Python Note

指導老師：蘇有 老師

作者：鄭羽晴 A10423014@gmail.com

1. Python 6 大型別
2. 運算子優先順序
3. String
4. List
5. Tuple
6. Dict
7. Set
8. For 迴圈
9. if判斷式
10. 例外處理Try Except
11. 正則運算Regular
12. BIF(Built In Function)
    - zip() & enumerate()
    - input() & eval()
    - isinstance() & format
    - map() & filter()
13. 自訂函式
14. 模組
    - datetime
    - random
    - openpyxl
    - pytube
    - os
    - subprocess
    - numpy
    - pandas
    - Xml.etree.ElementTree
    - matplotlib
    - yfinance & mplfinance
    - request

- pandas
  - Series
  - DataFrame
    - 資料拿取df.loc[]
    - 資料拿取df.iloc[](用index)
    - 操作列索引(index)
    - 操作行欄位(columns)
    - 行欄位(columns)重新安排、索引重塑
    - rank排序
    - 更改資料
    - 重複資料處理
    - 處理空格資料
    - 集合運算
    - 資料區間切分
    - pivot(Pandas的樞紐分析:寬表格變為長表格)
    - melt(取消樞紐分析:長表格變為寬表格)
    - apply()
    - 檔案讀取

Python 6 type

1. Number(數字immutable
   - int 整數
   - float 浮點數
2. String(字串immutable
   - 新增、修改、刪除、查詢
   - 函式
   - 索引與切片

Collections集合行資料型態

1. List (串列， 資料有先後順序、內容可改變、允許有重複值
   - 新增、修改、刪除、查詢
   - 函式
   - 索引與切片
2. Tuple(元組immutable， 資料有先後順序、內容不可改變、允許有重複值
   - 新增、修改、刪除、查詢
   - 函式
   - 索引與切片
3. Dict(字典， 資料有先後順序、內容可改變、不允許有重複值
   - 新增、修改、刪除、查詢
   - 函式
   - 索引與切片
4. Set(集合， 資料沒有先後順序、內容不可改變、不允許有重複值、但可隨時新增刪除資料
   - 新增、刪除
   - 交集、差集、聯集

變數

賦值 變數

The equal sign (=) is used to assign a value to a variable. Afterwards, no result is displayed before the next interactive prompt:

= 把右邊運算式的答案存到左邊的變數

1. 任何Unicode 字元都可以拿來當Python 的識別字名稱，習慣上仍是以英文二十六個字母大小寫、數字與底線為主。 2. 數字不能當變數的起始字元。 3. 變數及函數名稱通常會用有意義的小寫英文單字，或是多個英文單字的組合。 4. 類別名稱通常會採大寫駝峰型，方法及屬性則較常採小寫駝峰型，或用底線連接每個單字。5.Keyword不可使用6.內建函式(BIF)盡量不要使用，要使用建議加上底線

Keywords

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

Built-in Functions

A	E	L	R
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	**F**	**M**	
ascii()	filter()	map()	**S**
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	**G**	**N**	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	**H**	object()	
callable()	hasattr()	oct()	**T**
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		**P**	**V**
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	**Z**
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

運算子優先順序

運算式：Expression
運算子：Operator
運算元：Operand
除法：Division
整數除法：Floor Division

1.算術運算子

2.關係運算子(產生條件，傳出真True/假False)

3.邏輯運算子

Operator	Description
`(expressions...)`, `[expressions...]`, `{key: value...}`, `{expressions...}` ()括號最優先	Binding or parenthesized expression, list display, dictionary display, set display
`x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`	Subscription, slicing, call, attribute reference
`await x`	Await expression
`**` 次方	Exponentiation [5]
`+x`, `-x`, `~x`	Positive, negative, bitwise NOT
`*`, `@`, `/`, `//`, `%` //除法只取整數； % 取餘數 /除法後會自動變浮點數	Multiplication, matrix multiplication, division, floor division, remainder [6]
`+`, `-`	Addition and subtraction
`<<`, `>>`	Shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`in`, `not in`, `is`, `is not`, `<`, `<=`, `>`, `>=`, `!=`, `==`	Comparisons, including membership tests and identity tests
`not x`	Boolean NOT
`and`	Boolean AND
`or`	Boolean OR
`if – else`	Conditional expression
`lambda`	Lambda expression
`:=`	Assignment expression

一元運算子ex：Not (後面有運算元，前面沒有
二元運算子ex：/ (前後都需要有運算元
三元運算子ex：if else (一個運算式裡一定要有三個運算元

※其中一個數值為浮點數，則運算結果為浮點數
※次方(**)的運算順序為右邊優先(2**3**2會先計算3**2=9，再計算2**9=512)

字串String

1. 可使用前後一個單引號’，前後一個雙引號”，前後三個單引號’’’(多行文字使用)
2. 如需將文字中的單/雙引號列為字串的一部份，可在單/雙引號前加上\符號(Escape 跳脫字元)　‘doesn\’t’輸出”doesn’t”
3. Raw Strings = 在字串前加上r，關閉所有跳脫功能 print(r‘C:\some\name’)輸出 ‘C:\some\name’
4. 字串的運算只有+跟*，+為串接(Concatenated)，*為複製(3*’un’+’ium’ 輸出 ‘ununununium’)
5. 字串可執行索引(Index)與切片(Slice)
6. 索引從左開始為0,1,2…，也可從右邊開始，起始值則為-1,-2,-3…
7. 切片的最後一索引不會被印出 txt=‘Python’ txt[0:3] = ‘Pyt’，如要取到最後一值，不需打後方數字txt[1:]，從頭則不須開頭txt[:2]
8. 切片用法 字串[起始位置:終點位置:跳格] txt=‘鄭xx王xx林xx陳xx’ txt[::3]=‘鄭王林陳’
9. 字串是不可變(Immutable)資料型別，無法變更其元素(Item)，只能重新指派

