第3章 Python各種物件資料的運算與處理

File : python_training_2025.01.07_python_introduction.ipynb

Author : Ming-Chang Lee

Date : 2025.01.07

YouTube : https://www.youtube.com/@alan9956

RWEPA : http://rwepa.blogspot.tw/

GitHub : https://github.com/rwepa

Email : alan9956@gmail.com

大綱

3.1 Python與Anaconda簡介

- Python 簡介與安裝
  - 官網 https://www.python.org/
  - Python is a programming language that lets you work more quickly and integrate your systems more effectively.
  - 吉多·范羅蘇姆(Guido van Rossum, 荷蘭程式設計師) 1989年的聖誕節期間研發 Python 語言. https://en.wikipedia.org/wiki/Guido_van_Rossum
- Python 特性 (v3.13.1, 2025年1月5日)
  - 跨平台

    - 開放性
    - 易讀性(使用冒號 : , 目的為區隔程式區塊, Python 不使用大括號 { ... })
    - 直譯語言
    - 豐富套件(模組) 例: Cython 編譯成執行檔(.exe)
    - 結合其他程式語言 C, C++, Python.NET, R
    - 物件導向程式語言

注意: 本課程使用 Anaconda 免費軟體編輯與執行 Python 程式, 該軟體已經包括 Pyhon 主程式, 因此不用額外下載 Pyhon 主程式.

- Anaconda 簡介與安裝
    - 官網 https://www.anaconda.com/
- Anaconda 特性
    - Anaconda是一個開源的Python和R語言的發行版本，用於計算科學（資料科學、機器學習、巨量資料處理和預測分析），Anaconda致力於簡化軟體套件管理系統和部署。
    - Anaconda透過 Conda 進行軟體套件管理，並擁有許多適用於 Windows、Linux和MacOS的資料科學軟體套件。
- Anaconda 下載並安裝
    - https://www.anaconda.com/download/success
- PyPI (Python Package Index) - 約59萬 Python 專案
    - https://pypi.org/

實作練習

程式集 \ Anaconda (anaconda3) \ Jupyter Notebook \ New \ Python3 Jupyter Notebook 輸入以下程式碼練習

程式1

import numpy

numpy.random.rand(2, 3)

程式2

from numpy import *

random.rand(2, 3)

程式3【推薦使用此方法】

import numpy as np

np.random.rand(2, 3)

程式4【推薦使用此方法】

from numpy import random

random.rand(2, 3)

Jupyter Notebook – 更改預設目錄

程式集 \ Anaconda (anaconda3) \ Anaconda Prompt \ 輸入以下程式碼練習

cd C:
jupyter-notebook

Jupyter Notebook 快速鍵

- 按 [Esc] cell旁邊為藍色
- 按 x：刪除當前選擇的cell
- 按 a：在當前選擇的上方新增一個cell
- 按 b：在當前選擇的下方新增一個cell
- 按 Shift + Enter：執行當前的cell並且選到下一個cell
- 按 Ctrl + Enter：執行當前cell
- 按 M：轉成markerdown模式，可以看到紅色框框內容從code變成markerdown

實作練習

開啟下列 ipynb 檔案

Python 程式設計-李明昌 <免費電子書>

http://rwepa.blogspot.com/2020/02/pythonprogramminglee.html

https://github.com/rwepa/DataDemo/blob/master/Python_Programming_Lee_ipynb.zi

安裝 Orange

- 方法1 conda 安裝

  conda install -c conda-forge orange3

- 方法2 下載 windows 安裝版或免安裝版

  https://orangedatamining.com/download/

- Anaconda Prompt 開啟 Orange

  python -m Orange.canvas

- Python Orange - 關聯規則教學

  YouTube: https://youtu.be/rh5GxJamtNg

  LINK: https://rwepa.blogspot.com/2022/07/python-orange-associate-tutorial.html

  PDF: https://github.com/rwepa/orange3_associate/blob/main/2022.07.02-orange-associate.pdf

Anaconda 模組管理

Anaconda Prompt 輸入以下程式碼練習

- 顯示已安裝模組

  conda list

- 尋找各版本官網套件

  conda search matplotlib

- 安裝模組

  conda install 模組名稱

- 更新模組

  conda update 模組名稱

實作練習

熟悉 Spyder 自動換列等設定, Spyder \ Tools \ Preferences \ Editor \ Display \ Wrap lines

恭喜您, 開啟人生 Python 學習之旅 ^_^

In [1]:
```
"""
# Python 執行-命令提示列
# 建立 C:\mydata\helloworld.py, 輸入以下程式碼
print("Python大數據分析")

# cd C:\mydata
# python --version
# dir
# python helloworld.py
"""
```

```
<>:1: SyntaxWarning: invalid escape sequence '\m'
<>:1: SyntaxWarning: invalid escape sequence '\m'
C:\Users\rwepa\AppData\Local\Temp\ipykernel_91012\402394083.py:1: SyntaxWarning:
invalid escape sequence '\m'
  """
```

```
Out[1]: '\n# Python 執行-命令提示列\n# 建立 C:\\mydata\\helloworld.py, 輸入以下程式碼\npri
nt("Python大數據分析")\n\n# cd C:\\mydata\n# python --version\n# dir\n# python h
elloworld.py\n'
```

Python變數

```
In [2]: # 合法變數
大數據 = 1 # 中文亦可, 建議不要使用
```

```
In [3]: CustomerSaleReport = 1
print(CustomerSaleReport)
```

```
1
```

```
In [4]: _CustomerSaleReport = 1 # 使用一個下底線,表示 Private variable, 同理練以下 print 函
```

```
In [5]: Customer_Sale_Report = 1
```

```
In [6]: customer_sale_report = 1
```

不合法變數

```
In [7]: # SyntaxError: invalid syntax
# $CustomerSaleReport = 1
```

```
In [8]: try:
    eval('$CustomerSaleReport = 1')
except SyntaxError:
    print("SyntaxError-語法錯誤")
```

```
SyntaxError-語法錯誤
```

```
In [9]: # SyntaxError: invalid decimal literal
# 2020_sale = 100
```

```
In [10]: # SyntaxError: invalid syntax
# break = 123
```

```
In [11]: # 內建保留字
dir(__builtins__)
```

```
Out[11]: ['ArithmeticError',
          'AssertionError',
          'AttributeError',
          'BaseException',
          'BaseExceptionGroup',
          'BlockingIOError',
          'BrokenPipeError',
          'BufferError',
          'BytesWarning',
          'ChildProcessError',
          'ConnectionAbortedError',
          'ConnectionError',
          'ConnectionRefusedError',
          'ConnectionResetError',
          'DeprecationWarning',
          'EOFError',
          'Ellipsis',
          'EncodingWarning',
          'EnvironmentError',
          'Exception',
          'ExceptionGroup',
          'False',
          'FileExistsError',
          'FileNotFoundError',
          'FloatingPointError',
          'FutureWarning',
          'GeneratorExit',
          'IOError',
          'ImportError',
          'ImportWarning',
          'IndentationError',
          'IndexError',
          'InterruptedError',
          'IsADirectoryError',
          'KeyError',
          'KeyboardInterrupt',
          'LookupError',
          'MemoryError',
          'ModuleNotFoundError',
          'NameError',
          'None',
          'NotADirectoryError',
          'NotImplemented',
          'NotImplementedError',
          'OSError',
          'OverflowError',
          'PendingDeprecationWarning',
          'PermissionError',
          'ProcessLookupError',
          'RecursionError',
          'ReferenceError',
          'ResourceWarning',
          'RuntimeError',
          'RuntimeWarning',
          'StopAsyncIteration',
          'StopIteration',
          'SyntaxError',
          'SyntaxWarning',
          'SystemError',
          'SystemExit',
          'TabError',
          'TimeoutError',
          'True',
          'TypeError',
          'UnboundLocalError',
          'UnicodeDecodeError',
          'UnicodeEncodeError',
          'UnicodeError',
          'UnicodeTranslateError',
          'UnicodeWarning',
          'UserWarning',
          'ValueError',
          'Warning',
          'WindowsError',
          'ZeroDivisionError',
          '__IPYTHON__',
          '__build_class__',
          '__debug__',
          '__doc__',
          '__import__',
          '__loader__',
          '__name__',
          '__package__',
          '__spec__',
          'abs',
          'aiter',
          'all',
          'anext',
          'any',
          'ascii',
          'bin',
          'bool',
          'breakpoint',
          'bytearray',
          'bytes',
          'callable',
          'chr',
          'classmethod',
          'compile',
          'complex',
          'copyright',
          'credits',
          'delattr',
          'dict',
          'dir',
          'display',
          'divmod',
          'enumerate',
          'eval',
          'exec',
          'execfile',
          'filter',
          'float',
          'format',
          'frozenset',
          'get_ipython',
          'getattr',
          'globals',
          'hasattr',
          'hash',
          'help',
          'hex',
          'id',
          'input',
          'int',
          'isinstance',
          'issubclass',
          'iter',
          'len',
          'license',
          'list',
          'locals',
          'map',
          'max',
          'memoryview',
          'min',
          'next',
          'object',
          'oct',
          'open',
          'ord',
          'pow',
          'print',
          'property',
          'range',
          'repr',
          'reversed',
          'round',
          'runfile',
          'set',
          'setattr',
          'slice',
          'sorted',
          'staticmethod',
          'str',
          'sum',
          'super',
          'tuple',
          'type',
          'vars',
          'zip']
```

