129         print("this is static method..")
130
131
132 #    def statictest():
```

Source

PYTHONPATH manager

Move to top | Move up | Move down | Move to bottom

Add path | Remove path | Synchronize... | OK | Cancel

Select directory

user

搜尋 user

組合管理 新增資料夾

快速存取
桌面
下載
文件
圖片
Python (大)
相關檔案
美味大挑戰Vol_
美味大挑戰Vol_
Dropbox
OneDrive
本機
3D 物件

.conda .ipython .jupyter .LINE .matplotlib .QtWebEngineProcess .spyder-py3 __pycache__ 3D 物件 AppData
Dropbox MicrosoftEdgeBackups OneDrive twca twcabill 下載 文件 我的最愛 音樂 桌面
連結 連絡人 搜尋 圖片 影片 儲存的遊戲

資料夾:

英

選擇資料夾 取消

Namespaces and Scoping

- Variables are names (identifiers) that map to objects.
- **A *namespace* is a dictionary of variable names (keys) and their corresponding objects (values)**.
- A Python statement can access variables in a *local namespace* and in the *global namespace*.
  - If a local and a global variables have the same name, the local variable ***shadows*** the global variable.
- Each function has its own local namespace.
  - Class methods follow the same scoping rule as ordinary functions.
- Python assumes that any variables assigns a value in a function is the local.

Namespaces and Scoping

- In order to assign a value to a global variable within a function, you must first use the *global* statement.
- The statement *global VarName* tells Python that VarName is a global variable.
  - Python stops searching the local namespace for the variable.

```python
Money = 2000
def AddMoney():
    Money = Money + 1

print(Money)
AddMoney()
print(Money)
```

```
In [1]: runfile('C:/Users/user/untitled5.py', wdir='C:/Users/user')
2000
Traceback (most recent call last):

  File "C:\Users\user\untitled5.py", line 13, in <module>
    AddMoney()

  File "C:\Users\user\untitled5.py", line 10, in AddMoney
    Money = Money + 1

UnboundLocalError: local variable 'Money' referenced before assignment
```

Example (1)

```
# sample.py
myGlobal = 5

def func1():
    myGlobal = 42

def func2():
    print myGlobal

func1()
func2()
```

⇒ 5

```
def func1():
    global myGlobal
    myGlobal = 42
```

The global statement is a declaration which holds for the entire current code block.

⇒ 42

Example (2)

```python
x = 10
def outer():
    x = 100          # 這是在 outer() 函式範圍的 x
    def inner():
        nonlocal x
        x = 1000     # 改變的是 outer() 函式的 x
    inner()
    print(x)         # 顯示 1000

outer()
print(x)             # 顯示 10
```

The nonlocal statement causes the listed identifiers to refer to previously bound variables in the nearest enclosing scope excluding globals. (Python 3)

```
In [12]: runfile('C:/Users/user/untitled5.py', wdir='C:/Users/user')
1000
10
```

Example (3)

1

```python
a = 0

def function1():
    a = 1
    def function2():
        a = 2
        print("function2: ", a)
    function2()
    print("function1: ", a)

function1()
print("global: ", a)
```

```
function2:  2
function1:  1
global:  0
```

2

```python
a = 0

def function3():
    a = 1
    def function4():
        nonlocal a
        a = 2
        print("function4: ", a)
    function4()
    print("function3: ", a)

function3()
print("global: ", a)
```

```
function4:  2
function3:  2
global:  0
```

3

```python
a = 0

def function5():
    a = 1
    def function6():
        global a
        a = 2
        print("function6: ", a)
    function6()
    print("function5: ", a)

function5()
print("global: ", a)
```

```
function6:  2
function5:  1
global:  2
```

Example (2)

The nonlocal statement causes the listed identifiers to refer to previously bound variables in the nearest enclosing scope excluding globals. (Python 3)

The global statement is a declaration which holds for the entire current code block.

```
def scope_test():
    def do_local():
        spam = "local spam"

    def do_nonlocal():
        nonlocal spam
        spam = "nonlocal spam"

    def do_global():
        global spam
        spam = "global spam"

    spam = "test spam"
    do_local()
    print("After local assignment:", spam)
    do_nonlocal()
    print("After nonlocal assignment:", spam)
    do_global()
    print("After global assignment:", spam)

scope_test()
print("In global scope:", spam)
```

```
After local assignment: test spam
After nonlocal assignment: nonlocal s
After global assignment: nonlocal spa
In global scope: global spam
```

globals() locals(), and var()

- The *globals() locals()* and *var()* functions can be used to return the names in the global and local namespaces depending on the location from where they are called.
- If locals() is called from within a function, it will return all the names that can be accessed **locally** from that function.
- If globals() is called from within a function, it will return all the names that can be accessed **globally** from that function.
- var() returns *either a* dictionary of the current namespace (if called with no argument) or *the* dictionary of the argument.
- The return type of both these functions (i.e., locals and globals) is dictionary.
  - names can be extracted using the keys() function.

```python
class A():
    def __init__(self, id):
        self.id = id
        print("Class A locals:\t%s" % locals())
        print("Class A vars:\t%s" % vars())


def B():
    id = 1
    print("Function B locals:\t%s" % locals())
    print("Function B vars:\t%s" % vars())


if __name__ == '__main__':
    a = A(1)

    B()
    print("Module globals:\t%s\n" % globals())
    print("Module locals:\t%s\n" % locals())
    print("Module vars:\t%s\n" % vars())
```

```
Class A locals: {'self': <__main__.A object at 0x000002063392C3C8>, 'id': 1}
Class A vars:   {'self': <__main__.A object at 0x000002063392C3C8>, 'id': 1}
Function B locals:      {'id': 1}
Function B vars:        {'id': 1}
Module globals: {'__name__': '__main__', '__file__': 'C:\\Users\\user\\untitled0.py', '__nonzero__': <function
InteractiveShell.new_main_mod.<locals>.<lambda> at 0x00000206334EE798>, '__builtins__': {'__name__': 'builtins',
'__doc__': "Built-in functions, exceptions, and other objects.\n\nNoteworthy: None is the `nil' object; Ellipsis
represents `...' in slices.", '__package__': '', '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__sp
ModuleSpec(name='builtins', loader=<class '_frozen_importlib.BuiltinImporter'>), '__build_class__': <built-in fun
__build_class__>, '__import__': <built-in function __import__>, 'abs': <built-in function abs>, 'all': <built-in
function all>, 'any': <built-in function any>, 'ascii': <built-in function ascii>, 'bin': <built-in function bin>
'breakpoint': <built-in function breakpoint>, 'callable': <built-in function callable>, 'chr': <built-in function
'compile': <built-in function compile>, 'delattr': <built-in function delattr>, 'dir': <built-in function dir>,
'divmod': <built-in function divmod>, 'eval': <built-in function eval>, 'exec': <built-in function exec>, 'format
<built-in function format>, 'getattr': <built-in function getattr>, 'globals': <built-in function globals>, 'hasa
<built-in function hasattr>, 'hash': <built-in function hash>, 'hex': <built-in function hex>, 'id': <built-in fu
id>, 'input': <bound method Kernel.raw_input of <spyder_kernels.console.kernel.SpyderKernel object at
0x00000206334E5DC8>>, 'isinstance': <built-in function isinstance>, 'issubclass': <built-in function issubclass>,
```

The dir( ) Function

- The dir() built-in function returns a sorted list of strings containing the names defined by a module.
- The list contains the names of all the modules, variables and functions that are defined in a module.
- The special string variable *__name__* is the **module's name**, and *__file__* is the **filename** from which the module was loaded.

```
In [25]: import math

In [26]: content =dir(math)

In [27]: print(content)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'fl
'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma
'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'tau', 'trunc']
```

Packages in Python

- A package is a hierarchical file directory structure
  - It consists of modules and subpackages and sub-subpackages, and so on.
- Consider a file *Pots.py* available in *Phone* directory
- Another two files have different functions with the same directory as above:
  - *Phone/Isdn.py* file having function Isdn()
  - *Phone/G3.py* file having function G3()
- Now, create one more file __init__.py in *Phone* directory:
  - Phone/__init__.py
- *Phone* directory includes *Pots.py, Isdn.py, G3.py, and __init__.py.*

Packages in Python

- When you've imported Phone, you need to put explicit import statements in __init__.py as follows:
  - from Pots import Pots
  - from Isdn import Isdn
  - from G3 import G3

```
#!/usr/bin/python

# Now import your Phone Package.
import Phone

Phone.Pots()
Phone.Isdn()
Phone.G3()
```

```
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
```

Example

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
import sound.effects.echo
```

```
sound.effects.echo.echofilter(input, output, delay=0.7, atten=4)
```

```
from sound.effects import echo
```

```
echo.echofilter(input, output, delay=0.7, atten=4)
```

```
from sound.effects.echo import echofilter
```

```
echofilter(input, output, delay=0.7, atten=4)
```

Tic Tac Toe Game (Console)

```python
import os,random

def print_board(board):          # 輸出井字的樣子
    print(board['1'] + ' | ' + board['2'] + ' | ' + board['3'])
    print('--+---+--')
    print(board['4'] + ' | ' + board['5'] + ' | ' + board['6'])
    print('--+---+--')
    print(board['7'] + ' | ' + board['8'] + ' | ' + board['9'])
```

```
  |   |
--+---+--
  |   |
--+---+--
  |   |
```