Escape Sequence	Meaning	Notes
\newline	Backslash and newline ignored	
\\	Backslash (\)	
\'	Single quote (')	
\"	Double quote (")	
\a	ASCII Bell (BEL)	
\b	ASCII Backspace (BS)	
\f	ASCII Formfeed (FF)	
\n	ASCII Linefeed (LF) New Line	
\r	ASCII Carriage Return (CR) 回到本行行首	
\t	ASCII Horizontal Tab (TAB)	
\v	ASCII Vertical Tab (VT)	
\ooo	Character with octal value *ooo*	(1,3)
\xhh	Character with hex value *hh*	(2,3)

```
>>> print('abc\red\rg')
gdc
```

印出abc時遇到\r，所以游標會回到a前面繼續輸出ed，就會將ab覆蓋，此時輸出結果為edc，往後遇到\r，游標又回到e前，再輸出g，將e覆蓋，故結果為gdc

字串String方法

常用

Method	Description	Method	Description
capitalize()	Converts the first character to upper case	join()	Joins the elements of an iterable to the end of the string
casefold()	Converts string into lower case	ljust()	Returns a left justified version of the string
center()	Returns a centered string	lower()	Converts a string into lower case
count()	Returns the number of times a specified value occurs in a string	lstrip()	Returns a left trim version of the string
encode()	Returns an encoded version of the string	maketrans()	Returns a translation table to be used in translations
endswith()	Returns true if the string ends with the specified value	partition()	Returns a tuple where the string is parted into three parts
expandtabs()	Sets the tab size of the string	replace()	Returns a string where a specified value is replaced with a specified value
find()	Searches the string for a specified value and returns the position of where it was found	rfind()	Searches the string for a specified value and returns the last position of where it was found
format()	Formats specified values in a string	rindex()	Searches the string for a specified value and returns the last position of where it was found
format_map()	Formats specified values in a string	rjust()	Returns a right justified version of the string
index()	Searches the string for a specified value and returns the position of where it was found	rpartition()	Returns a tuple where the string is parted into three parts
isalnum()	Returns True if all characters in the string are alphanumeric	rsplit()	Splits the string at the specified separator, and returns a list
isalpha()	Returns True if all characters in the string are in the alphabet	rstrip()	Returns a right trim version of the string
isdecimal()	Returns True if all characters in the string are decimals	split()	Splits the string at the specified separator, and returns a list 用指定的字元分開
isdigit()	Returns True if all characters in the string are digits	splitlines()	Splits the string at line breaks and returns a list
isidentifier()	Returns True if the string is an identifier	startswith()	Returns true if the string starts with the specified value
islower()	Returns True if all characters in the string are lower case	strip()	Returns a trimmed version of the string 去頭去尾
isnumeric()	Returns True if all characters in the string are numeric	swapcase()	Swaps cases, lower case becomes upper case and vice versa
isprintable()	Returns True if all characters in the string are printable	title()	Converts the first character of each word to upper case
isspace()	Returns True if all characters in the string are whitespaces	translate()	Returns a translated string
istitle()	Returns True if the string follows the rules of a title	upper()	Converts a string into upper case
isupper()	Returns True if all characters in the string are upper case	zfill()	Fills the string with a specified number of 0 values at the beginning

參考網址：https://www.w3schools.com/python/python_strings_methods.asp

字串String – String Formatting(舊式寫法)

規則：[Flag] [n] [Conversion]

```
>>> print('%(language)s has %(number)03d quote types.' %
...       {'language': "Python", "number": 2})
Python has 002 quote types.
```

Flag	Meaning
`'#'`	The value conversion will use the “alternate form” (where defined below).
`'0'`	The conversion will be zero padded for numeric values.
`'-'`	The converted value is left adjusted (overrides the `'0'` conversion if both are given).
`' '`	(a space) A blank should be left before a positive number (or empty string) produced by a signed conversion.
`'+'`	A sign character (`'+'` or `'-'`) will precede the conversion (overrides a “space” flag).

Conversion	Meaning	Notes
`'d'`	Signed integer decimal.	
`'i'`	Signed integer decimal.	
`'o'`	Signed octal value.	(1)
`'u'`	Obsolete type – it is identical to `'d'`.	(6)
`'x'`	Signed hexadecimal (lowercase).	(2)
`'X'`	Signed hexadecimal (uppercase).	(2)
`'e'`	Floating point exponential format (lowercase).	(3)
`'E'`	Floating point exponential format (uppercase).	(3)
`'f'`	Floating point decimal format.	(3)
`'F'`	Floating point decimal format.	(3)
`'g'`	Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.	(4)
`'G'`	Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.	(4)
`'c'`	Single character (accepts integer or single character string).	
`'r'`	String (converts any Python object using `repr()`).	(5)
`'s'`	String (converts any Python object using `str()`).	(5)
`'a'`	String (converts any Python object using `ascii()`).	(5)
`'%'`	No argument is converted, results in a `'%'` character in the result.	

字串String – Format String

規則："{" [field_name] ["!" conversion] [":" format_spec] "}“.format()

```
format_spec      ::=  [[fill]align][sign][#][0][width][grouping_option][.precision][type]
fill             ::=  <any character>
align            ::=  "<" | ">" | "=" | "^"
sign             ::=  "+" | "-" | " "
width            ::=  digit+
grouping_option  ::=  "_" | ","
precision        ::=  digit+
type             ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"
```

Align	Meaning
`'<'`	Forces the field to be left-aligned within the available space (this is the default for most objects).
`'>'`	Forces the field to be right-aligned within the available space (this is the default for numbers).
`'='`	Forces the padding to be placed after the sign (if any) but before the digits. This is used for printing fields in the form '+000000120'. This alignment option is only valid for numeric types. It becomes the default for numbers when '0' immediately precedes the field width.
`'^'`	Forces the field to be centered within the available space.

Type	Meaning
`'b'`	Binary format. Outputs the number in base 2.
`'c'`	Character. Converts the integer to the corresponding unicode character before printing.
`'d'`	Decimal Integer. Outputs the number in base 10.
`'o'`	Octal format. Outputs the number in base 8.
`'x'`	Hex format. Outputs the number in base 16, using lower-case letters for the digits above 9.
`'X'`	Hex format. Outputs the number in base 16, using upper-case letters for the digits above 9. In case `'#'` is specified, the prefix `'0x'` will be upper-cased to `'0X'` as well.
`'n'`	Number. This is the same as `'d'`, except that it uses the current locale setting to insert the appropriate number separator characters.