In [12]:
```
len(dir(__builtins__)) # 161
```

Out[12]:
```
161
```

In [13]:
```
# 指派多個變數
x, y, z = "台北", "台中", "高雄"
print(x, y, z)
type(x) # str
```

```
台北 台中 高雄
```

Out[13]:
```
str
```

In [14]:
```
address = ["台北", "台中", "高雄"]
x, y, z = address
print(x)
print(y)
print(z)
type(x)
```

```
台北
台中
高雄
```

Out[14]:
```
str
```

In [15]:
```
# Python Style Rules
# https://google.github.io/styleguide/pyguide.html

# Python 註解
# 使用一個 #        用於1行註解
# 使用二個 """   用於超過1行註解或函數之說明文件

# Python採用內縮4個空白鍵之語法
```

3.2 資料型別與運算子

資料型別(資料型態)

- https://docs.python.org/3/library/stdtypes.html

廣義 Data Types

- Text Type: str 字串
- Numeric Types: int, float, complex 整數, 浮點數, 複數
- Boolean Type: bool 布林 [True, False]
- Binary Types: bytes, bytearray, memoryview
- Sequence Types: list, tuple, range
- Set Types: set, frozenset
- Mapping Type: dict

參考: https://www.w3schools.com/python/

In [16]:
```
# 資料型別-範例

# 整數 int
x1 = 1
type(x1)
```

Out[16]:
```
int
```

In [17]:
```
# 浮點數 float
x2 = 1.234
type(x2)
```

Out[17]:
```
float
```

In [18]:
```
# 複數  complex
x3 = 1+2j
type(x3)
```

```
Out[18]:  complex
```

```
In [19]:  # 布林值 (Boolean)
          x4 = True
          type(x4)
```

```
Out[19]:  bool
```

```
In [20]:  x4 > 10
```

```
Out[20]:  False
```

```
In [21]:  # None值
          import numpy as np

          None == False
```

```
Out[21]:  False
```

```
In [22]:  None == 0
```

```
Out[22]:  False
```

```
In [23]:  None == np.nan
```

```
Out[23]:  False
```

```
In [24]:  None == None
```

```
Out[24]:  True
```

```
In [25]:  False == 0
```

```
Out[25]:  True
```

```
In [26]:  True == 1
```

```
Out[26]:  True
```

```
In [27]:  # 整數亂數
          import random
          random.seed(168) # 設定亂數種子
          myrandom = random.randrange(1, 100) # 沒有包括100值
          print(myrandom)

         96
```

```
In [28]:  # 運算子
          3 + 5
```

```
Out[28]:  8
```

```
In [29]:  3 + (5 * 4)
```

```
Out[29]:  23
```

```
In [30]: 3 ** 2
Out[30]: 9

In [31]: "Hello" + "World"
         # 123 + "RWPEA"  # Error
Out[31]: 'HelloWorld'

In [32]: 1 + 1.234
Out[32]: 2.234

In [33]: 7 / 2
Out[33]: 3.5

In [34]: 7 // 2         # 商數(quotient)
Out[34]: 3

In [35]: 7 % 2          # 餘數(remainder)
Out[35]: 1

In [36]: divmod(7, 2)   # (商數, 餘數)
Out[36]: (3, 1)

In [37]: 2 ** 10        # 次方
Out[37]: 1024

In [38]: 1.234e3 - 1000
Out[38]: 234.0

In [39]: x5 = 1 == 2
         x5
Out[39]: False

In [40]: x5 + 10
Out[40]: 10

In [41]: # 位移運算子: << 向左位移
         # 位移運算子: >> 向右位移
         a = 4 << 3 # 0100 --> 0100000, 32 16 8 4 2 1
         print(a)
        32

In [42]: b = a * 4.5
         print(b)
        144.0
```

```
In [43]: c = (a+b)/2.5
         print(c)
```

```
70.4
```

```
In [44]: # 指派運算子
         x = 9
         x+=2
         print(x)
```

```
11
```

3.3 四大基本Python物件

1. Tuple 序列 (元組) - (value,...) 不可變 (Immutable)
2. List 串列(清單) - [value,...] 可變 (mutable)
3. Set 集合 - {value,...} 可變 (mutable)
4. Dict 字典 - {key:value,...} 可變 (mutable)

1. Tuple 序列 (元組)

- tuple 是 Python 的資料儲存容器之一, 最大的特點就是, 它是「不可變」的資料型態。
- 與list類似，最大的不同tuple是一種唯讀且不可變更的資料結構
- 不可取代tuple中的任意一個元素，因為它是唯讀不可變更的
- Tuple 是具有 ordered 特性
- Python 的索引(指標)從0開始

```
In [45]: # 建立序列
         x1 = 1
         x2 = 1,
         x3 = 1, 2, 3
         # 練習 type
```

```
In [46]: f = (2,3,4,5) # A tuple of integers
         print(f)
```

```
(2, 3, 4, 5)
```

```
In [47]: g = () # An emptmy tuple
         print(g)
```

```
()
```

```
In [48]: h = (2, [3,4], (10,11,12))      # A tuple containing mixed objects
         print(h)
```

```
(2, [3, 4], (10, 11, 12))
```

```
In [49]: # Tuples操作
         x = f[1] # Element access. x = 3
         x
```

```
Out[49]:  3

In [50]:  y = f[1:3] # Slices. y = (3,4)
          y

Out[50]:  (3, 4)

In [51]:  z = h[1][1]     # Nesting. z = 4
          z

Out[51]:  4

In [52]:  personal = ('Hannah', 14, 5*12+6)
          personal

Out[52]:  ('Hannah', 14, 66)

In [53]:  singleton = ("hello",)
          singleton

Out[53]:  ('hello',)

In [54]:  type(singleton) # tuple

Out[54]:  tuple

In [55]:  singleton1 = ("hello")
          singleton1

Out[55]:  'hello'

In [56]:  type(singleton1) # 結果與上述程式碼不同.

Out[56]:  str

In [57]:  # single format: tuple[index]
          # index : 0  ~  len(tuple)-1
          # index: -len(tuple)  ~  -1
          f= (2,3,4,5)
          f[0]

Out[57]:  2

In [58]:  f[-1] # 索引 -1 表示倒數第1個元素

Out[58]:  5

In [59]:  f[-2]

Out[59]:  4

In [60]:  f[len(f)-1]

          # slice format: tuple [start:end ]. Items from start to (end -1)
          t=((1,2), (2,"Hi"), (3,"RWEPA"), 2+3j, 6E23)
          t[2]
```

```
Out[60]: (3, 'RWEPA')

In [61]: t[:3]

Out[61]: ((1, 2), (2, 'Hi'), (3, 'RWEPA'))

In [62]: t[3:]

Out[62]: ((2+3j), 6e+23)

In [63]: t[-1]

Out[63]: 6e+23

In [64]: t[-3:]

Out[64]: ((3, 'RWEPA'), (2+3j), 6e+23)

In [65]: # tuple 長度
         len(t) # 5

Out[65]: 5

In [66]: # tuple 建構子
         # 使用 tuple(( ... )) 或 tuple([ ... ])
         employeeGender = tuple(("男", "女", "女"))
         employeeGender

Out[66]: ('男', '女', '女')

In [67]: # tuple unpacking - 將元素指派至變數
         fruits = ("apple", "banana", "cherry")
         (green, yellow, red) = fruits
         print(green)
         print(yellow)
         print(red)

         # TRY: green, yellow, red = fruits

        apple
        banana
        cherry

In [68]: # tuple unpacking - 使用萬用字元*
         fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
         (green, yellow, *red) = fruits
         print(green)
         print(yellow)
         print(red)

        apple
        banana
        ['cherry', 'strawberry', 'raspberry']

In [69]: # tuple - loop 處理
         fruits = ("apple", "banana", "cherry")

         # 方法1. tuple - 取出元素, 使用for
```

```
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

In [70]:
```
# 方法2. tuple - 取出元素, 使用while
i = 0
while i < len(fruits):
  print(fruits[i])
  i = i + 1
```

```
apple
banana
cherry
```

In [71]:
```
# 方法3. tuple - 取出元素, 使用指標 range, len
for i in range(len(fruits)):
  print(fruits[i])
```

```
apple
banana
cherry
```

In [72]:
```
# tuple - join 結合
tuple1 = ("台北", "台中", "高雄")
tuple2 = ("男", "女", "女")
tuple3 = tuple1 + tuple2
print(tuple3)
```