Tic Tac Toe Game (Console)

```python
def main():
    init_board = {
        '1': ' ', '2': ' ', '3': ' ',
        '4': ' ', '5': ' ', '6': ' ',
        '7': ' ', '8': ' ', '9': ' ' }
    begin = True
    while begin:
        curr_board = init_board.copy()  # 複製初始的空字典
        begin = False
        turn = random.choice(['x','o']) # 隨機讓兩者開始
        counter = 0
        print_board(curr_board)
        while counter < 9:
            move = input('輪到 %s ,請輸入位置: ' % turn)
            if curr_board[move] == ' ': # 判斷位置是否為' ',空則記錄現在的 turn
                counter += 1
                curr_board[move] = turn
                if turn == 'x':
                    turn = 'o'
                else:
                    turn = 'x'
            print_board(curr_board)
        choice = input('再玩一局 ? (yes | no) ')
        begin = choice == 'yes'

if __name__ == '__main__':
    main()
```

```
輪到 o ,請輸入位置: 5
  |   |
--+---+--
  | o |
--+---+--
  |   |

輪到 x ,請輸入位置: 9
  |   |
--+---+--
  | o |
--+---+--
  |   | x

輪到 o ,請輸入位置:
```

Python Image Library - Examples

Python Imaging Library

```
from PIL import Image
global ext
ext = ".jpg"
imageFile = "test.jpg"
im1 = Image.open(imageFile)
im1.show()
```

Original image

- BMP
- EPS
- GIF
- JPEG
- PNG
- TIFF
- PDF

- **Python Imaging Library (PIL)**
- http://www.pythonware.com/products/pil/
- **PIL 1.1.7**
- http://effbot.org/downloads/PIL-1.1.7.win32-py2.7.exe

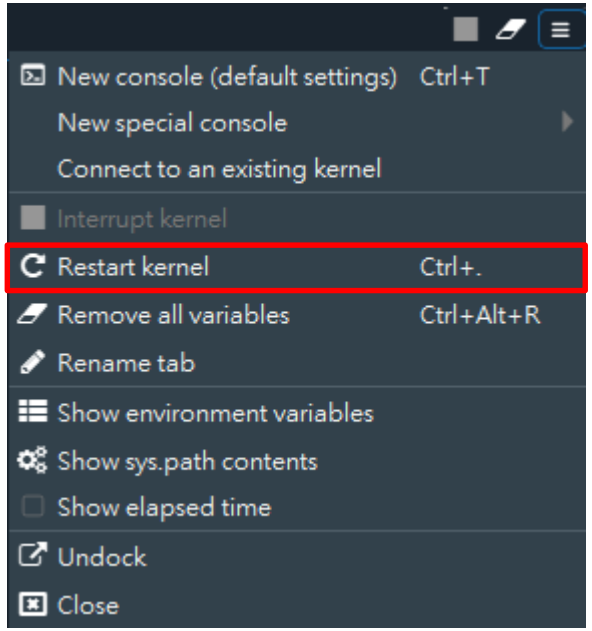
conda list

```
In [2]: conda list
# packages in environment at C:\Users\user\anaconda3:
#
# Name                    Version                   Build  Channel
_anaconda_depends         2020.02                  py37_0

Note: you may need to restart the kernel to use updated packages.
_ipyw_jlab_nb_ext_conf    0.1.0                    py37_0
alabaster                 0.7.12                   py37_0
anaconda                  custom                   py37_1
anaconda-client           1.7.2                    py37_0
anaconda-navigator        1.9.12                   py37_0
anaconda-project          0.8.4                      py_0
argh                      0.26.2                   py37_0
asn1crypto
astroid
astropy
atomicwrites
attrs
autopep8
babel
backcall
backports
```

conda **install** package_name

```
In [2]: conda install Image
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying with flexible solve.

Note: you may need to restart the kernel to use updated packages.

PackagesNotFoundError: The following packages are not available from current channels:

  - image

Current channels:

  - https://repo.anaconda.com/pkgs/main/win-64
  - https://repo.anaconda.com/pkgs/main/noarch
  - https://repo.anaconda.com/pkgs/r/win-64
  - https://repo.anaconda.com/pkgs/r/noarch
  - https://repo.anaconda.com/pkgs/msys2/win-64
  - https://repo.anaconda.com/pkgs/msys2/noarch

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org

and use the search bar at the top of the page.
```

```
conda upgrade --all
```




```
In [1]: conda upgrade --all
Collecting package metadata (current_repodata.json): ...working... done
Note: you may need to restart the kernel to use updated packages.
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\user\anaconda3


The following packages will be downloaded:


    package                    |            build
    ---------------------------|-----------------
    dask-2.15.0                |             py_0          14 KB
    dask-core-2.15.0           |             py_0         575 KB
    distributed-2.15.0         |           py37_0         997 KB
    ------------------------------------------------------------
                                           Total:         1.5 MB

The following packages will be UPDATED:

  dask                                       2.14.0-py_0 --> 2.15.0-py_0
  dask-core                                  2.14.0-py_0 --> 2.15.0-py_0
  distributed                              2.14.0-py37_0 --> 2.15.0-py37_0



Downloading and Extracting Packages

dask-2.15.0          | 14 KB     |            |   0%
dask-2.15.0          | 14 KB     | ########## | 100%

distributed-2.15.0   | 997 KB    |            |   0%
distributed-2.15.0   | 997 KB    | #######8   |  79%
distributed-2.15.0   | 997 KB    | ########## | 100%

dask-core-2.15.0     | 575 KB    |            |   0%
dask-core-2.15.0     | 575 KB    | ########## | 100%
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done
```

conda commands

- conda --version : 檢視 conda 版本
- conda update PACKAGE_NAME :更新指定套件
  - conda install 'numpy>=1.15,<1.16'
- conda remove PACKAGE_NAME :在目前的工作環境移除指定套件
- conda list :檢視指定工作環境安裝的套件清單
- conda --help :指令說明文

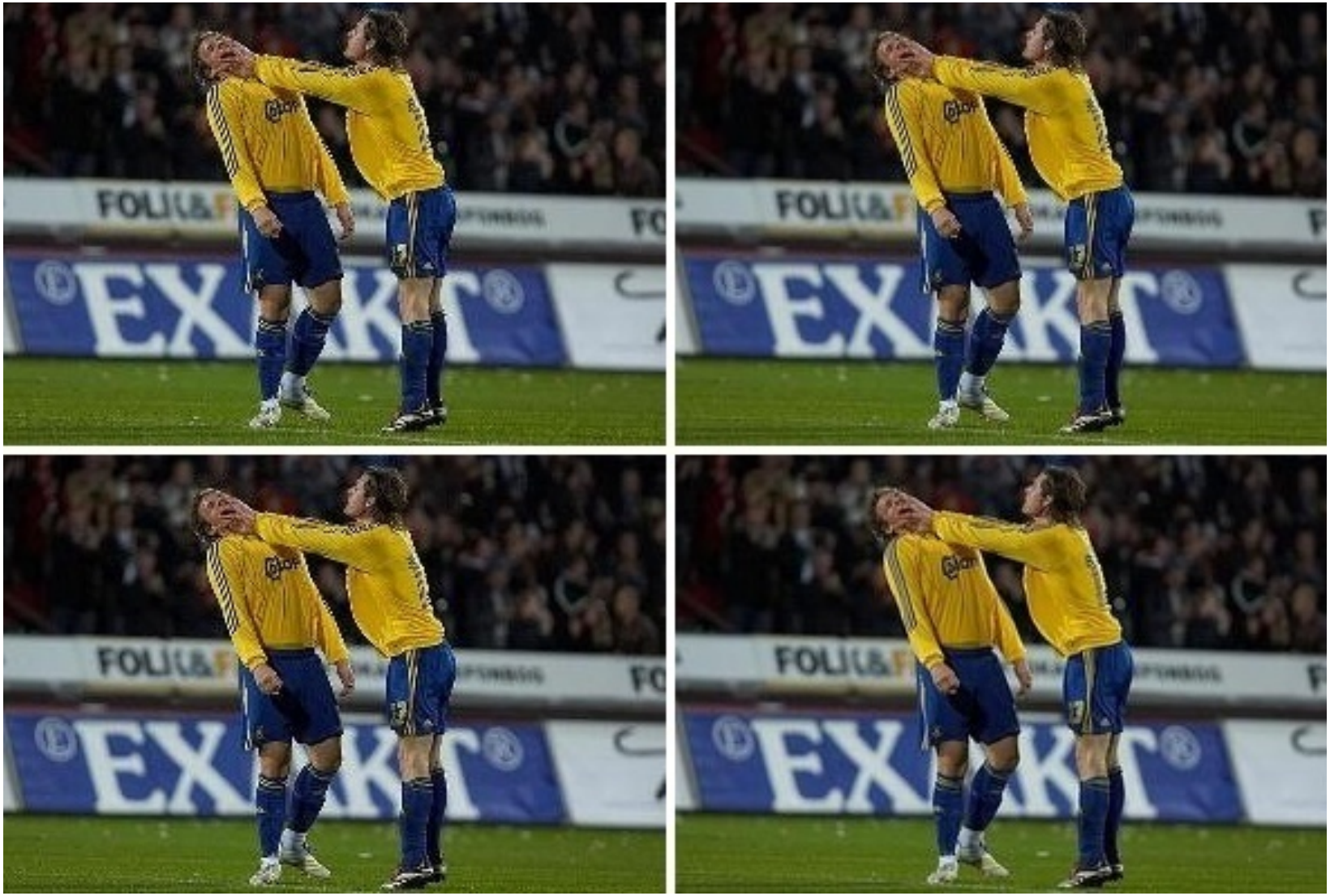
Resize

```
from PIL import Image
global ext
ext = ".jpg"
imageFile = "test.jpg"
im1 = Image.open(imageFile)



def imgResize(im):
     div = 2
     width = int(im.size[0] / div)
     height = int(im.size[1] / div)
     im2 = im.resize((width, height), Image.NEAREST) # use nearest neighbour
     im3 = im.resize((width, height), Image.BILINEAR) # linear interpolation in a 2x2 environment
     im4 = im.resize((width, height), Image.BICUBIC) # cubic spline interpolation in a 4x4 environme
     im5 = im.resize((width, height), Image.ANTIALIAS) # best down-sizing filter

     im2.save("NEAREST" + ext)
     im3.save("BILINEAR" + ext)
     im4.save("BICUBIC" + ext)
     im5.save("ANTIALIAS" + ext)
imgResize(im1)
```

im.resize((width, height), Image.NEAREST)