Option	Meaning
`'+'`	indicates that a sign should be used for both positive as well as negative numbers.
`'-'`	indicates that a sign should be used only for negative numbers (this is the default behavior).
space	indicates that a leading space should be used on positive numbers, and a minus sign on negative numbers.

```
"{:*^10s}{:15,d}".format('abc',  87654321)
```

```
'***abc****      87,654,321'
```

```
"{:*^10s}{:15,d}{:010.2f}".format('abc',  87654321,5)
```

```
'***abc****      87,654,3210000005.00'
```

```
"{2:*^10s}{0:15,d}{1:10.2f}".format(87654321,5,  'abc')
```

```
'***abc****      87,654,321      5.00'
```

串列List

集合型資料型態collections of data

1. 可迭代物件，用中括弧[ ]包著，可以儲存很多元素(所有Python型態都可以)，元素可以重複，可使用切片及索引

2. list1 = [“apple”, “banana”, “cherry”]，list1[1:2] = [“blackcurrant”, “watermelon”]，結果除了會將索引1的banana改成blackcurrant還會再新增一筆watermelon的資料
   print(list1) = [“apple”, “blackcurrant”, “watermelon”, “cherry”]

3. list[1] = [“blackcurrant”, “watermelon”]會直接將索引1的banana取代成[“blackcurrant”, “watermelon”]，
   print(list1) = [“apple”, [“blackcurrant”, “watermelon”], “cherry”]

5. 可用list()函式將物件強迫轉型為list，非iterable(可迭代物件=可以把資料一個一個拿出來的物件)會出錯

```
print(list('Python'))
print(list('你好嗎'))
print(list(1,2,3))
```

```
    print(list(123))
TypeError: 'int' object is not iterable
['P', 'y', 't', 'h', 'o', 'n']
['你', '好', '嗎']
```

串列List

集合型資料型態collections of data

list.sort()跟sorted(list)都可以做排序，建議使用sorted(list)：

list.sort()排完後不會回傳值且會改變原來串列內容

sorted(list)則可存於另一變數中，不改變原來串列值

串列List方法

Method	Description
append()	Adds an element at the end of the list `list.append(x)` Equivalent to `a[len(a):] = [x]`
clear()	Removes all the elements from the list `list.clear()` Equivalent to `del a[:]`
copy()	Returns a copy of the list `list.copy()`
count()	Returns the number of elements with the specified value `list.count(x)`
extend()	Add the elements of a list (or any iterable), to the end of the current list `list.extend(iterable)` Equivalent to `a[len(a):] = iterable` lista.extend(listb)跟lista + listb 有相同效果
index()	Returns the index of the first element with the specified value `list.index(x[, start[, end]])`
insert()	Adds an element at the specified position `list.insert(i, x)` `a.insert(len(a), x)` is equivalent to `a.append(x)`
pop()	Removes the element at the specified position `list.pop([i])` If no index is specified, a.pop() removes and returns the last item in the list.
remove()	Removes the item with the specified value `list.remove(x)` It raises a `ValueError` if there is no such item.
reverse()	Reverses the order of the list `list.reverse()`
sort()	Sorts the list `list.sort(*, key=None, reverse=False)`

`list.pop([i])` 中括弧代表Option，可寫可不寫

參考網址：https://www.w3schools.com/python/python_lists_methods.asp

串列List語法糖Syntactic sugar(只有Python可以用的語法)

1. 串列生成式：[運算式 for ... in ...]

單一選擇

```
newlist = [expression for item in iterable if condition == True]
```

```
[n**2 for n in range(5)]
```

```
[0, 1, 4, 9, 16]
```

```
[ 9-abs(n) for n in range(-8,9,1)]
```

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

```
["你好" for n in range(5)]
```

```
['你好', '你好', '你好', '你好', '你好']
```

2. 三元運算子的語法 [condition_is_true if condition else condition_is_false for... in ... if...]

```
newlist = ['陳xx' if item[0]=='陳' else item for item in list1]
print(newlist)
```

```
['邱丞宏', '葛育坤', '何欣惠', '李文賜', '傅采穎', '吳宜芳', '萬晉源', '蔡維章', '湯均尉', '謝博名', '余乾中', '楊子寬', '黃詩婷', '王品翔', '鮑湘鈺', '鄭羽晴', '陳xx', '劉禮銓', '呂杰祐', '符采文', '何小君', '陳xx', '陳xx', '容蓉', '李岱沅', '盧羿樺']
```

參考網址：https://www.w3schools.com/python/python_lists_methods.asp

串列List語法糖Syntactic sugar(只有Python可以用的語法)

```
keylist = list(list1[0].keys())
linelist = [[] for n in range(len(keylist))]
totalJan = 0
linelist = [[keylist[item]] for item in range(len(keylist))]
#for item in range(len(keylist)):
    #linelist[item].append(keylist[item])
[[keylist[i]]+[item[keylist[i]] for item in list1 if item['期別'][:3] == '108'] for i in range(len(keylist))]
#for item in list1:
   # if item['期別'][:3] == '108':
       # for i in range(len(keylist)):
           # linelist[i].append(item[keylist[i]])
```


```
[
  {
    "期別": "108年1月",
    "十八尖山": "138881",
    "青青草原": "22226",
    "城隍廟": "186932",
    "新竹漁港": "226935",
    "賞蟹步道": "8574",
    "青草湖": "14436",
    "十七公里腳踏車道": "7538",
    "新竹公園": "16902"
  },
  {
    "期別": "108年2月",
    "十八尖山": "137512",
    "青青草原": "40034",
    "城隍廟": "251404",
    "新竹漁港": "219437",
    "賞蟹步道": "20629",
    "青草湖": "16169",
    "十七公里腳踏車道": "17081",
    "新竹公園": "42939"
  },
```