```
('台北', '台中', '高雄', '男', '女', '女')
```

In [73]:
```
# tuple - 重複
tuple1*3
3*tuple1
```

Out[73]:
```
('台北', '台中', '高雄', '台北', '台中', '高雄', '台北', '台中', '高雄')
```

In [74]:
```
# count 次數統計
tuple = ("男", "女", "女", "男", "女")
tuple.count("男") # 2
```

Out[74]:
```
2
```

In [75]:
```
tuple.count("女") # 3
```

Out[75]:
```
3
```

2. List 串列(清單)

In [76]:
```
# 建立串列
a = [2, 3, 4]           # 整數串列
b = [2, 7, 3.5, "Hello"] # 混合資料串列
c = []                  # 空串列
d = [2, [a, b]]         # 巢狀串列
```

In [77]:
```
# 串列的操作
a
```
Out[77]:
```
[2, 3, 4]
```
In [78]:
```
a[1]        # 取得第2個元素
```
Out[78]:
```
3
```
In [79]:
```
a[-1]       # 取得最後一個元素
```
Out[79]:
```
4
```
In [80]:
```
b[1:3]      # 串列篩選
```
Out[80]:
```
[7, 3.5]
```
In [81]:
```
d[1][0][2] # 巢狀串列操作
```
Out[81]:
```
4
```
In [82]:
```
b[0]        # 2
```
Out[82]:
```
2
```
In [83]:
```
b[0] = 42   # 修改元素值
b[0]        # 42
```
Out[83]:
```
42
```
In [84]:
```
# 串列 slice format
t=[1, 2, (3,"Hi"), [4,"RWEPA"], 2+3j, 6E7]
t
```
Out[84]:
```
[1, 2, (3, 'Hi'), [4, 'RWEPA'], (2+3j), 60000000.0]
```
In [85]:
```
t[2]
```
Out[85]:
```
(3, 'Hi')
```
In [86]:
```
t[:3]
```
Out[86]:
```
[1, 2, (3, 'Hi')]
```
In [87]:
```
t[3:]
```
Out[87]:
```
[[4, 'RWEPA'], (2+3j), 60000000.0]
```
In [88]:
```
t[-1]
```
Out[88]:
```
60000000.0
```
In [89]:
```
t[-3:]
```
Out[89]:
```
[[4, 'RWEPA'], (2+3j), 60000000.0]
```

In [90]:
```
# 串列長度
len(t)
```

Out[90]:
```
6
```

In [91]:
```
# list 建構子
# 使用 list(( ... )) 或 list([ ... ])
mylist1 = list(("男", "女", "女"))
mylist1
```

Out[91]:
```
['男', '女', '女']
```

In [92]:
```
mylist2 = list(["男", "女", "女"])
mylist2
```

Out[92]:
```
['男', '女', '女']
```

In [93]:
```
mylist1 == mylist2
```

Out[93]:
```
True
```

In [94]:
```
# 串列 unpacking - 將元素指派至變數
fruits = ["apple", "banana", "cherry"]
green, yellow, red = fruits
print(green)
print(yellow)
print(red)
type(green) # str
```

```
apple
banana
cherry
```

Out[94]:
```
str
```

In [95]:
```
# 串列 unpacking - 使用萬用字元*
fruits = ["apple", "banana", "cherry", "strawberry", "raspberry"]
green, yellow, *red = fruits
print(green)
print(yellow)
print(red)
type(green) # str
```

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

Out[95]:
```
str
```

In [96]:
```
# 串列 - Loop 處理
mylist = [1, 2, 3, [4, 5], ["A", "B", "C"]]
```

In [97]:
```
# 練習 Loop 方法
# 方法1. list - 取出元素, 使用for
for x in mylist:
  print(x)
```

```
1
2
3
[4, 5]
['A', 'B', 'C']
```

In [98]:
```
# 方法2. list - 取出元素, 使用while
i = 0
while i < len(mylist):
  print(mylist[i])
  i = i + 1
```

```
1
2
3
[4, 5]
['A', 'B', 'C']
```

In [99]:
```
# 方法3. list - 取出元素, 使用指標 range, len
for i in range(len(mylist)):
  print(mylist[i])
```

```
1
2
3
[4, 5]
['A', 'B', 'C']
```

In [100…
```
# 方法4. list - 取出元素, 使用串列包含法 (List Comprehension)
[print(x) for x in mylist]
```

```
1
2
3
[4, 5]
['A', 'B', 'C']
```

Out[100…
```
[None, None, None, None, None]
```

In [101…
```
# 串列包含法應用

# for 資料篩選-包括字母 a
codes = ["Python", "R", "SQL", "Julia", ".NET", "Java", "JavaScript"]
newlist = []
for x in codes:
  if "a" in x:
    newlist.append(x)
print(newlist)
```

```
['Julia', 'Java', 'JavaScript']
```

In [102…
```
# 串列包含法應用1
# 亦可用於序列, 集合, 字典等可反覆運算物件(可迭代物件, iterable object)
codes = ["Python", "R", "SQL", "Julia", ".NET", "Java", "JavaScript"]
newlist = [x for x in codes if "a" in x]
print(newlist)
```

```
['Julia', 'Java', 'JavaScript']
```

In [103…
```
# 串列包含法應用2
newlist = [x.upper() for x in codes]
print(newlist)
```

```
['PYTHON', 'R', 'SQL', 'JULIA', '.NET', 'JAVA', 'JAVASCRIPT']
```

In [104…
```
# AttributeError: 'list' object has no attribute 'upper'
# codes.upper()
```

In [105…
```
# 串列包含法應用3
newlist = ['RWEPA' for x in codes]
print(newlist)
```

```
['RWEPA', 'RWEPA', 'RWEPA', 'RWEPA', 'RWEPA', 'RWEPA', 'RWEPA']
```

In [106…
```
# 串列 join 結合
e = a + b  # Join two lists
e
```

Out[106…
```
[2, 3, 4, 42, 7, 3.5, 'Hello']
```

In [107…
```
# 串列 repeat 重複
f1 = a*3    # repeat lists
f1
```

Out[107…
```
[2, 3, 4, 2, 3, 4, 2, 3, 4]
```

In [108…
```
f2 = 3*a
f2
```

Out[108…
```
[2, 3, 4, 2, 3, 4, 2, 3, 4]
```

In [109…
```
# 串列排序-預設為遞增排序,英文字母先大寫,再小寫
codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
codes.sort()
print(codes)
```

```
['.NET', 'JavaScript', 'Julia', 'R', 'SQL', 'java', 'python']
```

In [110…
```
# 串列排序-先全部小寫,再排序
codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
codes.sort(key = str.lower)
print(codes)
```

```
['.NET', 'java', 'JavaScript', 'Julia', 'python', 'R', 'SQL']
```

In [111…
```
# 串列排序-遞減排序
codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
codes.sort(reverse =True)
print(codes)
```

```
['python', 'java', 'SQL', 'R', 'Julia', 'JavaScript', '.NET']
```

In [112…
```
# 串列反序
codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
codes.reverse()
print(codes)
```

```
['JavaScript', 'java', '.NET', 'Julia', 'SQL', 'R', 'python']
```

In [113…
```
# 串列複製,等號會建立參考物件
a = [1, 2, 3]
a
b = a
```

```
b[0] = 999 # 修改b,亦會修改a
b
```

Out[113…

```
[999, 2, 3]
```

In [114…

```
a # a已經更新
```

Out[114…

```
[999, 2, 3]
```

In [115…

```
# 串列複製-使用 copy
a = [1, 2, 3]
b = a.copy()
b
b[0] = 999
```

In [116…

```
b
```

Out[116…

```
[999, 2, 3]
```

In [117…

```
a # a保持不變
```

Out[117…

```
[1, 2, 3]
```

In [118…

```
# 串列複製-使用 list
a = [1, 2, 3]
c = list(a)
c
c[0] = 123
c
```

Out[118…

```
[123, 2, 3]
```

In [119…

```
a # a保持不變
```

Out[119…

```
[1, 2, 3]
```

In [120…

```
# 附加元素 append
a = [1, 2, 3]
a.append(['BigData', 'SQL']) # 新增1個元素
a
a.append('2021/8/14')
a
```

Out[120…

```
[1, 2, 3, ['BigData', 'SQL'], '2021/8/14']
```

In [121…

```
# 延伸元素 extend
a.extend(['Python', 'R', "Julia"]) # 新增一個串列
a
```

Out[121…

```
[1, 2, 3, ['BigData', 'SQL'], '2021/8/14', 'Python', 'R', 'Julia']
```

In [122…

```
# 延伸元素 extend - 加入tuple,list,set,dict
a = [1, 2, 3]
a.extend(('4', '5', 'RWEPA')) # 延伸一個序列
a
```

Out[122…