NEAREST：最近濾波。從輸入影象中選取最近的畫素作為輸出畫素。它忽略了所有其他的畫素。
BILINEAR：雙線性濾波。在輸入影象的2x2矩陣上進行線性插值。
BICUBIC：雙立方濾波。在輸入影象的4x4矩陣上進行立方插值。
ANTIALIAS：平滑濾波。這是PIL 1.1.3版本中新的濾波器。對所有可以影響輸出畫素的輸入畫素進行高質量的重取樣濾波，以計算輸出畫

Resize



Crop

```
def imgCrop(im):
    box = (50, 50, 200, 300)
    region = im.crop(box)
    region.save("CROPPED" + ext)

imgCrop(im1)
```

Transpose

```
def imgTranspose(im):
  box = (50, 50, 200, 300)
  region = im.crop(box)
  region =region.transpose(Image.ROTATE_180)
  im.paste(region, box)
  im.save("TRANSPOSE"+ext)

imgTranspose(im1)
```

- Image.FLIP_LEFT_RIGHT (左右翻轉)
- Image.FLIP_TOP_DOWN (上下翻轉)
- Image.ROTATE_90 (旋轉90度)
- Image.ROTATE_180 (旋轉180度)
- Image.ROTATE_270 (旋轉270度)

Band Merge

```
def bandMerge(im):
    r, g, b = im.split()
    im = Image.merge("RGB", (g,g,g))
    im.save("MERGE" + ext)

bandMerge(im1)
```

```
from PIL import Image
if __name__ == '__main__':
    im = Image.open('test.jpg')
    r,g,b = im.split()
    b.show()
    imx = Image.merge("RGB", (g, b, r))
    imx.show()
```

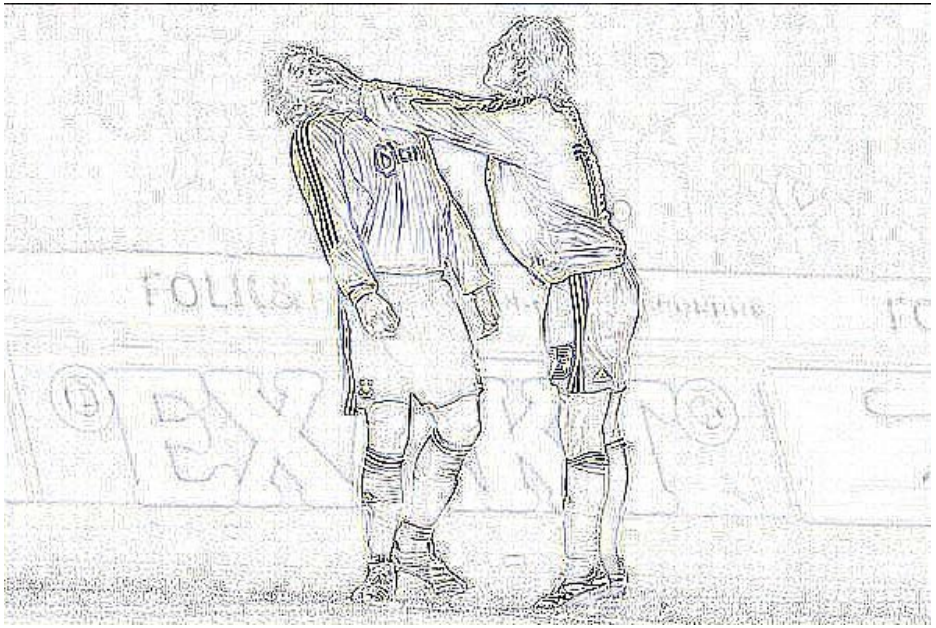
Blur

```
from PIL import ImageFilter
def filterBlur(im):
    im1 = im.filter(ImageFilter.BLUR)
    im1.save("BLUR" + ext)

filterBlur(im1)
```

加入濾鏡

Find Contours

```
from PIL import ImageFilter
def filterContour(im):
    im1 = im.filter(ImageFilter.CONTOUR)
    im1.save("CONTOUR" + ext)

filterContour(im1)
```



Find Edges

```
from PIL import ImageFilter
def filterFindEdges(im):
    im1 = im.filter(ImageFilter.FIND_EDGES)
    im1.save("EDGES" + ext)

filterFindEdges(im1)
```

ImageFilter.BLUR ：模糊濾鏡
ImageFilter.CONTOUR ：只顯示輪廓
ImageFilter.EDGE_ENHANCE ：邊界加強
ImageFilter.EDGE_ENHANCE_MORE ：邊界加強(閥值更大)
ImageFilter.EMBOSS ：浮雕濾鏡
ImageFilter.FIND_EDGES ：邊界濾鏡
ImageFilter.SMOOTH ：平滑濾鏡
ImageFilter.SMOOTH_MORE ：平滑濾鏡(閥值更大)
ImageFilter.SHARPEN ：銳化濾鏡

Python

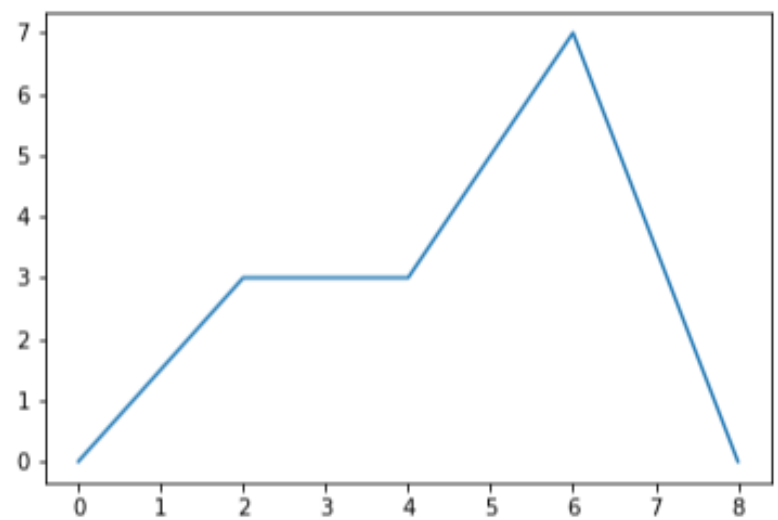
```
from PIL import Image,ImageDraw,ImageFont
if __name__ == '__main__':
    im = Image.open('test.jpg')
    dr_im = ImageDraw.Draw(im)
    w,h = im.size
    myFont = ImageFont.truetype('timesbd.ttf',80)
    dr_im.text([0.01*w, 0.5*h], "Python", fill = (255,0,0), font=myFont)
    im.show()
```

```python
from PIL import Image
def deffun(c):
    return c*3

if __name__ == '__main__':
    im = Image.open("test.jpg")
    im = Image.eval(im, deffun)
    im.show()
```

pyplot

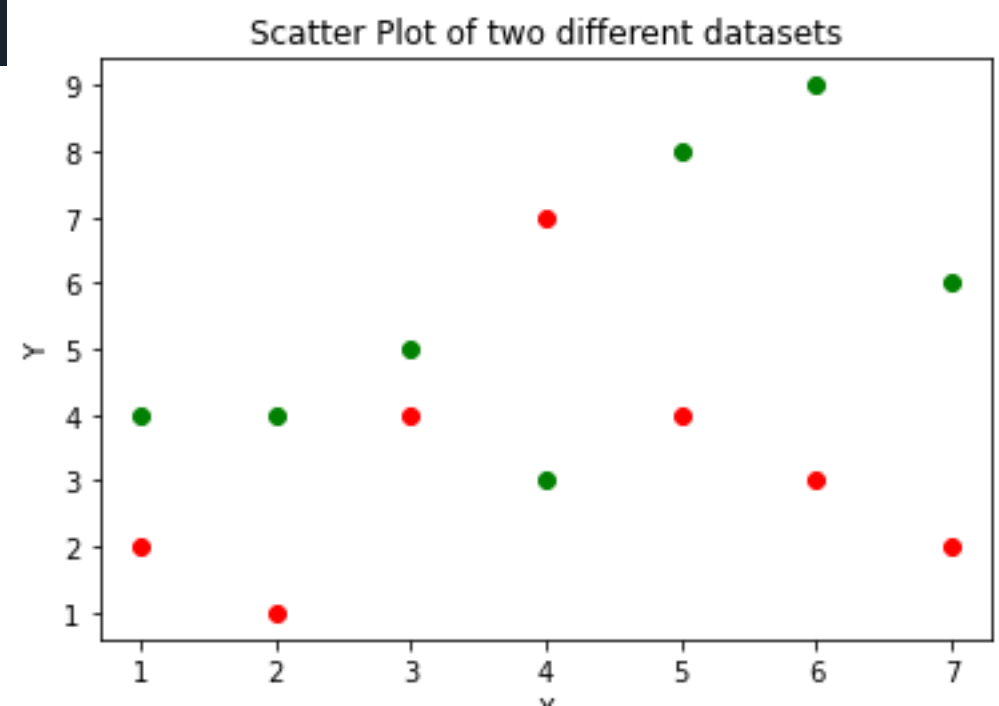
- matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc..

```
import matplotlib.pyplot as pyp

x = [0,2,4,6,8]

y = [0,3,3,7,0]

pyp.plot(x,y)

pyp.savefig("1.png")
```



pyplot

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5,6,7]
y1=[2,1,4,7,4,3,2]
y2=[4,4,5,3,8,9,6]

plt.scatter(x,y1,c="red")
plt.scatter(x,y2,c="green")
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Scatter Plot of two different datasets")
plt.show()
```



NumPy

- NumPy is the fundamental package for scientific computing in Python.

- It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C `long`; normally either `int64` or `int32`)
intc	Identical to C `int` (normally `int32` or `int64`)
intp	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for `float64`.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for `complex128`.
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

種類	代替字串	說明
bool	?	boolean 布林值。
int	i	(signed) integer 帶有符號的整數（正負整數），等同 int64。
int8	i1	整數 -128 ~ 127。
int16	i2	整數 -32768 ~ 32767。
int32	i4	整數 -2147483648 ~ 2147483647。
int64	i8	整數 -9223372036854775808 ~ 9223372036854775807。
uint	u	unsigned integer 無有符號的整數（正整數），等同 uint64。
uint8	u1	正整數 0 ~ 255。
uint16	u2	正整數 0 ~ 65535。
uint32	u4	正整數 0 ~ 4294967295。
uint64	u8	正整數 0 ~ 18446744073709551615。

float	f	floating-point 浮點數，等同 float64。
float16	f2	半精度浮點數。
float32	f4	單精度浮點數。
float64	f8	雙精度浮點數。
complex	c	complex floating-point 浮點類型複數，等同 complex128。
complex64	c8	雙 32 位複數。
complex128	c16	雙 64 位複數。
object	O	object 物件。
byte	b	(signed) byte 帶有符號的位元。
ubyte	B	unsigned byte 無符號的位元。
unicode	U	Unicode。
	S	(byte-)string 字串。
	m	timedelta 時間間隔。
	M	datetime 日期時間。
	V	void 原始數據。

NumPy