原list1資料

linelist.txt


```
[['期別','108年1月','108年2月','108年3月','108年4月','108年5月','108年6月','108年7月','108年8月','108年9月','108年10月','108年11月','108年12月'],
 ['十八尖山','138881','137512','101301','186020','160695','169084','144669','135557','127176','180991','180144','147301'],
 ['青青草原','22226','40034','32227','64880','30172','31216','22904','26594','20196','28184','30951','35819'],
 ['城隍廟','186932','251404','157317','205830','212146','187465','198434','231224','195564','191720','216609','278915'],
 ['新竹漁港','226935','219437','241011','272600','309255','331952','244742','355109','383046','361919','388662','324479'],
 ['賞蟹步道','8574','20629','20187','29660','24648','28083','17773','39719','18363','22722','9780','10636'],
 ['青草湖','14436','16169','12116','17830','15267','20814','11831','11184','10686','23675','18252','24947'],
 ['十七公里腳踏車道','7538','17081','8621','18440','26268','25789','14751','22155','12789','23162','22773','16739'],
 ['新竹公園','16902','42939','26910','26210','21465','20129','15135','31347','16797','20279','20274','105026']]
```


參考網址：https://www.w3schools.com/python/python_lists_methods.asp

元組Tuple

1. 使用小括弧( )包覆
2. 可使用切片及索引
3. 不可新增、刪除、修改，只能查詢
4. 元組是不可變(Immutable)資料型別，無法變更其元素(Item)，只能重新指派
5. 元組單一元素須加上逗號，否則會被誤認為字串

```
t = ("text")
t1 = ("text",)
print(type(t))
print(type(t1))
```

```
<class 'str'>
<class 'tuple'>
```

6. Unpack 解包

```
a,b,c=1,2,3 #多重指派 先打包 再解包， 等式左右兩邊變數數量與元素數量需相等
print(a,b,c)
```

```
1 [2, 3] 4
```

```
a,b,c=1,*(2,3) #強制解包加上"*"
print(a,b,c)
```

```
1 2 3
```

7. Pack打包

```
t = 1,2,3
t
```

```
(1, 2, 3)
```

```
a,*b,c=1,2,3,4 #強制打包成"串列"
print(a,b,c)
```

```
1 [2, 3] 4
```

```
a,*b,c=1,2 #加上*可以讓等號左右兩邊數量不相等
print(a,b,c)
```

```
1 [] 2 電腦會自動打包成空字串
```

```
a=100
b=200
a,b =b,a #交換a,b的值，先將ab打包成(100,200)
         #再重新指派(200,100)給a,b
print(a,b)
```

```
200 100
```

字典Dict

1. 字典內的元素需使用Key:Value方式編寫
2. 任何物件皆可以當字典的Value，Key值只能是Numbers ,Strings ,Tuple
3. 使用大括弧包覆{}
4. Key值不能重複，重複的話前面的值會被覆蓋
5. 無切片及索引
6. 可動態新增資料 dict[newkey] = Value
7. Dictiionary.setdefault(keyname,value)如果keyname已存在，就回傳值，若不存在就新增，value未填寫預設為None
8. Sorted(字典)只會依照Key值排序
9. 依照Value值排序 sorted(d.items(), key=lambda x: (x[1],x[0]), reverse=True)

d.items() ➔ 先將字典的資料(key:value)打包,((key1,value1), (key2,value2).....)，再傳給後方的運算式

key=lambda x: (x[1],x[0])➔lambda匿名函式 參數(可以很多個) ：要做的事(只能做一件事) #一行函式的概念
➔(x[1],x[0])就是要回傳的值，如果x[1]相同的話就依照x[0]去排序(藉此可達到多欄排序功能)

字典統計次數好用!!!

```
#1 到 10的數字隨便出20次，統計每個號碼出現幾次
import random
import pandas as pd
ls1 =[random.randint(1,10) for n in range(20)]

dict1 = { n+1:0 for n in range(10)}
for item in ls1:
  dict1[item] += 1

sortlist = sorted(dict1.items(), key = lambda x:(x[1],x[0]), reverse = True) #先照出現次數大到小排序，如相同再依照Key值大到小排序
sortlist1 = sorted(dict1.items(), key = lambda x:(x[1]), reverse = True)
print('list1: ',ls1)
print('多欄: ',sortlist)
print('單欄排序: ',sortlist1)
```

```
list1:  [4, 5, 10, 1, 2, 5, 9, 10, 9, 10, 7, 5, 9, 6, 10, 3, 10, 6, 3, 5]
多欄:  [(10, 5), (5, 4), (9, 3), (6, 2), (3, 2), (7, 1), (4, 1), (2, 1), (1, 1), (8, 0)]
單欄排序:  [(10, 5), (5, 4), (9, 3), (3, 2), (6, 2), (1, 1), (2, 1), (4, 1), (7, 1), (8, 0)]
```

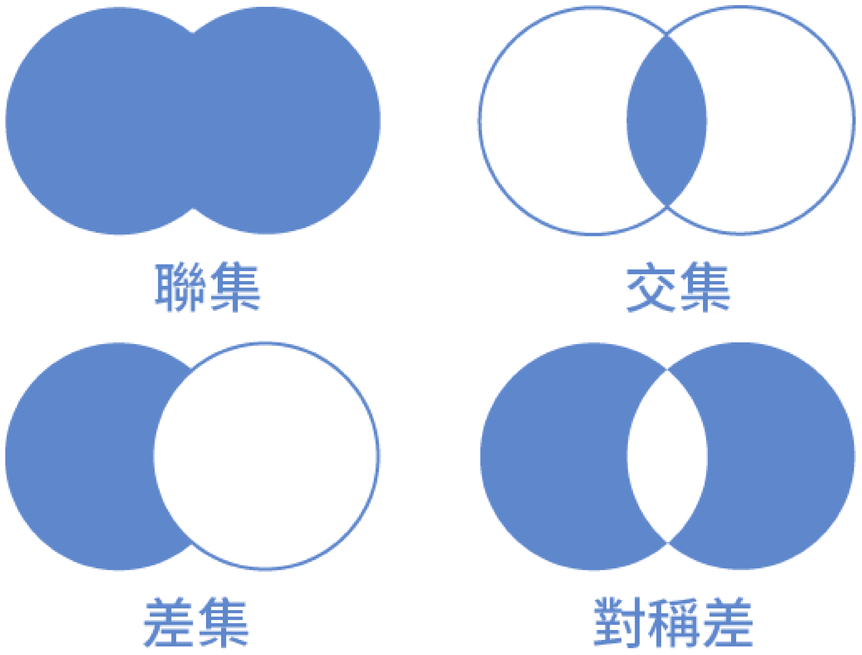
字典Dict方法

常用於處理集合資料

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value `dict.fromkeys(keys, value)`
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