```
[1, 2, 3, '4', '5', 'RWEPA']
```

In [123…
```
a.extend({'8', '8', '10'}) # 延伸一個集合
a
```

Out[123…
```
[1, 2, 3, '4', '5', 'RWEPA', '10', '8']
```

In [124…
```
a.extend({'a':'R', 'b':'Python'}) # 延伸一個字典-ONLY KEY, NO VALUE
a
```

Out[124…
```
[1, 2, 3, '4', '5', 'RWEPA', '10', '8', 'a', 'b']
```

In [125…
```
# 串列 - insert 插入元素
a = list(range(5))
a
```

Out[125…
```
[0, 1, 2, 3, 4]
```

In [126…
```
a.insert(2, 999) # 在指標為2的位置,插入新元素
a
```

Out[126…
```
[0, 1, 999, 2, 3, 4]
```

In [127…
```
# 串列 - remove, pop, del
# 刪除指定元素
a.remove(999)
a
```

Out[127…
```
[0, 1, 2, 3, 4]
```

In [128…
```
# 刪除指定指標元素
a.pop(1)
a
```

Out[128…
```
[0, 2, 3, 4]
```

In [129…
```
# 刪除指定指標元素
del a[1]
a
```

Out[129…
```
[0, 3, 4]
```

In [130…
```
# 刪除第一個元素
a.pop(0)
a
```

Out[130…
```
[3, 4]
```

In [131…
```
# 刪除最後一個元素
a.pop()
a
```

Out[131…
```
[3]
```

In [132…
```
# 清空物件元素,物件仍存在記憶體
a.clear()
a
```

Out[132…
```
[]
```

In [133…
```python
# 刪除物件,物件不存在記憶體
del a

# NameError: name 'a' is not defined
# print(a)
```

In [134…
```python
# 串列 - zip 應用
a = ("x1", "x2", "x3")
b = ("y1", "y2", "y3")
c = (1, 2, 3)

x = zip(a, b, c)
x
```

Out[134…
```
<zip at 0x23aa360c780>
```

In [135…
```python
list(x)
```

Out[135…
```
[('x1', 'y1', 1), ('x2', 'y2', 2), ('x3', 'y3', 3)]
```

In [136…
```python
# 顯示方法
print(dir(list))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__d
elitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribut
e__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__imul_
_', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rev
ersed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__s
ubclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
```

In [137…
```python
# 實作練習
# 如何顯示不以 __ 開始串列方法的總個數 11
```

3. Set 集合

- 集合與字典相似, 但字典沒有key,只有值
- 集合內容不可以修改
- 集合是 unordered
- 集合是 unindexed
- 集合會忽略重複的值

In [138…
```python
a = set() # 空集合
type(a)
```

Out[138…
```
set
```

In [139…
```python
b = {"台北市", "新北市", "桃園市", "台中市", "台北市", "新北市", "高雄市"}
b
```

```
# b[0] = 1 # TypeError: 'set' object does not support item assignment
# b[0]     # TypeError: 'set' object is not subscriptable
```

Out[139…

```
{'台中市', '台北市', '新北市', '桃園市', '高雄市'}
```

In [140…

```
len(b)
```

Out[140…

```
5
```

In [141…

```
# 使用 myset 練習集合 - Loop 方法
myset = {"台北市", "新北市", "桃園市", "台中市", "高雄市"}
myset
```

Out[141…

```
{'台中市', '台北市', '新北市', '桃園市', '高雄市'}
```

In [142…

```
# 集合新增元素 add, 因為集合是unordered, 不一定新增在最後一個
myset = {"台北市", "新北市", "桃園市", "台中市", "高雄市"}
myset.add("台南市")
myset
```

Out[142…

```
{'台中市', '台北市', '台南市', '新北市', '桃園市', '高雄市'}
```

In [143…

```
# 集合新增集合
myset.update({"澎湖", "金門"})
myset
```

Out[143…

```
{'台中市', '台北市', '台南市', '新北市', '桃園市', '澎湖', '金門', '高雄市'}
```

In [144…

```
# 刪除指定元素
myset.remove("澎湖")
myset
```

Out[144…

```
{'台中市', '台北市', '台南市', '新北市', '桃園市', '金門', '高雄市'}
```

In [145…

```
# 清空物件兀素,物件仍存在記憶體
myset.clear()
myset
```

Out[145…

```
set()
```

In [146…

```
# 刪除物件,物件不存在記憶體
del myset

# NameError: name 'myset' is not defined
# myset
```

In [147…

```
# 集合運算
x = {1,2,3,4,5}
y = {1,3,5,7}

x & y # {1, 3, 5} # 交集
```

Out[147…

```
{1, 3, 5}
```

In [148…

```
x.intersection(y) # 交集
```

Out[148…

```
{1, 3, 5}
```

In [149…
```
x | y # {1, 2, 3, 4, 5, 7} # 聯集
```
Out[149…
```
{1, 2, 3, 4, 5, 7}
```
In [150…
```
x.union(y) # 聯集
```
Out[150…
```
{1, 2, 3, 4, 5, 7}
```
In [151…
```
x ^ y # {2, 4, 7} # XOR 互斥
```
Out[151…
```
{2, 4, 7}
```
In [152…
```
x - y # 差集
```
Out[152…
```
{2, 4}
```
In [153…
```
x.difference(y) # 差集
```
Out[153…
```
{2, 4}
```

4. Dict 字典

In [154…
```
# 宣告字典
mydict = {
    "language": "Python",
    "designer": "Guido van Rossum",
    "year": 1991
    }

print(mydict)
type(mydict) # dict
```
```
{'language': 'Python', 'designer': 'Guido van Rossum', 'year': 1991}
```
Out[154…
```
dict
```
In [155…
```
# 重複 key, 只保留1個
mydict1 = {
    "language": "Python",
    "designer": "Guido van Rossum",
    "year": 1991,
    "year": 2021
    }

print(mydict1)
```
```
{'language': 'Python', 'designer': 'Guido van Rossum', 'year': 2021}
```
In [156…
```
# 字典存取元素
b = {
     "uid": 168,
     "login": "marvelous",
     "name" : 'Alan Lee'
     }
b
```

Out[156…

```
{'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee'}
```

In [157…

```python
# dict 取得所有 keys
mykeys = b.keys()
print(mykeys)
```

```
dict_keys(['uid', 'login', 'name'])
```

In [158…

```python
# dict 取得所有 values
myvalues = b.values()
print(myvalues)
```

```
dict_values([168, 'marvelous', 'Alan Lee'])
```

In [159…

```python
# dict 取得key的值
u = b["uid"] # 168
print(u)
```

```
168
```

In [160…

```python
# dict 更新值
b.update({"uid": 123})
print(b)
```

```
{'uid': 123, 'login': 'marvelous', 'name': 'Alan Lee'}
```

In [161…

```python
# dict 新增元素
b["shell"] = "/bin/sh"
print(b)
```

```
{'uid': 123, 'login': 'marvelous', 'name': 'Alan Lee', 'shell': '/bin/sh'}
```

In [162…

```python
# dict 刪除元素 - pop
b.pop("shell")
print(b)
```

```
{'uid': 123, 'login': 'marvelous', 'name': 'Alan Lee'}
```

In [163…

```python
# dict 刪除元素 - del
del b["login"]
print(b)
```

```
{'uid': 123, 'name': 'Alan Lee'}
```

In [164…

```python
# dict 清空整個物件 - clear
b.clear()
b
```

Out[164…

```
{}
```

In [165…

```python
# dict 刪除整個物件 -del
del b
# b
```

In [166…

```python
# 字典複製-使用 copy
mydict = {
    "uid": 168,
    "login": "marvelous",
    "name" : 'Alan Lee'
    }
mydict
```

Out[166…

```
{'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee'}
```

In [167…

```
mydict2 = mydict.copy()
print(mydict2)

# 字典複製-使用 dict
mydict3 = dict(mydict)
print(mydict3)

mydict2 == mydict3 # True
```

```
{'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee'}
{'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee'}
```

Out[167…

```
True
```

In [168…

```
# 巢狀字典 (Nested Dictionaries)
# 方法1 一次建立一個巢狀字典
mycodes = {
    "code1" : {
         "name" : "Fortran77",
         "year" : 1977
         },
    "code2" : {
        "name" : "Python",
        "year" : 1991
        },
    "code3" : {
        "name" : "R",
        "year" : 2000
        }
    }

mycodes
```

Out[168…

```
{'code1': {'name': 'Fortran77', 'year': 1977},
 'code2': {'name': 'Python', 'year': 1991},
 'code3': {'name': 'R', 'year': 2000}}
```

In [169…

```
# 方法2 建立三個字典,再合併為一項字典
mycode1 = {
    "name" : "Fortran77",
    "year" : 1977
    }

mycode2 = {
    "name" : "Python",
    "year" : 1991
    }

mycode3 = {
    "name" : "R",
    "year" : 2000
    }

mycodes2 = {
  "程式1" : mycode1,
  "程式2" : mycode2,
  "程式3" : mycode3
}
```

```
mycodes2
```

Out[169…

```
{'程式1': {'name': 'Fortran77', 'year': 1977},
 '程式2': {'name': 'Python', 'year': 1991},
 '程式3': {'name': 'R', 'year': 2000}}
```

In [170…