```python
import numpy as np
a = np.array([1.1, 2.2, 3.3, 0, -1])
b = np.array(a, dtype='?')
c = np.array(a, dtype='U')
d = np.array(a, dtype='S')
e = np.array(a, dtype='B')
f = np.array(a, dtype='i')
print(a)   # [ 1.1  2.2  3.3  0.  -1. ]
print(b)   # [ True  True  True False  True]
print(c)   # ['1.1' '2.2' '3.3' '0.0' '-1.0']
print(d)   # [b'1.1' b'2.2' b'3.3' b'0.0' b'-1.0']
print(e)   # [  1   2   3   0 255]
print(f)   # [ 1  2  3  0 -1]
```

```
[ 1.1  2.2  3.3  0.  -1. ]
[ True  True  True False  True]
['1.1' '2.2' '3.3' '0.0' '-1.0']
[b'1.1' b'2.2' b'3.3' b'0.0' b'-
[  1   2   3   0 255]
[ 1  2  3  0 -1]
```

```python
import numpy as np
a = np.array([1, 2, 3, 4], dtype="int32")
b = a.astype('float32')
print(a)    # 1, 2, 3, 4
print(b)    # 1., 2., 3., 4,
```

```
[1 2 3 4]
[1. 2. 3. 4.]
```

```python
import numpy as np
dt = np.dtype([('a','U5'),('b','f'),('c','?')])
a = np.array([(1.1,2.2,3.3), (1.1,2.2,3.3)], dtype=dt)
print(a)    # [('1.1', 2.2,  True) ('1.1', 2.2,  True)]
```

```
[('1.1', 2.2,  True) ('1.1', 2.2,  True)]
```

```python
import numpy as np
a = np.array([1,2,3,4,5])
b = np.array([1,2,3,4,5], dtype='U10')
print(a.dtype)    # int64
print(b.dtype)    # <U10
```

```
int32
<U10
```

方法	適用維度	說明
numpy.array()	多維	根據現有的串列資料建立陣列。
numpy.empty()	多維	建立指定大小的空陣列。
numpy.zeros()	多維	建立每個項目數值為 0 的陣列。
numpy.ones()	多維	建立每個項目數值為 1 的陣列。
numpy.eye()	多維	建立對角線項目為 1，其他項目為 0 的陣列。
numpy.tile()	多維	複製現有的陣列的內容，依據新的維度建立新陣列。
numpy.arange()	一維	建立兩個數值間，指定「間距」的「等差」連續資料的陣列。
numpy.linspace()	一維	建立兩個數值間，指定「數量」的「等差」連續資料陣列。
numpy.logspace()	一維	建立兩個數值間，指定「數量」的「log 對數」連續資料陣列。

```python
import numpy as np

a = np.array([1,2,3,4])                          # 建立一維陣列
b = np.array([[1,2],[3,4]])                      # 建立二維陣列
c = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])      # 建立三維陣列
print("array a:\n", a)
print("array b:\n", b)
print("array c:\n", c)
```

```
array a:
 [1 2 3 4]
array b:
 [[1 2]
 [3 4]]
array c:
 [[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

```python
import numpy as np

a = np.empty(2)          # 建立一維
b = np.empty([2,2])      # 建立二維
c = np.empty([2,2,2])    # 建立三維
print("array a:\n", a)
print("array b:\n", b)
print("array c:\n", c)
```

```
array a:
 [4.24399158e-314 8.48798317e-314]
array b:
 [[4.24399158e-314 8.48798317e-314]
 [1.27319747e-313 1.69759663e-313]]
array c:
 [[[4.94065646e-324 2.12199579e-314]
  [4.24399158e-314 2.12199579e-314]]

 [[6.36598737e-314 4.24399158e-314]
  [2.12199579e-314 4.94065646e-324]]]
```

```python
import numpy as np

a = np.zeros(2)          # 建立一維 zero 陣列
b = np.zeros([2,2])      # 建立二維 zero 陣列
c = np.zeros([2,2,2])    # 建立三維 zero 陣列
print("array a:\n", a)
print("array b:\n", b)
print("array c:\n", c)
```

```
array a:
 [0. 0.]
array b:
 [[0. 0.]
 [0. 0.]]
array c:
 [[[0. 0.]
  [0. 0.]]

 [[0. 0.]
  [0. 0.]]]
```

```python
import numpy as np

a = np.arange(10)          # 建立 1～10，間隔為 1 的陣列
b = np.arange(5,10)        # 建立 5
c = np.arange(5,10,0.1)    # 建立 5
d = np.arange(5,10,dtype='float')
print("array a:\n", a)
print("array b:\n", b)
print("array c:\n", c)
print("array d:\n", d)
```

```
array a:
 [0 1 2 3 4 5 6 7 8 9]
array b:
 [5 6 7 8 9]
array c:
 [5.  5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.  6
 6.8 6.9 7.  7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9
 8.6 8.7 8.8 8.9 9.  9.1 9.2 9.3 9.4 9.5 9.6 9.7
array d:
 [5. 6. 7. 8. 9.]
```

narray[起始索引 : 終止索引 [:間隔值]]

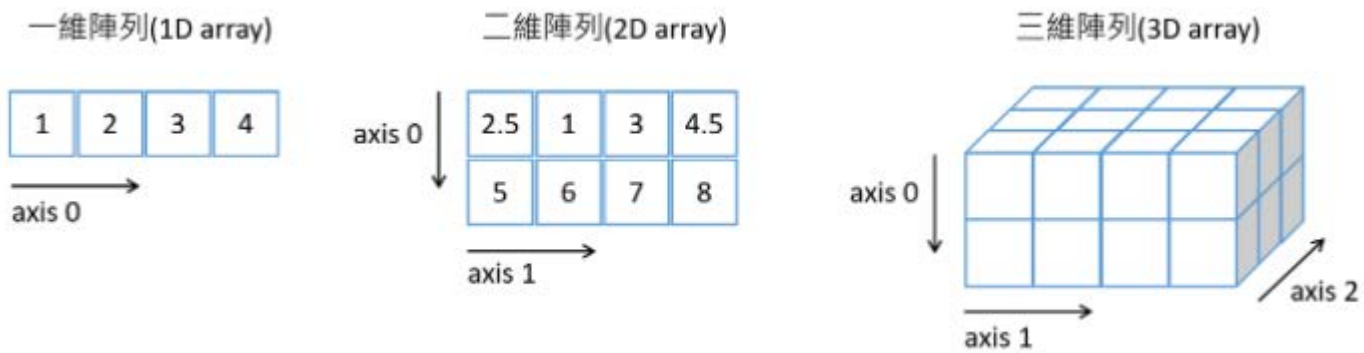
NumPy

```python
import numpy as np

# 陣列元素可以使用 list 或 tuple 傳入
array1 = np.array([1, 2, 3])

print(array1)
```

```
In [12]: runfile('C:/Users/user/untitled7.py', wdir='C:/U
[1 2 3]
```

```python
import numpy as np

# 陣列元素可以使用 list 或 tuple 傳入
array1 = np.array([1, 2, 3])

print(array1[0])
# 取 1 到 3（不含）元素
print(array1[1:3])
# 透過 mask 布林遮罩可以決定要取出哪些符合條件的元素陣列
# 符合的只有 index 2，元素為 3 的值
mask = (array1 % 3 == 0)
print(array1[mask])
```

```
In [14]: runfile('C:/Users/user/numpy2.py', wdir='C:/Users/user')
1
[2 3]
[3]
```



```python
import numpy as np
a = np.array([1,2,3,4,5,6,7,8,9,10])
print(a[a>5])                # [ 6  7  8  9 10] 篩選數值大於
print(a[a%2==0])             # [ 2  4  6  8 10] 篩選出數值為
print(a[(a>5) & (a<10)])     # [ 6  7  8  9] 篩選出數
```

```
[ 6  7  8  9 10]
[ 2  4  6  8 10]
[6 7 8 9]
```

```python
import numpy as np

a = np.array([1,2,3,4,5,6,7,8,9])
print(a[2:5])    # [3 4 5]                 從第三項取值到
print(a[:5])     # [1 2 3 4 5]             從第一項取值到
print(a[2:])     # [3 4 5 6 7 8 9]         從第三項開始取
print(a[2:-1])   # [3 4 5 6 7 8]           從第三項取值到
print(a[::2])    # [1 3 5 7 9]             間隔 2 取值
print(a[2:8:2])  # [3 5 7]                 第三項到第七項
print(a[::-1])   # [9 8 7 6 5 4 3 2 1]     反轉
print(a[[1,3,5,7]])   # [2 4 6 8]          取出第二、第四、第
```

```
[3 4 5]
[1 2 3 4 5]
[3 4 5 6 7 8 9]
[3 4 5 6 7 8]
[1 3 5 7 9]
[3 5 7]
[9 8 7 6 5 4 3 2 1]
[2 4 6 8]
```