組合Set

1. 使用大括弧{}包覆(跟字典一樣，但沒有key)
2. Set資料不可重複(如有重複值只會保留第一個)，值沒有順序概念，沒辦法使用索引 (index)來印出
3. 因值不可重複，可將資料型態轉型為set來去除重複資料
4. 可新增(add)刪除(remove)資料
5. set1 & set2 ➔兩個Set的交集資料，同set1.intersection(set2)
6. set1 | set2 ➔兩個Set的聯集資料，同set1.union(set2)
7. set1 - set2➔set1差集set2，同set1.difference(set2)，差集有前後關係，set1跟set2寫相反會得到不同的結果
8. set1 ^ set2➔對稱差集，同set1.symmetric_difference(set2)



組合Set方法

Method		Description
add()		Adds an element to the set
clear()		Removes all the elements from the set
copy()		Returns a copy of the set
difference()		Returns a set containing the difference between two or more sets
difference_update()	去掉與別set重複的資料	Removes the items in this set that are also included in another, specified set `x.difference_update(y)`
discard()	刪除特定資料 資料不存在不會有錯誤	Remove the specified item
intersection()		Returns a set, that is the intersection of two other sets
intersection_update()	去掉與別set無重複資料	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	如兩個set有重複資料會回傳False；反之回傳True	Returns whether two sets have a intersection or not
issubset()		Returns whether another set contains this set or not
issuperset()		Returns whether this set contains another set or not
pop()	隨機刪除一筆資料	Removes an element from the set
remove()	刪除特定資料 資料不存在會有錯誤	Removes the specified element
symmetric_difference()		Returns a set with the symmetric differences of two sets
symmetric_difference_update()		inserts the symmetric differences from this set and another
union()		Return a set containing the union of sets
update()		Update the set with the union of this set and others

迴圈

可搭配:

1. if + continue 中途返回(略過執行內容，直接跳下一個值)
2. Break 強制脫離(強制結束迴圈)

固定次數迴圈

For 迴圈變數 in 可迭代物件:

一定要退格! 程式內容

1.常見可迭代物件(iterable) Range(起始值,結束值,間隔)，Range(結束值)
2.Range跟Index一樣預設會從0開始，所以單獨寫Range(2)輸出為0,1
3.Range的結束值不會輸出，Range(0,10,2)輸出為0,2,4,6,8
4.Range傳出為不可變序列型資料(Immutable sequence type)

單迴圈

```
for n in range(6):
    penup()
    goto(-300, -300 + n * 100)
    setheading(0)
    color('silver')
    pendown()
    forward(600)
    end_fill()
```

雙迴圈

大迴圈n控制列
小迴圈i控制行

```
for n in range(5):
    for i in range(5):
        penup()
        goto(-200 + n * 100, 200 + i * -100)
        dot(25, 'white')
```

條件迴圈

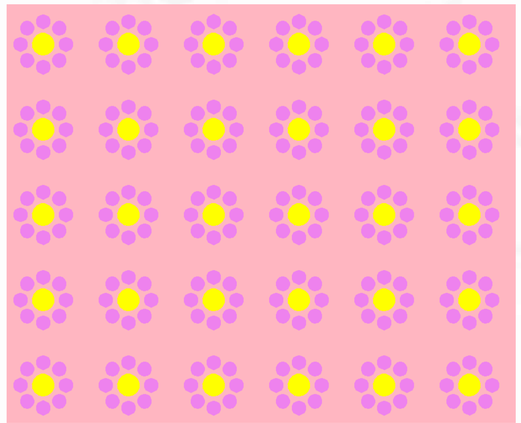
while條件:

一定要退格! 程式內容

```
list1 = ['陳姿婷','陳怡君','林怡君','柯怡君','羽晴','明瑩']
x=0
while x < len(list1):
    print(list1[x])
    x=x+1
```

```
陳姿婷
陳怡君
林怡君
柯怡君
羽晴
明瑩
```

大迴圈n = 5控制列
小迴圈i = 6控制行



判斷式

單一選擇

if 條件:

一定要退格! 程式內容

多重選擇

if條件:

一定要退格! 程式內容

elif條件:

一定要退格! 程式內容

else:

一定要退格! 程式內容

可以有很多個

※”None”的判斷要用 ”is” 不是用 ”==“

例外處理

1. 使用 try: 敘述，至少要有一個檢查 except (捕捉)配合，針對不是執行的錯誤處理(非語法錯誤)
2. 多個檢查 except (捕捉)敘述區段，由上至下 except 逐一檢查，若遇到符合條件的例外類別，則執行該對應敘述區段，則以下的 except 就不再處理。
3. 沒有錯誤的話會執行 try ➔ finally；有錯會執行try ➔ except ➔ finally

```
try: 一定要有一個try跟一個except
    受監視的敘述區段
except 例外類別 1 [as e] :
    處理錯誤的敘述區段 1
except 例外類別 2 [as e] :
    處理錯誤的敘述區段 2
except Exception [as e] :
    處理其它錯誤的敘述區段
finally: 可以省略
    最後會執行敘述區段
```

Exception是很多例外的父類別，可以用這行處理很多例外

```
except Exception as e:
    print(e)#可以把例外類別印出
```

參考網址：1.https://docs.python.org/3/reference/compound_stmts.html#the-try-statement 2. https://docs.python.org/3/library/exceptions.html 3. https://docs.python.org/3/tutorial/errors.html