```
# 實作練習
# 將 list 轉換為 dictionary
# 輸入: lst = ['a', 1, 'b', 2, 'c', 3]
# 結果: {'a': 1, 'b': 2, 'c': 3}
```

模組 Modules

In [171…

```
# 使用模組
import math
math.sqrt(9)
```

Out[171…

```
3.0
```

In [172…

```
from math import sqrt
sqrt(9)
```

Out[172…

```
3.0
```

In [173…

```
# 模組的搜尋路徑
import sys
sys.path
# '' 表示現行目錄
```

Out[173…

```
['C:\\Users\\rwepa\\anaconda3\\python312.zip',
 'C:\\Users\\rwepa\\anaconda3\\DLLs',
 'C:\\Users\\rwepa\\anaconda3\\Lib',
 'C:\\Users\\rwepa\\anaconda3',
 '',
 'C:\\Users\\rwepa\\anaconda3\\Lib\\site-packages',
 'C:\\Users\\rwepa\\anaconda3\\Lib\\site-packages\\win32',
 'C:\\Users\\rwepa\\anaconda3\\Lib\\site-packages\\win32\\lib',
 'C:\\Users\\rwepa\\anaconda3\\Lib\\site-packages\\Pythonwin',
 'C:\\Users\\rwepa\\anaconda3\\Lib\\site-packages\\setuptools\\_vendor']
```

In [174…

```
# 切換工作目錄
import os
os.getcwd() # 讀取工作目錄
os.chdir("C:/") # 變更工作目錄
os.getcwd()
os.listdir(os.getcwd()) # 顯示檔案清單
```

Out[174…
```
['$Recycle.Bin',
 'Documents and Settings',
 'DumpStack.log.tmp',
 'hiberfil.sys',
 'Hncb',
 'LJP1100_P1560_P1600_Full_Solution',
 'mydata',
 'OEM',
 'OneDriveTemp',
 'PageFile.sys',
 'PerfLogs',
 'Program Files',
 'Program Files (x86)',
 'ProgramData',
 'rdata',
 'Recovery',
 'rtools44',
 'swapfile.sys',
 'System Volume Information',
 'Users',
 'Windows']
```

3.4 使用NumPy模組與reshape應用

In [175…
```
import numpy as np

####################
# 一維陣列
##############################

# 使用 tuple 或 list 建立一維陣列
a = np.array([1, 2, 3, 4, 5])
b = np.array((1, 2, 3, 4, 5), dtype=float)

print(a)
print(b)
print(type(a))
print(type(b))
```

```
[1 2 3 4 5]
[1. 2. 3. 4. 5.]
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

In [176…
```
print(a[0], a[1], a[2], a[3])
```

```
1 2 3 4
```

In [177…
```
b[0] = 5
print(b)
```

```
[5. 2. 3. 4. 5.]
```

In [178…
```
b[4] = 0
print(b)
```

```
[5. 2. 3. 4. 0.]
```

In [179…
```
# 二維陣列

# 使用巢狀清單建立二維陣列
# axis 0:列, axis 1:行
a = np.array([[1,2,3],[4,5,6]])
a
print(type(a))
print(a[0, 0], a[0, 1], a[0, 2])
print(a[1, 0], a[1, 1], a[1, 2])
```

```
<class 'numpy.ndarray'>
1 2 3
4 5 6
```

In [180…
```
a[0, 0] = 6
a[1, 2] = 1
print(a)
```

```
[[6 2 3]
 [4 5 1]]
```

In [181…
```
# np.arrange
a = np.arange(5) # [0 1 2 3 4]
print(a)
```

```
[0 1 2 3 4]
```

In [182…
```
b = np.arange(1, 11, 2) # 1<= x < 11
print(b) # [1 3 5 7 9]
```

```
[1 3 5 7 9]
```

In [183…
```
# np.zeros
np.zeros(5) # array([0., 0., 0., 0., 0.])
```

Out[183…
```
array([0., 0., 0., 0., 0.])
```

In [184…
```
np.zeros(5, dtype=int) # array([0, 0, 0, 0, 0])
```

Out[184…
```
array([0, 0, 0, 0, 0])
```

In [185…
```
np.zeros((3, 2)) # 建立3列,2行皆為零的陣列
# array([[0., 0.],
#        [0., 0.],
#        [0., 0.]])
```

Out[185…
```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [186…
```
# np.ones
np.ones(3) # array([1., 1., 1.])
```

Out[186…
```
array([1., 1., 1.])
```

In [187…
```
# np.full
np.full(shape = (3, 4), fill_value = 99)
# array([[99, 99, 99, 99],
#        [99, 99, 99, 99],
#        [99, 99, 99, 99]])
```

Out[187…
```
array([[99, 99, 99, 99],
       [99, 99, 99, 99],
       [99, 99, 99, 99]])
```

In [188…
```
# zeros_like
a = np.array([[1,2,3], [4,5,6]])
a
# array([[1, 2, 3],
#        [4, 5, 6]])
```

Out[188…
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [189…
```
np.zeros_like(a)
# [[0 0 0]
#  [0 0 0]]
```

Out[189…
```
array([[0, 0, 0],
       [0, 0, 0]])
```

In [190…
```
# ones_like
np.ones_like(a)
# [[1 1 1]
#  [1 1 1]]

##############################
```

Out[190…
```
array([[1, 1, 1],
       [1, 1, 1]])
```

陣列儲存與載入

In [191…
```
# 使用 Spyder 練習
"""
# 實作練習
# 使用 save 將 Numpy 陣列 a 儲存成外部檔案
import numpy as np

outputfile = 'myarray.npy'
with open(outputfile, 'wb') as fp:
    np.save(fp, a)

# 使用 load 將外部檔案匯入至Numpy陣列
outputfile = "myarray.npy"
with open(outputfile, 'rb') as fp:
    mydata = np.load(fp)
print(mydata)
"""
```

Out[191…
```
'\n# 實作練習\n# 使用 save 將 Numpy 陣列 a 儲存成外部檔案\nimport numpy as np\n\no
utputfile = \'myarray.npy\'\nwith open(outputfile, \'wb\') as fp:\n    np.save
(fp, a)\n\n# 使用 load 將外部檔案匯入至Numpy陣列\noutputfile = "myarray.npy"\nwit
h open(outputfile, \'rb\') as fp:\n    mydata = np.load(fp)\nprint(mydata)\n'
```

常數 Constants

In [192…
```
import numpy as np
```

```
x = np.Inf # 無限大 inf
print(x)
```

```
inf
```

In [193…

```
y = np.NAN # nan
print(y)
```

```
nan
```

In [194…

```
# 新版本使用 nan
np.nan
```

Out[194…

```
nan
```

In [195…

```
np.pi # 3.141592653589793
```

Out[195…

```
3.141592653589793
```

In [196…

```
# Euler’s constant, base of natural logarithms
# Napier’s constant(蘇格蘭數學家約翰·納皮爾)
np.e # 2.718281828459045
```

Out[196…

```
2.718281828459045
```

In [197…

```
# 三角函數
# sin(30度) = sin(pi/6) = 0.5
# sin(45度) = sqrt(2)/2 = 0.707
# sin(60度) = sqrt(3)/2 = 0.866
# sin(90度) = 1
a = np.array([30, 45, 60, 90])
np.sin(a*np.pi/180)
```

Out[197…

```
array([0.5       , 0.70710678, 0.8660254 , 1.        ])
```

In [198…

```
# 亂數
import numpy as np

np.random.seed(123) # 設定亂數種子, 須輸入 >= 1 的整數

# random 產生0.0~1.0之間的1個亂數
x1 = np.random.random()
print(x1)
```

```
0.6964691855978616
```

In [199…

```
# random 產生0.0~1.0之間的3個亂數
x2 = np.random.random(3)
print(x2)
```

```
[0.28613933 0.22685145 0.55131477]
```

In [200…

```
# rand 產生0.0~1.0之間的1個亂數
x3 = np.random.rand()
print(x3)
```

```
0.7194689697855631
```

In [201…
```
# rand 產生0.0~1.0之間的3個亂數
x4 = np.random.rand(3)
print(x4)
```
```
[0.42310646 0.9807642  0.68482974]
```

In [202…
```
# rand(row, column) 產生亂數值陣列
x5 = np.random.rand(3, 2) # 3列,2行
print(x5)
```
```
[[0.4809319  0.39211752]
 [0.34317802 0.72904971]
 [0.43857224 0.0596779 ]]
```

In [203…
```
# randint 產生 min 與 max 之間的整數亂數,不包括max
# randint(max, size)