方法	說明
單純讀取	使用 Python 的迴圈進行讀取。
numpy.nditer()	將陣列轉換成 nditer 物件，快速迭代多維陣列內的所有元素。
numpy.ndenumerate()	回傳迭代陣列時，該元素的索引值與內容。

```python
import numpy as np

a = np.array([1,2,3,4,5,6])
for i in a:
    print(i, end=' ')      # 1 2 3 4 5 6
print("\n")

b = np.array([[1,2,3],[4,5,6]])
for i in b:
    print(i, end=' ')      # [1 2 3] [4 5 6]
print("\n")

for i in b:
    for j in i:
        print(j, end=' ')    # 1 2 3 4 5 6
```

```
1 2 3 4 5 6

[1 2 3] [4 5 6]

1 2 3 4 5 6
```

```python
import numpy as np

b = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
for i in b:
    for j in i:
        for k in j:
            print(k, end=' ')    # 1 2 3 4 5 6 7 8 使用三次 for
print()
for i in np.nditer(b):
    print(i, end= ' ')           # 1 2 3 4 5 6 7 8 只使用一次 f
```

```
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
```

```python
import numpy as np

a = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
for i in np.nditer(a[...,::2], flags=['external_loop']):
    print(i, end= ' ')    # [1 3] [4 6] [7 9] [10 12]
```

```
[1 3] [4 6] [7 9] [10 1
```

external_loop causes the values given to be one-dimensional arrays with multiple values instead of zero-dimensional arrays.

```python
import numpy as np

a = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
for i,j in np.ndenumerate(a):
    print(i,j)
```

```
(0, 0, 0) 1
(0, 0, 1) 2
(0, 0, 2) 3
(0, 1, 0) 4
(0, 1, 1) 5
(0, 1, 2) 6
(1, 0, 0) 7
(1, 0, 1) 8
(1, 0, 2) 9
(1, 1, 0) 10
(1, 1, 1) 11
(1, 1, 2) 12
```

```python
import numpy as np

a = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
for i in np.nditer(a[...,::2], order='C'):
    print(i, end= ' ')    # 1 3 4 6 7 9 10 12
print()
for i in np.nditer(a[...,::2], order='F'):
    print(i, end= ' ')    #  7 4 10 3 9 6 12
```

```
1 3 4 6 7 9 10 12
1 7 4 10 3 9 6 12
```

方法	說明
reshape()	改變陣列形狀。
flatten()、numpy.ravel()	扁平化陣列。
numpy.transpose()、T	互換維度。
numpy.rollaxis()、numpy.swapaxes()	根據指定「軸」，將陣列項目「滾動」或「交換」位置。

```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(4, 3)
print(newarr)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```python
import numpy as np

a = np.array([1,2,3,4,5,6,7,8])
b = a.reshape((4,2))
print("b is\n", b)
'''
[[1 2]
[3 4]
[5 6]
[7 8]]
'''
c = a.reshape((2,4))
print("c is\n", c)
'''
[[1 2 3 4]
[5 6 7 8]]
'''
```

```
b is
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]
c is
 [[1 2 3 4]
 [5 6 7 8]]
```

```python
import numpy as np

a = np.array([[1,2],[3,4],[5,6],[7,8]])
b = a.reshape(-1)          # 轉換成一維陣列
print("b is\n", b)                  # [1 2 3 4 5 6 7 8]
c = a.reshape((2,-1))      # 等同 a.reshape((2,4))
print("c is\n", c)
'''
[[1 2 3 4]
[5 6 7 8]]
[[[1 2]
  [3 4]]
'''
d = a.reshape((2,2,-1))  # 等同 a.reshape((2,2,2))
print("d is\n", d)
'''
[[5 6]
  [7 8]]]
'''
```

```
b is
 [1 2 3 4 5 6 7 8]
c is
 [[1 2 3 4]
 [5 6 7 8]]
d is
 [[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

```python
import numpy as np

a = np.array([[1,2],[3,4],[5,6],[7,8]])
b = a.flat[3]
print("b is", b)     # 4
```

```
b is 4
```

NumPy

```python
import numpy as np

# 陣列元素可以使用 list 或 tuple 傳入
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[7, 8, 9], [1, 2, 3]])

result1 = A + B

print(result1)
```

```
In [16]: runfile('C:/Users/user/numpy2.py', wdir='C:/Users/user')
[[ 8 10 12]
 [ 5  7  9]]
```

```python
import numpy as np

# 陣列元素可以使用 list 或 tuple 傳入
A = np.array([[1, 2, 3], [4, 5, 6]])

print(A.T)
```

```
In [17]: runfile('C:/Users/user/numpy2.py', wdir='C:/Users/user')
[[1 4]
 [2 5]
 [3 6]]
```

```python
import numpy as np

# 陣列元素可以使用 list 或 tuple 傳入
A = np.array([[1, 2, 3], [4, 5, 6]])
C = np.array([[7, 8, 9], [1, 2, 3], [1, 2, 3]])

# 注意：矩陣乘法需要 第一個欄數等於第二個矩陣列數
result1 = A.dot(C)

print(result1)
```

```
In [18]: runfile('C:/Users/user/numpy2.py', wdir='C:/Users/user')
[[12 18 24]
 [39 54 69]]
```

方法	參數	說明
numpy.where()	判斷式,arr,val	根據判斷式找出索引值，回傳索引值的陣列，如果有設定 arr 和 val 會根據 arr 和 val 回傳新的陣列。
numpy.nonzero()	arr	取得內容「非 0」的項目索引值，並將索引值回傳為為新陣列。
numpy.count_nonzero()	arr	計算陣列中「非 0」項目的數量。
numpy.extract()	條件 arr, 目標 arr	根據條件陣列中的 True 或 False 項目索引值，取出目標 arr 對應的項目為新陣列。

```
(array([1, 5], dtype=int64),)
```

```python
import numpy as np

a = np.array(['a','b','c','d','a','b','c'])
b = np.where(a=='b')
print(b)    # (array([1, 5]),)   b 位在 1 和 5 的位置
```

```python
import numpy as np

a = np.array([[1,2,3,4],[5,6,7,8]])
b = np.array([['a','a','a','a'],['b','b','b','b']])
c = np.where(a>3,b,0)
print(c)
'''
[['0' '0' '0' 'a']
 ['b' 'b' 'b' 'b']]
```

```
[['0' '0' '0' 'a']
 ['b' 'b' 'b' 'b']]
```

```
[['1' '1' '1' 'a']
 ['b' 'b' 'b' 'b']]
```

```python
import numpy as np

a = np.array([[1,0,1,0],[1,0,1,0]])
b = np.nonzero(a)
print(b)
# (array([0, 0, 1, 1]), array([0, 2, 0, 2]))
# 對應 (0, 0) (0, 2) (1, 0) (1, 2)
```

```
(array([0, 0, 1, 1], dtype=int64), array([0, 2, 0, 2], dtype=int64))
```

```python
import numpy as np

a = np.array([1,0,1,0,1,0,1,0])
b = np.count_nonzero(a)
print(b)   # 4
```

```
4
```

```python
import numpy as np

a = np.array([1,0,1,0,1,0,1,0])
b = np.array(['a','b','c','d','e','f','g','h'])
c = np.extract(a,b)        # 預設 1 為 True，0 為 False
print(c)                   # ['a' 'c' 'e' 'g']
d = np.extract(a==0,b)     # 加入判斷條件，0 變成 True
print(d)                   # ['b' 'd' 'f' 'h']
```

```
['a' 'c' 'e' 'g']
['b' 'd' 'f' 'h']
```

random 隨機函數	說明
rand()	根據給予的維度形狀，產生 0 ~ 1 之間的隨機浮點數資料(不包含 1)
randn()	根據給予的維度形狀，返回標準常態分佈的隨機浮點數資料
randint(最小值[, 最大值, size])	返回隨機整數，會依照所設定的最大最小值範圍區間， 返回所要求的隨機整數(包含最小值，不包含最大值)
random(size)	根據給予的維度形狀 size，產生 0 ~ 1 之間的隨機浮點數資料
random_sample(size)	根據給予的維度形狀 size，產生 0 ~ 1 之間的隨機浮點數資料
sample(size)	根據給予的維度形狀 size，產生 0 ~ 1 之間的隨機浮點數資料
ranf(size)	根據給予的維度形狀 size，產生 0 ~ 1 之間的隨機浮點數資料
choice(array, size[,replace=True])	從給予的一維陣列中，根據給予的維度形狀 size 返回隨機整數; 這邊參數 replace=True 代表會返回重複的資料

```python
from numpy import random

print(random.random())
print("\n")        # 0.4068905682640278
print(random.random(3))        # [0.61705999 0.102875
print("\n")
print(random.random((3,2)))    # 二維陣列
print("\n")
print(random.random((3,2,2)))  # 三維陣列
print("\n")
```

```
0.6350543718514433

[0.00680787 0.78241369 0.71

[[0.95749442 0.86366161]
 [0.57645279 0.52414018]
 [0.19322981 0.06558013]]

[[[0.05558463 0.89136838]
  [0.64655945 0.18446667]]

 [[0.13274655 0.93905385]
  [0.84668827 0.12338257]]

 [[0.98686637 0.77392801]
  [0.5068022  0.10591722]]]
```

```python
from numpy import random
a = [1,2,3,4,5,6,7,8,9]
b = random.permutation(a)
print("b is\n",b)     #  [8 2 7 5 6 1 3 4 9]

c = [[1,2,3],[4,5,6],[7,8,9]]
d = random.permutation(c)   # 只重排第一個維度的項目[
print("d is\n",d)     # [[7 8 9] [1 2 3] [4 5 6]]

e = [[[1,2],[3,3]],[[4,5],[6,6]],[[7,8],[9,9]]]
f = random.permutation(e)   # 只重排第一個維度的項目
print("f is\n",f)     # [[[4 5] [6 6]] [[1 2] [3 3]] [[7 8] [
```