例外處理

Built in Exception List

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      |    +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
      |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      |    +-- RecursionError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- EncodingWarning
           +-- ResourceWarning
```

常用

例外類別	說明
ZeroDivisionError	除數為 0 的算術運算錯誤。
ValueError	數值錯誤。如使用內建函式時，參數型別與傳入值不符。
NameError	變數名稱未定義，而直接運算產生的錯誤。
IndexError	串引註標(索引)超出宣告範圍
IOError	I/O 異常處理所產生錯誤。
FileNotFoundError	檔案或資料夾找不到時所產生的錯誤。
Exception	程式執行時，所有內建、非系統引發所產生的異常錯誤。

例外處理

Assert

可使用try搭配Assert來處理自定義的錯誤(不在Built in Exception的錯誤)

Assert False會拋出AssertionError，所以可以在Assert後放一個判斷，並加上try except將不合規定時的提示拋出

```
try :
  a = eval( input('a num :') )
  b = eval( input('b num :') )
  assert a>b,'a 一定要大於 b 喔'
  print('a>b')
except AssertionError as e :
  print(e)
```

```
a num :1
b num :10
a 一定要大於 b 喔
```

```
try :
 a = eval( input('a num :') )
 b = eval( input('b num :') )
 assert a>b,'a 一定要大於 b 喔'
 print('a>b')
except AssertionError as e :
 print(e)
```

```
try :
  a = eval( input('請輸入偶數:') )
  assert a % 2 == 0,'喂  叫你輸入偶數'
  print('a 是偶數')
except AssertionError as e :
  print(e)
```

```
請輸入偶數:1
喂  叫你輸入偶數
```

```
try :
 a = eval( input('請輸入偶數:') )
 assert a % 2 == 0,'喂  叫你輸入偶數'
 print('a 是偶數')
except AssertionError as e :
 print(e)
```

```
a = eval( input('請輸入偶數:') )
if not a % 2 == 0:
  raise AssertionError('喂  叫你輸入偶數')
print('a 是偶數')
```

```
請輸入偶數:1

---------------------------------------
AssertionError
<ipython-input-3-d27c49df1d0e> in <modul
      1 a = eval( input('請輸入偶數:') )
      2 if not a % 2 == 0:
----> 3   raise AssertionError('喂  叫你
      4 print('a 是偶數')

AssertionError: 喂  叫你輸入偶數
```

```
##也可以使用raise拋出AssertionError
a = eval( input('請輸入偶數:') )
if not a % 2 == 0:
 raise AssertionError('喂  叫你輸入偶數')
print('a 是偶數')
```

正則運算Regular

Regular Expressions Cheat Sheet

brought to you by e-nor

Meta Characters (must be escaped)

^	Start of a string
$	End of a string
.	Match any single character
*	Match 0 or more of the previous item
\	Escape special meaning of meta characters
[ ]	Match one item in this character set
( )	Group and remember contents as an item
{ }	Repeats the previous item
+	Match one or more of the previous item
|	Either/Or

?	Match 0 or 1 of the previous item
-	Specifies a range of characters

Groups and Ranges

.	Any character
(a|b)	a or b (case sensitive)
[abc]	Match single character that is a or b or c
[^abc]	Negative range (Not a or b or c)
[a-z]	Lowercase letter between a and z
[A-Z]	Uppercase letter between A and Z
[0-7]	Digit between 0 to 7

Character Classes

\s	White space character
\S	Non-white space character
\d	Digit character
\D	Non-digit character
\w	Word
\W	Non-word (e.g. punctuation, spaces)

GA Filter Group Accessors

$Ax	Access group x in field A (e.g. $A1)
$Bx	Access group x in field B (e.g. $B1)

Quantifiers

*	0 or more
+	1 or more
?	0 or 1
{X}	Exactly X (e.g. 3)
{X,}	X or more, (e.g. 3)
{X,Y}	Between X and Y (e.g. 3 and 5)

Anchors

^	Start of string, or start of line in multi-line pattern
$	End of string, or end of line in multi-line pattern

正則運算Regular(簡寫re)

re物件

Match物件

Pattern物件

經過re.compile(“正則運算式”)後會回傳re.Pattern物件

pattern.findall(String,[pos,[endpos]])找到字串中符合pattern格式的字串，並存為list

正則運算Regular

(\d\d)-(\d\d\d\d)-(\d\d\d\d)：符合➔電話號碼格式 00-0000-0000

.*\..*\..*：符合➔兩個以上小數點的字串

.*　　= 任何文字，幾個都可以，可有可無
\.　　= 一個小數點

^\d*\.?\d+$：符合➔數字字串，字串中不能有超過一個的小數點(判斷數值或浮點數，".23"是正確的，"23."會錯誤)

^\d*　= 開頭一定要是數字，幾個都可以，可有可無
\.?　　= ?➔0或1，可以有一個小數點或沒有小數點
\d+$　= +➔1個或多個、$結尾，結尾至少要有一個數值，可以有很多個

```
#很酷的反向思考，判斷是否為數字
import re
def is_number(num):
  pattern = re.compile(r'(.*)\.(.*)\.(.*)')
  if pattern.match(num):
    return False
  return num.replace(".","").isdigit()
```

```
print(is_number('0.2.2.2'))  False
print(is_number('+0.2.2'))   False
print(is_number('2.2.2'))    False
print(is_number('123.4'))    True
print(is_number('1123'))     True
print(is_number('2.p'))      False
print(is_number('0.1123'))   True
```

先透過(.*)\.(.*)\.(.*)將有兩個小數點的資料直接判定為False，再將剩餘資料小數點replace去掉，用isdigit()判斷是否為數字

BIF

垂直打包zip(序列1,序列2,序列3……)

打包的序列要有相等數量的變數，打包輸出為元組

```
t = [0 for n in range(8)]
place = list(list1[0].keys())[1:]
for item in list1:
    if item['期別'][:3] == '109':
        for i in range(8):
            t[i] += int(item[place[i]])

tp = list(zip(t,place))
print(t)
print(place)
print(tp)
```

```
[1965616, 422134, 2520746, 4831286, 292292, 134932, 207628, 516611]
['十八尖山', '青青草原', '城隍廟', '新竹漁港', '賞蟹步道', '青草湖', '十七公里腳踏車道', '新竹公園']
[(1965616, '十八尖山'), (422134, '青青草原'), (2520746, '城隍廟'), (4831286, '新竹漁港'), (292292, '賞蟹步道'), (134932, '青草湖'), (207628, '十七公里腳踏車道'), (516611, '新竹公園')]
```

列舉enumerate()

可同時列出資料內容及其項目編號