# 建立 5~10之間的1個整數亂數
x6 = np.random.randint(5, 10)
print(x6)
```
```
5
```

In [204…
```
# randint(min, max, size), min <= x < max

# 建立 1~11之間的10個整數亂數
x7 = np.random.randint(1, 11, size=10)
print(x7)
```
```
[1 5 2 8 4 3 5 8 3 5]
```

In [205…
```
# 建立 1~11之間的4列5行陣列的整數亂數
x8 = np.random.randint(1, 11, size=(4, 5))
print(x8)
```
```
[[ 9  1  8 10  4]
 [ 5  7  2  6  7]
 [ 3  2  9  4  6]
 [ 1  3  7  3  5]]
```

In [206…
```
# 標準常態分配隨機樣本
# https://numpy.org/doc/stable/reference/random/generator.html

from numpy import random

# 舊版用法
vals = random.standard_normal(3)
print(vals)
```
```
[0.29822755 0.46437133 0.11822163]
```

In [207…
```
more_vals = random.standard_normal(3)
print(more_vals)
```
```
[ 1.94369786  2.42320729 -1.26530807]
```

In [208…
```
# 新版用法
from numpy.random import default_rng

rng = default_rng()
vals = rng.standard_normal(3)
print(vals)
```

```
[-0.37821592 -1.50330268  0.06131135]
```

In [209…

```
more_vals = rng.standard_normal(3)
print(more_vals)
```

```
[-0.68362479 -1.1493593  -0.19296574]
```

In [210…

```
# 陣列的屬性

import numpy as np

a = np.array([0,1,2,3,4,5])
a
```

Out[210…

```
array([0, 1, 2, 3, 4, 5])
```

In [211…

```
a.dtype    # dtype('int32')
```

Out[211…

```
dtype('int32')
```

In [212…

```
a.size     # 6
```

Out[212…

```
6
```

In [213…

```
a.ndim     # 1
```

Out[213…

```
1
```

In [214…

```
a.shape    # (6,)
```

Out[214…

```
(6,)
```

In [215…

```
a.itemsize # 4 bytes
```

Out[215…

```
4
```

In [216…

```
a.nbytes   # 24
```

Out[216…

```
24
```

In [217…

```
b = np.array([[1,2,3,4], [4,5,6,7], [7,8,9,10.]])
b
```

Out[217…

```
array([[ 1.,  2.,  3.,  4.],
       [ 4.,  5.,  6.,  7.],
       [ 7.,  8.,  9., 10.]])
```

In [218…

```
b.dtype    # float64
```

Out[218…

```
dtype('float64')
```

In [219…

```
b.size     # 12
```

Out[219…

```
12
```

In [220…

```
b.ndim     # 2
```

Out[220…

```
2
```

In [221…

```
b.shape    # (3, 4)
```

Out[221…

```
(3, 4)
```

In [222…

```
b.itemsize # 8
```

Out[222…

```
8
```

In [223…

```
b.nbytes   # 12*8=96
```

Out[223…

```
96
```

In [224…

```
# 資料型別轉換
b.astype('int32')
```

Out[224…

```
array([[ 1,  2,  3,  4],
       [ 4,  5,  6,  7],
       [ 7,  8,  9, 10]])
```

In [225…

```
b = b.astype('int32')
```

In [226…

```
b.dtype    # int32
```

Out[226…

```
dtype('int32')
```

In [227…

```
# array 一維陣列 - loop 處理

a = np.array([1,2,3,4])
a
```

Out[227…

```
array([1, 2, 3, 4])
```

In [228…

```
# 方法1. array - 取出元素, 使用for
for x in a:
  print(x)
```

```
1
2
3
4
```

In [229…

```
# 方法2. array - 取出元素, 使用while
i = 0
while i < len(a):
  print(a[i])
  i = i + 1
```

```
1
2
3
4
```

In [230…

```
# 方法3. array - 取出元素, 使用指標 range, len
for i in range(len(a)):
  print(a[i])
```

```
1
2
3
4
```

In [231…

```
# 方法4. array - 取出元素, 使用陣列包含法
[print(x) for x in a]
```

```
1
2
3
4
```

Out[231…

```
[None, None, None, None]
```

In [232…

```
# array 二維陣列 - Loop 處理

a = np.array([[1,2,3,4], [5,6,7,8]])
a
```

Out[232…

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [233…

```
for x in a:
  print(x)
```

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

In [234…

```
for x in a:
    for item in x:
        print(str(item) + "@", end = "*")
```

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

In [235…

```
# array 運算

a = np.array([1,2,3])
b = np.array([4,5,6])
```

In [236…

```
a+b # 加
```

Out[236…

```
array([5, 7, 9])
```

In [237…

```
a-b # 減
```

Out[237…

```
array([-3, -3, -3])
```

In [238…

```
a*b # 乘
```

Out[238…

```
array([ 4, 10, 18])
```

In [239…

```
a/b # 除
```

Out[239…

```
array([0.25, 0.4 , 0.5 ])
```

In [240…

```
# 矩陣相乘(dot)
a = np.array([[1,2],[3,4],[5,6]])
a
```

Out[240…
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [241…
```
b = np.array([[1,2],[3,4]])
b
```

Out[241…
```
array([[1, 2],
       [3, 4]])
```

In [242…
```
a.shape
```

Out[242…
```
(3, 2)
```

In [243…
```
b.shape
```

Out[243…
```
(2, 2)
```

In [244…
```
c = a.dot(b) # 矩陣相乘(dot)
c
```

Out[244…
```
array([[ 7, 10],
       [15, 22],
       [23, 34]])
```

In [245…
```
np.transpose(c) # 矩陣轉置
```

Out[245…
```
array([[ 7, 15, 23],
       [10, 22, 34]])
```

In [246…
```
c.T                # 矩陣轉置
```

Out[246…
```
array([[ 7, 15, 23],
       [10, 22, 34]])
```

In [247…
```
# inv() : 反矩陣,逆矩陣 (inverse matrix)
from numpy.linalg import inv

x = np.array([[1, 2], [3, 4]])

inv(x)
# array([[-2. ,  1. ],
#        [ 1.5, -0.5]])
```

Out[247…
```
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
```

In [248…
```
# 單位矩陣 (Identity matrix)
x.dot(inv(x))
# array([[1.00000000e+00, 1.11022302e-16],
#        [0.00000000e+00, 1.00000000e+00]])
```

Out[248…
```
array([[1.00000000e+00, 1.11022302e-16],
       [0.00000000e+00, 1.00000000e+00]])
```

In [249…
```
x.dot(inv(x)).round(1)
# array([[1., 0.],
#        [0., 1.]])
```

Out[249…

```
array([[1., 0.],
       [0., 1.]])
```

In [250…

```
# 計算矩陣行列式值 (determinant)
np.linalg.det(x)
# -2.0000000000000004
```

Out[250…

```
-2.0000000000000004
```

In [251…

```
# 計算方形矩陣的特徵值 (eigenvalue) 與特徵向量 (eigenvector)
np.linalg.eig(x)
```

Out[251…

```
EigResult(eigenvalues=array([-0.37228132,  5.37228132]), eigenvectors=array([[-
0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]]))
```

In [252…

```
# 陣列應用 -高維度影像: MNIST 手寫數字辨識資料集
# http://yann.lecun.com/exdb/mnist/

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# 方法1 回傳 Bunch 資料物件
# 原圖為28*28, 2維展開為1維 28*28=784

mnist_data = fetch_openml("mnist_784")
xdata = mnist_data["data"] # 70000*784
ydata = mnist_data["target"] # 70000

xdata.ndim  # 2
xdata.shape # (70000, 784)
type(xdata) # pandas.core.frame.DataFrame
xdata.dtypes
```

Out[252…

```
pixel1      int64
pixel2      int64
pixel3      int64
pixel4      int64
pixel5      int64
            ...
pixel780    int64
pixel781    int64
pixel782    int64
pixel783    int64
pixel784    int64
Length: 784, dtype: object
```

In [253…

```
# 方法2 直接回傳 X, y

# Load data from https://www.openml.org/d/554
X, y = fetch_openml('mnist_784', return_X_y=True)
# X : 70000*84
# y : 70000

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state=123,
                                                    test_size=10000)
```

```
type(X_train) # DataFrame (早期版本為 numpy.ndarray)