```
b is
 [9 7 6 1 8 4 2 3 5]
d is
 [[7 8 9]
 [1 2 3]
 [4 5 6]]
f is
 [[[4 5]
  [6 6]]

 [[7 8]
```

```python
from numpy import random

a = [1,2,3,4,5,6,7,8]
b = [[1,2,3,4],[5,6,7,8]]
c = [[[1,2],[3,4]],[[5,6],[7,8]]]
random.shuffle(a)
random.shuffle(b)
random.shuffle(c)
print("a is \n", a)   # [6, 1, 7, 8, 3, 5, 4, 2]
print("b is \n", b)   # [[5, 6, 7, 8], [1, 2, 3, 4]]
            # 只重排第一個維度的項目
print("c is \n", c)   # [[[5, 6], [7, 8]], [[1, 2], [3, 4]]]
            # 只重排第一個維度的項目
```

```
a is
 [6, 4, 2, 3, 5, 8, 7, 1]
b is
 [[5, 6, 7, 8], [1, 2, 3, 4]]
c is
 [[[5, 6], [7, 8]], [[1, 2], [3, 4]]]
```

```python
from numpy import random
# 產生十個一個 0～10 隨機整數
print(random.choice(10))      # 8

# 產生十個 0～10 隨機整數
print(random.choice(10,10))   # [5 6 9 6 3 7 1 6 5 1]

# 產生十個不重複的 0～10 隨機整數
print(random.choice(10,10, replace=False))  # [3 2 6 9 8 5 1 7

# 根據機率產生十個 a、b、c、d 組合的隨機數陣列
print(random.choice(['a','b','c','d'],10, p=[0.1,0.8,0.1,0]))
```

```
7
[3 0 8 7 8 7 4 6 1 0]
[8 3 7 2 9 1 6 5 0 4]
['c' 'b' 'b' 'a' 'b' 'b' 'b' 'b' 'c' 'a']
```

Probability
[a,b,c,d]

生成演算法	中文名稱	參考
random.normal	常態分布、高斯分布	NumPy 官方文件、Wiki
random.standard_normal	標準常態分布	NumPy 官方文件
random.power	冪定律分布	NumPy 官方文件、Wiki
random.beta	貝它分布	NumPy 官方文件、Wiki
random.gamma	伽瑪分布	NumPy 官方文件、Wiki
random.binomial	二項式分布	NumPy 官方文件、Wiki
random.chisquare	卡方分布	NumPy 官方文件、Wiki
random.dirichlet	狄利克雷分布	NumPy 官方文件、Wiki
random.exponential	指數分布	NumPy 官方文件、Wiki
random.standard_exponential	標準指數分布	NumPy 官方文件

random.f	F-分布	NumPy 官方文件、Wiki
random.geometric	幾何分布	NumPy 官方文件、Wiki
random.gumbel	甘別分布	NumPy 官方文件、Wiki
random.hypergeometric	超幾何分布	NumPy 官方文件、Wiki
random.laplace	拉普拉斯分布	NumPy 官方文件、Wiki
random.logistic	邏輯分布	NumPy 官方文件、Wiki
random.lognormal	對數常態分布	NumPy 官方文件、Wiki
random.logseries	對數分布	NumPy 官方文件、Wiki
random.multinomial	多項分布	NumPy 官方文件、Wiki
random.multivariate_normal	多元常態分布	NumPy 官方文件、Wiki
random.negative_binomial	負二項式分布	NumPy 官方文件、Wiki
random.noncentral_chisquare	Noncentral chi-squared 分布	NumPy 官方文件、Wiki

random.noncentral_f	Noncentral F 分布	NumPy 官方文件、Wiki
random.pareto	柏拉圖分布	NumPy 官方文件、Wiki
random.poisson	卜瓦松分布	NumPy 官方文件、Wiki
random.rayleigh	瑞利分布	NumPy 官方文件、Wiki
random.standard_cauchy	標準柯西分布	NumPy 官方文件、Wiki
random.standard_t	司徒頓 t 分布	NumPy 官方文件、Wiki
random.triangular	三角形分布	NumPy 官方文件、Wiki
random.uniform	連續型均勻分布	NumPy 官方文件、Wiki
random.vonmises	von Mises 分布	NumPy 官方文件、Wiki
random.wald	逆高斯分布	NumPy 官方文件、Wiki
random.weibull	韋伯分布	NumPy 官方文件、Wiki
random.zipf	齊夫定律分布	NumPy 官方文件、Wiki

NumPy

```python
import numpy as np

print("產生印出 (4x2) 隨機資料:\n", np.random.rand(4,2), "\n")
print("產生印出 (4x2) 常態分布隨機資料:\n", np.random.randn(4,2), "\n")
print("產生印出 3個 0~1 之間的浮點數資料:\n", np.random.random(3), "\n")
print("產生印出 11~25，8 個隨機整數:\n", np.random.randint(11, 25, 8), "\n")
print("產生印出 7個 0~29 不重複的隨機整數:\n", np.random.choice(30, 7, replace=False), "\n")
```

```
產生印出 (4x2) 隨機資料:
 [[0.49804227 0.93421214]
 [0.37104652 0.49484279]
 [0.84975811 0.43122559]
 [0.92401737 0.77715325]]

產生印出 (4x2) 常態分布隨機資料:
 [[-0.18366335 -0.16072549]
 [ 0.7401085  -0.32052699]
 [ 1.22535927  0.21268666]
 [-0.23074766 -0.38511585]]

產生印出 3個 0~1 之間的浮點數資料:
 [0.22457291 0.76087862 0.6313101 ]

產生印出 11~25，8 個隨機整數:
 [15 15 24 23 15 11 16 13]

產生印出 7個 0~29 不重複的隨機整數:
 [28 18  2  9 13  3  0]
```

```python
import numpy as np
a1 = np.random.randint(1, 100, 20)
print(a1)
a1.sort()
print(a1)
print(np.unique(a1))
```

```
In [19]: runfile('C:/Users/user/numpy2.py', wdir='C:/Users/user')
[59 78 58 62 82 74 85 18  2 60 71 83 69 38 60 28 32  4 70 87]
[ 2  4 18 28 32 38 58 59 60 60 62 69 70 71 74 78 82 83 85 87]
[ 2  4 18 28 32 38 58 59 60 62 69 70 71 74 78 82 83 85 87]
```

NumPy

day.csv

	A	B	C	D	E
1	V1	V2	V3	V4	V5
2	23	132	77	37	63
3	45	64	34	103	119
4	94	96	152	39	32
5	55	132	53	11	68
6	67	53	78	29	22

```python
import numpy as np

fileValue = np.genfromtxt('day.csv', delimiter=',', skip_header=1)
print(fileValue) # 印出取得數值
print(fileValue.shape) # 印出形狀
```

```
[[ 23. 132.  77.  37.  63.]
 [ 45.  64.  34. 103. 119.]
 [ 94.  96. 152.  39.  32.]
 [ 55. 132.  53.  11.  68.]
 [ 67.  53.  78.  29.  22.]]
(5, 5)
```

```python
import numpy as np

fileValue = np.genfromtxt('day.csv', delimiter=',', skip_header=1)

print("印出排序前的陣列:\n", fileValue, "\n") # 印出取得數值
print("印出排序後的陣列:\n", np.sort(fileValue), "\n") # 印出排序後的陣列
print("印出排序後的索引:\n", np.argsort(fileValue)) # 印出排序後的索引
```

```
印出排序前的陣列:
 [[ 23. 132.  77.  37.  63.]
 [ 45.  64.  34. 103. 119.]
 [ 94.  96. 152.  39.  32.]
 [ 55. 132.  53.  11.  68.]
 [ 67.  53.  78.  29.  22.]]

印出排序後的陣列:
 [[ 23.  37.  63.  77. 132.]
 [ 34.  45.  64. 103. 119.]
 [ 32.  39.  94.  96. 152.]
 [ 11.  53.  55.  68. 132.]
 [ 22.  29.  53.  67.  78.]]

印出排序後的索引:
 [[0 3 4 2 1]
 [2 0 1 3 4]
 [4 3 0 1 2]
 [3 2 0 4 1]
```

np.genfromtxt([資料檔名稱], dtype=[資料格式], unpack=[True 或 False], skip_header=[列數], usecols=[欄位索引值])

運算函數	說明
sum	加總
prod	乘積
mean	平均值
max	最大值
min	最小值

```python
import numpy as np

fileValue = np.genfromtxt('day.csv', delimiter=',', skip_header=1)
print("印出取得數值:\n", fileValue, "\n") # 印出取得數值
print("取得最大值:", np.max(fileValue)) # 取得最大值
print("取得最小值:", np.min(fileValue)) # 取得最小值

# axis=0 行, axis=1 列
print("取得「每行最大值」:", np.max(fileValue, axis=0)) # 取得每行最大值
print("取得「每列最小值」:", np.min(fileValue, axis=1)) # 取得每列最小值

print("取得「每行加總」後數值:", np.sum(fileValue, axis=0)) # 取得每行加總後數值
print("取得「每列加總」後數值:", np.sum(fileValue, axis=1)) # 取得每列加總後數值

print("取得「每行乘積」後數值:", np.prod(fileValue, axis=0)) # 取得每行乘積後數值
print("取得「每列乘積」後數值:", np.prod(fileValue, axis=1)) # 取得每列乘積後數值

print("取得「每行平均」後數值:", np.mean(fileValue, axis=0)) # 取得每行平均值後數值
print("取得「每列平均」後數值:", np.mean(fileValue, axis=1)) # 取得每列平均值後數值
```

```
印出取得數值:
 [[ 23. 132.  77.  37.  63.]
 [ 45.  64.  34. 103. 119.]
 [ 94.  96. 152.  39.  32.]
 [ 55. 132.  53.  11.  68.]
 [ 67.  53.  78.  29.  22.]]