```
str1 = '傅采穎陳怡君蔡維章陳佩君李怡君盧羿樺'

for n,item in enumerate(str1):

        print(n,item)
```

```
0 傅
1 采
2 穎
3 陳
4 怡
5 君
6 蔡
7 維
8 章
9 陳
10 佩
11 君
12 李
13 怡
14 君
15 盧
16 羿
17 樺
```

```
str1 = '傅采穎陳怡君蔡維章陳佩君李怡君盧羿樺'
t = 0
for n,item in enumerate(str1):
    if item == '君':
        print(' "君"  出現在第',n,'號位置')
        t += 1
print(' "君"  出現',t,'次')
```

```
 "君"  出現在第 5 號位置
 "君"  出現在第 11 號位置
 "君"  出現在第 14 號位置
 "君"  出現 3 次
```

輸入input()

input的內容會被轉為字串

```
ans  =  input('請輸入一個運算式:  ')
print(ans)
```

```
請輸入一個運算式: 123+456
123+456
```

eval()

1. Eval會將運算式字串自動轉為數值運算式
2. 只能接收運算式
3. 會將內容的文字變數變成數字(如變數不是數值或沒宣告會出異常
4. 單獨輸入數字字串會將該字串轉為數值

```
ans  =  eval(input('請輸入一個運算式:  '))
print(ans)
```

```
請輸入一個運算式: 123+456
579
```

isinstance(‘參數’,資料型態)

判定參數是否為後方指定的資料型態，為array-wise，不管內容元素資料型態為何

```
list1=  [[1,  2,  3],[4,  5,  6]]
print(isinstance(list1,list))
print(isinstance(list1,int))
```

```
True
False
```

format(value,字串格式)

將value以指定的字串格式回傳，可用字串格式同String.foramt

```
num = 3.456

print("數字為"+format(num,'.2f'))
print("數字為{:.2f}".format(num))
```

```
數字為3.46
數字為3.46
```

map()

map內必須放一個函式，會將指定結構的資料一個一個丟給函式，並取得函式的回傳值
map(function, list)，python3 之後 map 回傳的是「map object」，需要強制轉型成list，才會變成看得懂的內容！

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

x = map(myfunc, ('apple', 'banana', 'cherry'), ('orange', 'lemon', 'pineapple'))

print(x)

#convert the map into a list, for readability:
print(list(x))
```

```
<map object at 0x034244F0>
['appleorange', 'bananalemon', 'cherrypineapple']
```

filter()

跟map很像，差別在於map是對傳入的資料進行處理，filter是回傳符合條件的資料(filter後的結果都會存在於原本的資料內容裡)

```
ages = [5, 12, 17, 18, 24, 32]

def myFunc(x):
  if x < 18:
    return False
  else:
    return True

adults = filter(myFunc, ages)

print(adults)
print(list(adults))
```

```
<filter object at 0x1497d123a130>
[18, 24, 32]
```

自訂函式

1. def 函式名稱([參數]):

   程式內容

```
# 函式
def drawflower(x0,y0):
drawflower(-300,100)
drawflower(300,-100)
```

2. 函式要寫在呼叫前
3. 函式裡不可import
4. 放在不同目錄底下的import自訂函式方法

```
import sys
sys.path.insert(0, 'D:\Python\Program')
from ycmodel import sortandrank,openxlsxfile
```

5. def ask_ok(prompt, retries=4, reminder="Please try again!"),對位參數(positional argument)須寫在鍵值參數(keyword argument有預設值)前面
6. 鍵值參數在定義函式前就被指定值的話，後續則無法再改變其值

```
i = 5

def f(arg=i):
    print(arg)

i = 6
f()
```

```
5
```

7. 參數預設值盡量不要使用空串列/字典等可變資料型態，因預設值只會建立一次，使用空串列不會每次都將串列清空，會累積

```
def f(a, L=[]):
    L.append(a)
    return L

print(f(1))
print(f(2))
```

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

輸出結果累積

```
def f(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L

print(f(1))
print(f(2))
```

```
[1]
[2]
```

較正確用法，預設為None，進入函式後再宣告為空串列

自訂函式

*參數：可以擺任意數量的對位參數(不給也可以)，傳入時會自動打包成元組

**參數：可以擺任意數量的鍵值參數(不給也可以)，傳入時會自動打包成字典

一星參數(*參數)一定要放在二星參數(**參數)前，def (*arg, **keyarg)

```
def cheeseshop(kind, *arguments, **keywords):
    print("-- Do you have any", kind, "?")
    print("-- I'm sorry, we're all out of", kind)
    for arg in arguments:
        print(arg)
    print("-" * 40)
    for kw in keywords:
        print(kw, ":", keywords[kw])
```

```
cheeseshop("Limburger", "It's very runny, sir.",
           "It's really very, VERY runny, sir.",
           shopkeeper="Michael Palin",
           client="John Cleese",
           sketch="Cheese Shop Sketch")
```

```
-- Do you have any Limburger ?
-- I'm sorry, we're all out of Limburger
It's very runny, sir.
It's really very, VERY runny, sir.
----------------------------------------
shopkeeper : Michael Palin
client : John Cleese
sketch : Cheese Shop Sketch
```

def 函式名稱(參數, /, 參數,*,參數)：可使用/跟*來做區隔，/前一定是position argument，*後一定是keyword argument，中間可兩種

模組

datetime(可直接做日期運算)

datetime常用類別

1.class datetime.**date**(日期,年月日)
An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Attributes: year, month, and day.

2.class datetime.**time**(時間,時/分/秒/毫秒-millisecond/微秒-microsecond)
An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. (There is no notion of “leap seconds” here.) Attributes: hour, minute, second, microsecond, and tzinfo.

3.class datetime.**datetime**(日期+時間)
A combination of a date and a time. Attributes: year, month, day, hour, minute, second, microsecond, and tzinfo.