# 將 DataFrame 轉換成 array 物件
X_train = X_train.to_numpy()
y_train = y_train.to_numpy()

type(X_train) # numpy.ndarray
X_train.ndim  # 2
X_train.shape # (60000, 784)
X_train.dtype # dtype('float64')

type(y_train) # numpy.ndarray
y_train.ndim  # 1
y_train.shape # (60000,)
y_train.dtype # dtype('O'), 表示字串
```

Out[253…

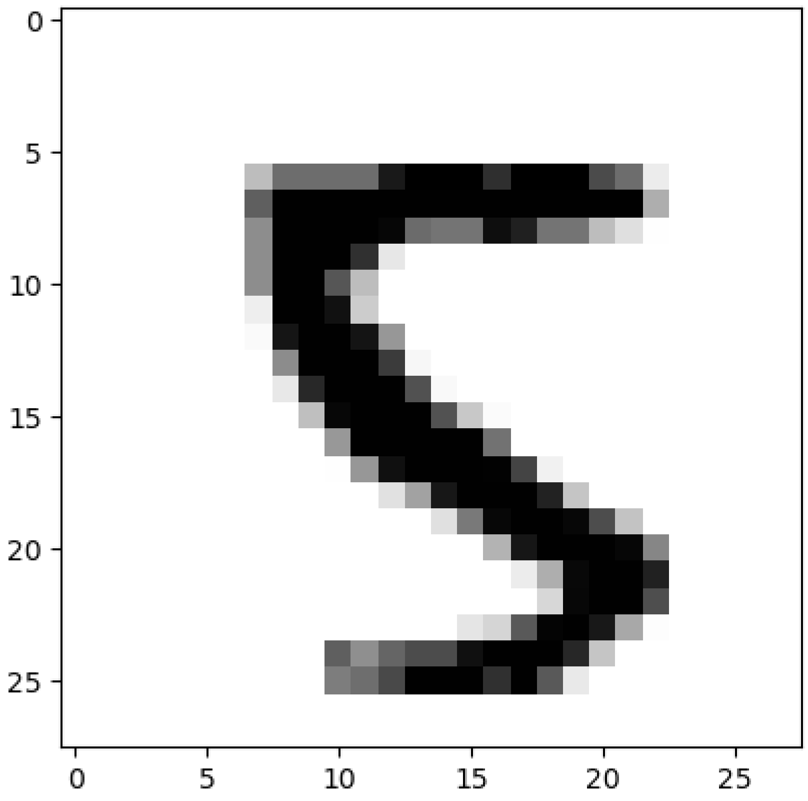
```
dtype('O')
```

In [254…

```
# 繪製數字影像
plt.imshow(X_train[0].reshape(28,28), cmap='binary')
```

Out[254…

```
<matplotlib.image.AxesImage at 0x23aa8bb7e60>
```



In [255…

```
# 實際值
y_train[0] # '5'
```

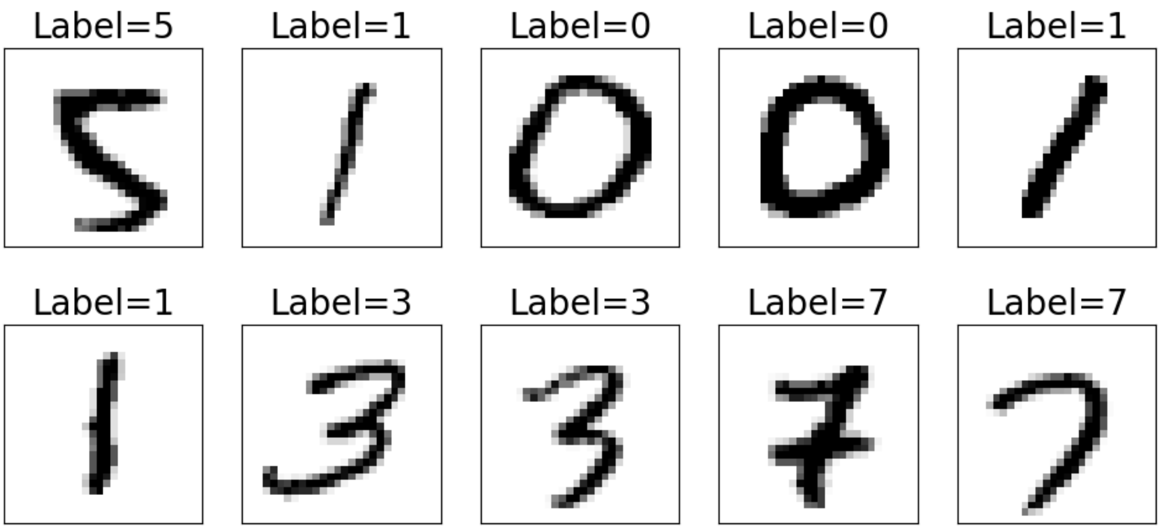
Out[255…

```
'5'
```

In [256…

```
# 繪製多個數字影像, 最多一次顯示25個
def plot_images_labels(images, labels, idx, num=10):
    fig = plt.gcf() # 取得目前的 figure
    fig.set_size_inches(12, 14) # 設定圖形大小
    if num > 25: num=25
    for i in range(0, num):
        ax=plt.subplot(5, 5, 1+i)
```

```
            ax.imshow(images[idx].reshape(28,28), cmap='binary')
            title= "Label=" + str(labels[idx])
            ax.set_title(title, fontsize=20)
            ax.set_xticks([])
            ax.set_yticks([])
            idx+=1
    plt.show()
plot_images_labels(X_train, y_train, 0, 10)
```



reshape 應用

In [257…

```
import numpy as np

z = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
z
```

Out[257…

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

In [258…

```
z.reshape(-1) # -1: unknown dimension
# array([ 1,  2,  3, ..., 10, 11, 12])
```

Out[258…

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

In [259…

```
z.reshape(-1,1) # row -1: unknown , column 1
```

Out[259…

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

In [260…

```
z.reshape(-1, 2) # row -1: unknown , column 2
```

Out[260…

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

In [261…

```
z.reshape(1,-1) # row 1 , column: unknown
```

Out[261…

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

In [262…

```
z.reshape(2, -1) # row 2 , column: unknown
```

Out[262…

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

In [263…

```
z.reshape(3, -1) # row 3 , column: unknown
```

Out[263…

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

In [264…

```
# ValueError: can only specify one unknown dimension
# z.reshape(-1, -1)
```

計算時間

In [265…

```
import timeit
import numpy as np

normal_py_sec = timeit.timeit('sum(x*x for x in range(1000))', number=10000)
naive_np_sec = timeit.timeit('sum(na*na)', setup="import numpy as np; na=np.aran
good_np_sec = timeit.timeit('na.dot(na)', setup="import numpy as np; na=np.arang

print("Normal Python: %f sec"%normal_py_sec)
print("Naive NumPy: %f sec"%naive_np_sec)
print("Good NumPy: %f sec"%good_np_sec)

# print "Hello World"    # python 2
print("Hello World")   # python 3
```

```
Normal Python: 0.718218 sec
Naive NumPy: 0.779572 sec
Good NumPy: 0.009574 sec
Hello World
```

判斷式 if elif else

In [266…

```
"""
# case 1
if 布林值:
        若布林值為 True，執行命令

# case 2
if 布林值:
        若布林值為 True，執行命令
else:
    若布林值為 False，執行命令
```

```
# case 3
if 布林值一:
        若布林值一為 True，執行命令
elif 布林值二:
        若布林值二為 True，執行命令
...
else:
        若布林值一和二...都是 False，執行命令
"""
```

Out[266…

```
'\n# case 1\nif 布林值:\n \t若布林值為 True，執行命令\n\n# case 2\nif 布林值:\n \t若布林值為 True，執行命令\nelse:\n     若布林值為 False，執行命令\n\n# case 3\nif 布林值一:\n \t若布林值一為 True，執行命令\nelif 布林值二:\n \t若布林值二為 True，執行命令\n...\nelse:\n \t若布林值一和二...都是 False，執行命令\n'
```

In [267…

```python
# elif敘述
a = '+'

if a == '+':
        op = 'PLUS'
elif a == '-':
        op = 'MINUS'
else:
        op = 'UNKNOWN'

op
```

Out[267…

```
'PLUS'
```

In [268…

```python
# 沒有像C語言一樣，有switch的語法
# 布林表示式 - and, or, not
a = 1
b = 6
c = 9

if b >= a and b <= c:
        print('b is between a and c')
```

```
b is between a and c
```

In [269…

```python
if not (b < a or c > c):
        print('b is still between a and c')
```

```
b is still between a and c
```

In [270…

```python
# 邏輯錯誤 (Logical Errors)
# if 範例 - age > 200 不會執行
name = 'RWEPA'
age = 300
if name == 'Alan':
    print('Hi, Alan.')
elif age < 20:
    print('You are not Alan.')
elif age > 100:
    print('You are not Alan. 大大')
elif age > 200:
    print('年齡異常')
# You are not Alan. 大大
```

```
You are not Alan. 大大
```

迴圈 (Loops)

In [271…

```
# while 迴圈
"""
name = ''
while name != 'Alan Lee':
    print('Please type your name.')
    name = input()
print('Thank you!')
"""
```

Out[271…

```
"\nname = ''\nwhile name != 'Alan Lee':\n    print('Please type your name.')\n
name = input()\nprint('Thank you!')\n"
```

In [272…

```
# while + break
"""
while True:
    print('Please type your name.')
    name = input()
    if name == 'Alan Lee':
        break
print('Thank you!')
"""
```

Out[272…