取得最大值: 152.0
取得最小值: 11.0
取得「每行最大值」: [ 94. 132. 152. 103. 119.]
取得「每列最小值」: [23. 34. 32. 11. 22.]
取得「每行加總」後數值: [284. 477. 394. 219. 304.]
取得「每列加總」後數值: [332. 365. 413. 319. 249.]
取得「每行乘積」後數值: [3.58513650e+08 5.67381197e+09 1.64506742e+09 4.74126510e+07
 3.58896384e+08]
取得「每列乘積」後數值: [5.44922532e+08 1.20020544e+09 1.71181670e+09 2.87815440e+08
 1.76711964e+08]
取得「每行平均」後數值: [56.8 95.4 78.8 43.8 60.8]
取得「每列平均」後數值: [66.4 73.  82.6 63.8 49.8]
```

統計函數	說明
std	標準差
var	變異數
median	中位數
percentile	百分比
ptp	最大值與最小值差值

```python
import numpy as np

numpyArray = np.random.randint(50, size=20)

print("隨機產出的數值為:\n", numpyArray)

print("取得「標準差」:", np.std(numpyArray)) # 取得標準差
print("取得「變異數」:", np.var(numpyArray)) # 取得變異數
print("取得「中位數」:", np.median(numpyArray)) # 取得中位數
print("取得「百分比」:", np.percentile(numpyArray, 100)) # 取得百分比
print("取得「最大值與最小值差值」:", np.ptp(numpyArray)) # 取得最大值與最小值差值
```

```
隨機產出的數值為:
 [27 48 43  2 29 21  0 41 40  5 23 27 42 19 20 44 14 11 43  7]
取得「標準差」: 15.175967843930087
取得「變異數」: 230.31000000000003
取得「中位數」: 25.0
取得「百分比」: 48.0
取得「最大值與最小值差值」: 48
```

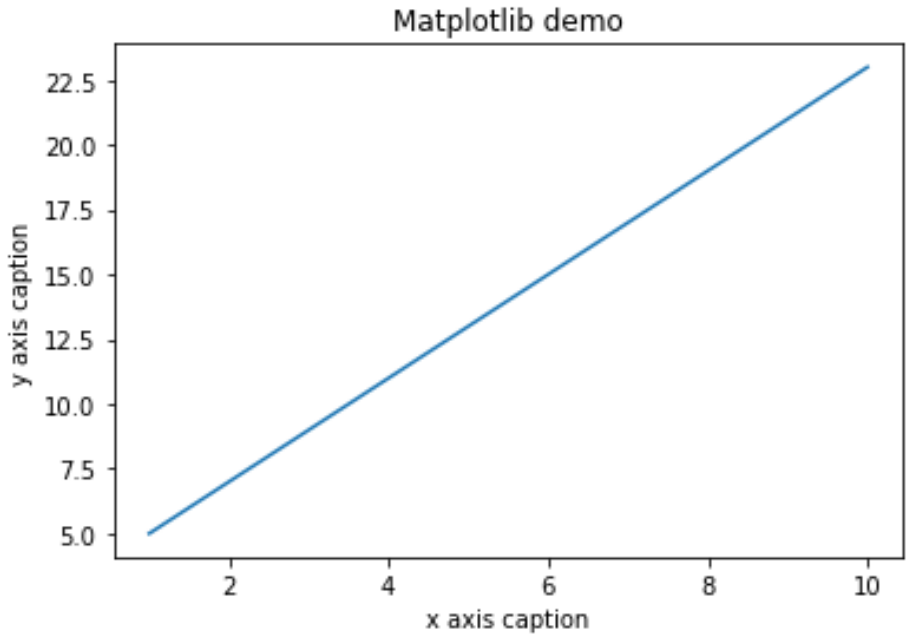
$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

μ為平均值（$\overline{x}$）。

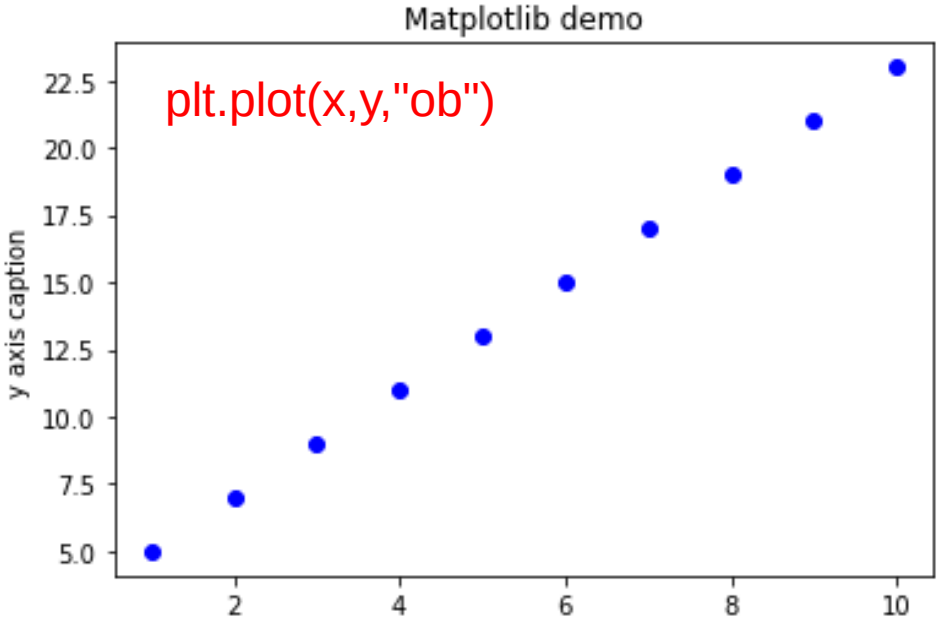
$$\mathrm{Var}(X) = \mathrm{E}\left[X^2 - 2X\,\mathrm{E}[X] + (\mathrm{E}[X])^2\right] = \mathrm{E}\left[X^2\right] - 2\,\mathrm{E}[X]\,\mathrm{E}[X] + (\mathrm{E}[X])^2 = \mathrm{E}\left[X^2\right] - (\mathrm{E}[X])^2$$

NumPy + pyplot

```
import numpy as np
from matplotlib import pyplot as plt

x = np.arange(1,11)
y =  2 * x + 3
plt.title("Matplotlib demo")
plt.xlabel("x axis caption")
plt.ylabel("y axis caption")
plt.plot(x,y)
plt.show()
```

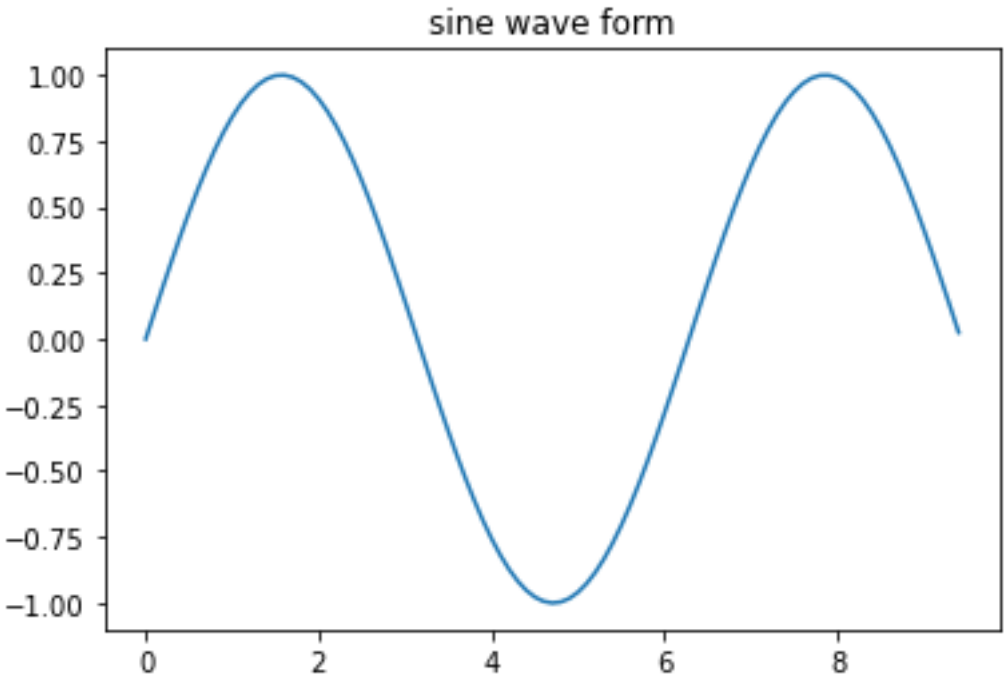


#藍色:'b' | 綠色: 'g' | 紅色: 'r' | 藍綠色:'c' | 紅紫色:'m' | 黃色:'y' | 黑色:'k' | 白色:'w' #實線:'-' | 虛線: '--' | 虛點線:'-.' | 點線:':' | 點:'.' | #圓形:'o' | 上三角:'^' | 下三角:'v' | 左三角:'<' | 右三角:'>' | 方形:'s' | 加號:'+' | 叉形:'x' | 棱形:'D' | 細棱形:'d' #三腳朝下:'1' | 三腳朝上:'2' | 三腳朝左:'3' | 三腳朝右:'4' | 六角形:'h' | 旋轉六角形:'H' | 五角形:'p' | 垂直線:'|'



NumPy + pyplot