```
"\nwhile True:\n    print('Please type your name.')\n    name = input()\n    if
name == 'Alan Lee':\n        break\nprint('Thank you!')\n"
```

In [273…

```
# while + break + continue
"""
while True:
    print('Who are you?')
    name = input()
    if name != 'Alan':
        continue
    print('Hello, Alan. What is the password?')
    password = input()
    if password == 'alan9956@gmail.com':
        break
print('Access granted.')
"""
```

Out[273…

```
"\nwhile True:\n    print('Who are you?')\n    name = input()\n    if name !=
'Alan':\n        continue\n    print('Hello, Alan. What is the password?')\n
password = input()\n    if password == 'alan9956@gmail.com':\n        break\npr
int('Access granted.')\n"
```

In [274…

```
# 顯示list元素
for i in [3, 4, 10, 25]:
        print(i)
```

```
3
4
10
25
```

In [275…

```
# 顯示一個字元
for c in "Hello":
        print(c)
```

```
H
e
l
l
o
```

In [276…
```python
# 顯示 range 元素
for i in range(1, 4):
        print(i)
```

```
1
2
3
```

In [277…
```python
for i in range(4, -2, -1):
        print(i)
```

```
4
3
2
1
0
-1
```

In [278…
```python
# 零數值判斷, 注意以下結果
0 == False
```

Out[278…
```
True
```

In [279…
```python
0.0 == False
```

Out[279…
```
True
```

In [280…
```python
0.000 == False
```

Out[280…
```
True
```

In [281…
```python
'' == False
```

Out[281…
```
False
```

In [282…
```python
# 非零數值判斷
1 == True     # True
```

Out[282…
```
True
```

In [283…
```python
1.23 == True  # False
```

Out[283…
```
False
```

In [284…
```python
1.23 == False # False
```

Out[284…
```
False
```

檔案處理

In [285…
```python
# os模組-建立與切換工作目錄
import os
```

In [286…

```
dir = os.path.join("C:/mydata")
if not os.path.exists(dir):
    os.mkdir(dir)       # 建立目錄

os.chdir(dir)           # 變更工作目錄
os.listdir(os.getcwd()) # 顯示檔案名稱
```

Out[286…

```
['coding.dat', 'output.txt']
```

In [287…

```
os.getcwd()             # 已經變更為 C:/mydata
```

Out[287…

```
'C:\\mydata'
```

In [288…

```
# 方法1.檔案的開啟/寫入/關閉
f = open("coding.dat", "w") # Open a file for writing
f.write("Hello World\n")
f.write("Python\n")
f.write("R\n")
f.write("SQL\n")
f.write("Excel VBA\n")
f.close()
```

In [289…

```
g = open("coding.dat","a")      # Open a file for appending
g.write(".NET")
g.close()
```

In [290…

```
# 方法2.使用 with區塊

# with open("coding.dat", "r") as infile:

# with區塊特性
# 檔案會自動關閉, 可以不用撰寫 .close()
# 即使出現以下狀況, 檔案仍會自動關閉:
#  (1)發生例外 (Exception)
#  (2)執行 return, continue, break 等而跳出 with 區塊

# read 讀取全部資料
with open("coding.dat", "r") as infile:
    mydata = infile.read()
    print(type(mydata)) # str
    print(mydata)
```

```
<class 'str'>
Hello World
Python
R
SQL
Excel VBA
.NET
```

In [291…

```
# readline 一次讀一列資料, while 迴圈-預設加入分隔列
with open("coding.dat", "r") as infile:
    while True:
        line = infile.readline()  # 一次讀一列資料
        if not line:              # 所有資料讀取完畢
            break
        print(line)               # 預設加入分隔列
```

```
Hello World

Python

R

SQL

Excel VBA

.NET
```

In [292…

```python
# readline 一次讀一列資料, while 迴圈自訂分隔列符號
with open("coding.dat", "r") as infile:
    while True:
        line = infile.readline()     # 一次讀一列資料
        if not line:                 # 所有資料讀取完畢
            break
        print(line, end='*')          # end='*' 自訂分隔列符號
```

```
Hello World
*Python
*R
*SQL
*Excel VBA
*.NET*
```

In [293…

```python
# readlines 一次讀取所有資料
with open("coding.dat", "r") as infile:
    for line in infile.readlines():  # 一次讀取所有資料，再逐列處理
        print(line, end='')
```

```
Hello World
Python
R
SQL
Excel VBA
.NET
```

In [294…

```python
# readlines 簡化版本
with open("coding.dat", "r") as infile:
    for line in infile:
        print(line, end='')
```

```
Hello World
Python
R
SQL
Excel VBA
.NET
```

3.5 日期時間資料

In [295…

```python
# 使用 datetime 模組
from datetime import date, time, datetime
```

In [296…

```python
date(year=2021, month=8, day=10) # datetime.date(2021, 8, 10)
```

Out[296…
```
datetime.date(2021, 8, 10)
```

In [297…
```
time(hour=13, minute=30, second=31) # datetime.time(13, 30, 31)
```

Out[297…
```
datetime.time(13, 30, 31)
```

In [298…
```
datetime(year=2021, month=8, day=10, hour=13, minute=30, second=31)
# datetime.datetime(2021, 8, 10, 13, 30, 31)
```

Out[298…
```
datetime.datetime(2021, 8, 10, 13, 30, 31)
```

In [299…
```
# 現在日期,時間
today = date.today()
today
```

Out[299…
```
datetime.date(2025, 1, 6)
```

In [300…
```
now = datetime.now()
now
```

Out[300…
```
datetime.datetime(2025, 1, 6, 23, 32, 36, 462463)
```

In [301…
```
current_time = time(now.hour, now.minute, now.second)
current_time
```

Out[301…
```
datetime.time(23, 32, 36)
```

In [302…
```
datetime.combine(today, current_time)
```

Out[302…
```
datetime.datetime(2025, 1, 6, 23, 32, 36)
```

In [303…
```
# 字串轉換為日期-fromisoformat
mystr = "2021-07-21"
mydate = date.fromisoformat(mystr)
mydate
print(mydate)
```

```
2021-07-21
```

In [304…
```
# 字串轉換為日期-strptime
# https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior

# Year    %Y (4位數值年)
# Month   %m (2位數值月)
# Date    %d (2位數字日)
# Hour    %H (2位數字24小時的時)
# Minute  %M (2位數字分)
# Second  %S (2位數字秒)

date_string = "06-30-2021 12:34:56"
format_string = "%m-%d-%Y %H:%M:%S"
datetime.strptime(date_string, format_string)
# datetime.datetime(2021, 6, 30, 12, 34, 56)
```

Out[304…
```
datetime.datetime(2021, 6, 30, 12, 34, 56)
```

In [305…
```
# 範例-日期計算
PYCON_DATE = datetime(year=2025, month=5, day=14, hour=8)
```

```
countdown = PYCON_DATE - datetime.now()
type(countdown) # datetime.timedelta
countdown
countdownDay = countdown.days

txt = "距離 2025年5月14日 USA PyCon 還有 {} 天"
print(txt.format(countdownDay))
```

```
距離 2025年5月14日 USA PyCon 還有 127 天
```

In [306…

```
# Time Zones 時區 - 使用 dateutil 模組
# https://dateutil.readthedocs.io/en/stable/

from dateutil import tz
from datetime import datetime

now = datetime.now(tz=tz.tzlocal())
now
```

Out[306…

```
datetime.datetime(2025, 1, 6, 23, 32, 36, 541458, tzinfo=tzlocal())
```

In [307…

```
# 範例-計算程式執行時間
from datetime import datetime
from numpy.random import default_rng

# 開始計算時間
starttime = datetime.now()
print(starttime)

# 程式執行
rng = default_rng()
vals = []
x = abs(rng.standard_normal(100000000))
x[0:3]
vals = x**0.5
vals[0:3]

# 結束時間
endtime = datetime.now()
print(endtime)

# 程式執行時間
print(endtime - starttime)
```

```
2025-01-06 23:32:36.553592
2025-01-06 23:32:38.451952
0:00:01.898360
```

In [308…

```
# 實作練習
# 檔案日期時間處理
# https://www.kaggle.com/shawon10/web-log-dataset
# 檔案名稱: weblog.csv
# 欄位個數:4
# 資料筆數:16007
# IP     Time     URL      Staus
# 10.128.2.1     [29/Nov/2017:06:58:55   GET /login.php HTTP/1.1 200
# 10.128.2.1     [29/Nov/2017:06:59:02   POST /process.php HTTP/1.1      302
# 10.128.2.1     [29/Nov/2017:06:59:03   GET /home.php HTTP/1.1   200

# 下載 https://github.com/rwepa/DataDemo/blob/master/weblog.csv
# 練習使用 open, read, datetime, re 等處理技術(不可使用 pandas),計算下列3個時段的資
# 06:00-14:00, 14:00-22:00, 22:00-06:00
```

In [309…

```
# end
```