```
import numpy as np
import matplotlib.pyplot as plt
# 計算正弦曲線上點的 x 和 y 座標
x = np.arange(0,  3  * np.pi,  0.1)
y = np.sin(x)
plt.title("sine wave form")
# 使用 matplotlib 來繪製點
plt.plot(x, y)
plt.show()
```



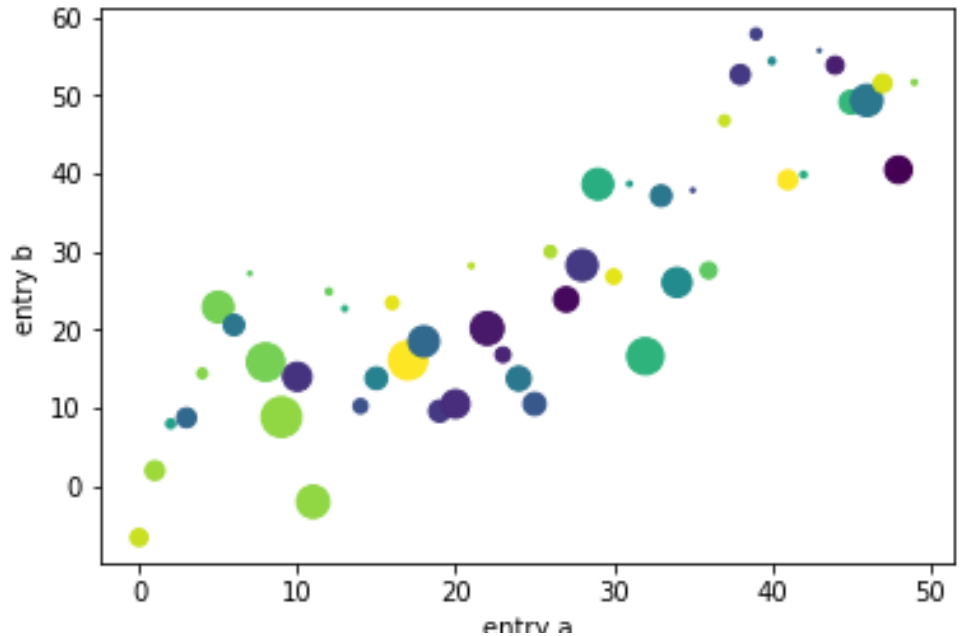
arange([start,] stop[, step,], dtype=None)

[start]：選填，起始值
stop：結束值，產生的數列不包含此結束值
[step]：選填，資料值間隔
[dtype]：選填，資料值的資料型態

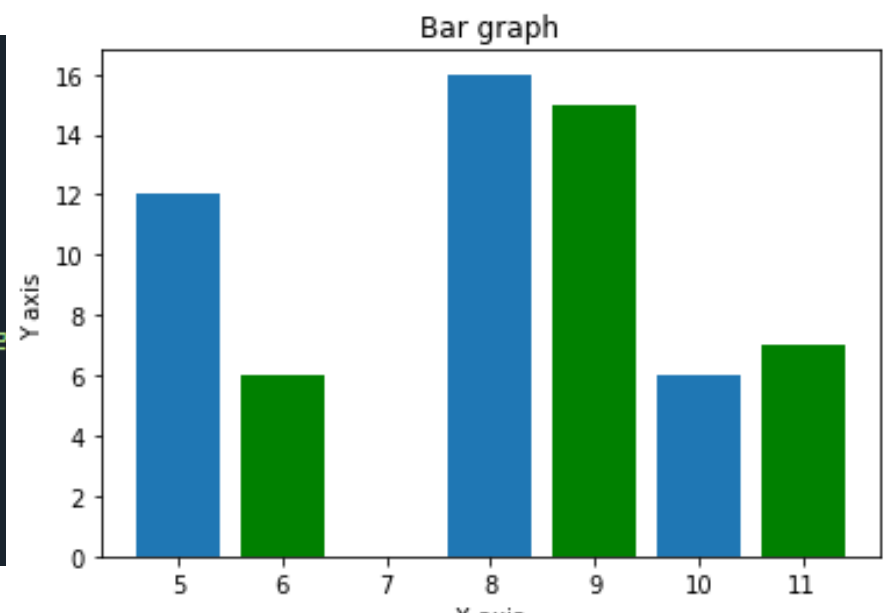
np.random.randn(size)：由一個平均為0，變異數為1的高斯分布中隨機取點，並以list儲存。
np.random.randint(low, high, size, dtype='l')：由low到high中產生一個size大小的list。

x y

```
import numpy as np
from matplotlib import pyplot as plt
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```



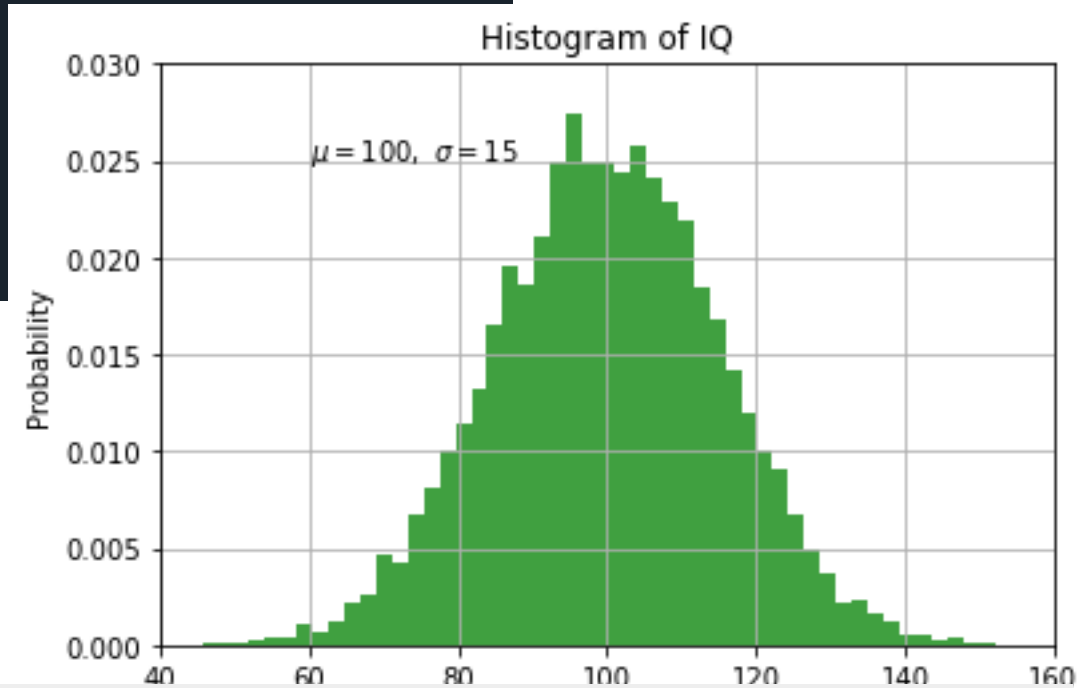

```
from matplotlib import pyplot as plt
x =  [5,8,10]
y =  [12,16,6]
x2 =  [6,9,11]
y2 =  [6,15,7]
plt.bar(x, y, align =  'center')
plt.bar(x2, y2, color =  'g', align =  'cente
plt.title('Bar graph')
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```



NumPy + pyplot

```
import numpy as np
from matplotlib import pyplot as plt
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

x y



matplotlib.pyplot.hist(x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rw

Subplot

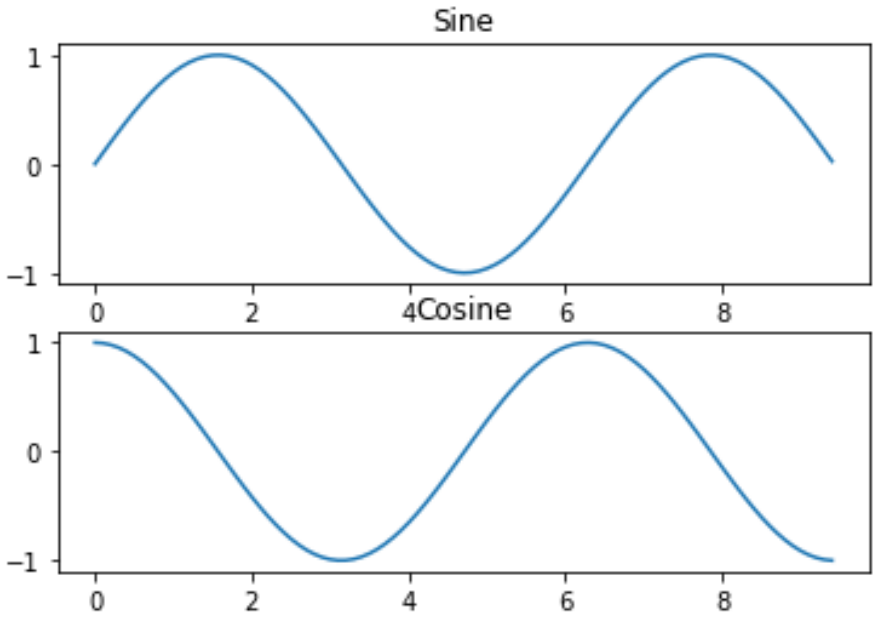
```
subplot(nrows, ncols, plot_number)
```

- *nrows* and *ncols* are used to notionally split the figure into nrows * ncols sub-axes,
- *plot_number* is used to identify the particular subplot that this function is to create within the notional grid.

- *plot_number* starts at 1, increments across rows first and has a maximum of nrows * ncols.

Subplot

```python
import numpy as np
import matplotlib.pyplot as plt
# 計算正弦和餘弦曲線上的點的 x 和 y 座標
x = np.arange(0,  3  * np.pi,  0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
# 建立 subplot 網格，高爲 2，寬爲 1
plt.subplot(2,  1,  1)
# 繪製第一個圖像
plt.plot(x, y_sin)
plt.title('Sine')
# 繪製第二個圖像
plt.subplot(2,  1,  2)
plt.plot(x, y_cos)
plt.title('Cosine')
# 展示圖像
plt.show()
```



Subplot

```
import matplotlib.pyplot as plt
import numpy as np
import  pandas as pd
x=np.arange(1,100)
fig=plt.figure()
ax1=fig.add_subplot(221) #2*2的圖形 在第一個位置
ax1.plot(x,x)
ax2=fig.add_subplot(222)
ax2.plot(x,-x)
ax3=fig.add_subplot(223)
ax3.plot(x,x**2)
ax3=fig.add_subplot(224)
ax3.plot(x,np.log(x))
plt.show()
```