4.class datetime.**timedelta**(時間間隔)
A duration expressing the difference between two date, time, or datetime instances to microsecond(微秒) resolution.

datetime.now().now：取得當下的年月日時分秒以及timestamp(到1970年1月1日0分0秒之間的秒數)

datetime.now().year：取得當下的年

datetime.now().month：取得當下的月

datetime.now().day：取得當下的日

datetime.now().weekday()+1：取得當下的星期會從0開始數，所以要+1

datetime.now().date()：只顯示日期

datetime.now().time()：只顯示時間

datetime(2018,5,21,19,50) - datetime(2018,5,18,20,32) ：直接做運算 #Output = datetime.timedelta(days=2, seconds=83880)

datetime(2018,5,18,20,32) + timedelta(days = 1)：日期加減用timedelta或其他時間格式，不可以直接+1，型態不同會出錯

參考網址https://docs.python.org/3/library/datetime.html

datetime(可直接做日期運算)

一定要會的函式

datetime.strftime(format) ：將日期格式變為字串(畫圖的時候可以用，如為日期格式會被跳過，改為字串就不會有跳過的問題)

Return a string representing the date and time, controlled by an explicit format string. For a complete list of formatting directives

classmethod datetime.strptime(date_string, format)：將字串轉為日期類別，後面可以直接取得年月日

Return a datetime corresponding to date_string, parsed according to format.

```
import datetime
str1 = input('請輸入生日(yyyy-mm-dd:)')
d1 = datetime.datetime.strptime(str1,'%Y-%m-%d')
print(d1)
print(d1.year)
print(d1.month)
print(d1.day)
```

```
請輸入生日(yyyy-mm-dd:)2000-01-01
2000-01-01 00:00:00
2000
1
1
```

datetime(可直接做日期運算)

日期格式

Directive	Meaning	Example
%a	Weekday as locale's abbreviated name.	Sun, Mon, …, Sat (en_US); So, Mo, …, Sa (de_DE)
%A	Weekday as locale's full name.	Sunday, Monday, …, Saturday (en_US); Sonntag, Montag, …, Samstag (de_DE)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, …, 6
%d	Day of the month as a zero-padded decimal number.	01, 02, …, 31
%b	Month as locale's abbreviated name.	Jan, Feb, …, Dec (en_US); Jan, Feb, …, Dez (de_DE)
%B	Month as locale's full name.	January, February, …, December (en_US); Januar, Februar, …, Dezember (de_DE)
%m	Month as a zero-padded decimal number.	01, 02, …, 12
%y	Year without century as a zero-padded decimal number.	00, 01, …, 99
%Y	Year with century as a decimal number.	0001, 0002, …, 2013, 2014, …, 9998, 9999
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, …, 23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, …, 12
%p	Locale's equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)

Directive	Meaning	Example
%M	Minute as a zero-padded decimal number.	00, 01, …, 59
%S	Second as a zero-padded decimal number.	00, 01, …, 59
%f	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001, …, 999999
%z	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).	(empty), +0000, -0400, +1030, +063415, -030712.345216
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT
%j	Day of the year as a zero-padded decimal number.	001, 002, …, 366
%U	Week number of the year (Sunday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	00, 01, …, 53
%W	Week number of the year (Monday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	00, 01, …, 53
%c	Locale's appropriate date and time representation.	Tue Aug 16 21:30:00 1988 (en_US); Di 16 Aug 21:30:00 1988 (de_DE)
%x	Locale's appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)
%X	Locale's appropriate time representation.	21:30:00 (en_US); 21:30:00 (de_DE)
%%	A literal '%' character.	%

參考網址https://docs.python.org/3/library/datetime.html

datetime(可直接做日期運算)

```python
import datetime
now = datetime.date.today()
str1 = '110/4/12'
y,m,d = str1.split('/')
y = int(y) +1911

str2 = str(y) + '/' + m + '/' + d
bth = datetime.datetime.strptime(str2, '%Y/%m/%d')


age = now.year - bth.year

if now.month > bth.month:  #今天的月份大於生日月份 - 生日過了
    age = age
elif now.month < bth.month: #今天的月份小於生日月份 - 生日還沒過所以歲數-1
    age = age - 1
else:                       #今天的月份等於生日月份，接著判斷日期
    if now.day >= bth.day:  #今天的日期大於生日日期，生日還沒過
        age = age
    else:
        age = age -1        ##今天的月份小於生日日期 - 生日還沒過所以歲數-1

print('今年',age,'歲')
```

```
今年 1 歲
```

```python
import datetime
now = datetime.date.today()
str1 = '110/4/12'
y,m,d = str1.split('/')
y = int(y) +1911

str2 = str(y) + '/' + m + '/' + d
bth = datetime.datetime.strptime(str2, '%Y/%m/%d')


age = now.year - bth.year

if now.month > bth.month:     #今天的月份大於生日月份 - 生日過了
   age = age
elif now.month < bth.month:   #今天的月份小於生日月份 - 生日還沒過所以歲數-1
   age = age - 1
else:                         #今天的月份等於生日月份，接著判斷日期
   if now.day >= bth.day:     #今天的日期大於生日日期，生日還沒過
      age = age
   else:
      age = age -1            #今天的月份小於生日日期 - 生日還沒過所以歲數-1

print('今年',age,'歲')
```

未來製作程式做資料填寫，不要有年齡給User填
年齡會每年變動，應使用計算算出(導出資料)

Random模組

`random.` **`randrange`**(*stop*)
`random.` **`randrange`**(*start*, *stop*[, *step*])

隨機產生範圍內的亂數(不含stop)

Return a randomly selected element from `range(start, stop, step)`. This is equivalent to `choice(range(start, stop, step))`, but doesn't actually build a range object.

The positional argument pattern matches that of `range()`. Keyword arguments should not be used because the function may use them in unexpected ways.

Changed in version 3.2: `randrange()` is more sophisticated about producing equally distributed values. Formerly it used a style like `int(random()*n)` which could produce slightly uneven distributions.

Deprecated since version 3.10: The automatic conversion of non-integer types to equivalent integers is deprecated. Currently `randrange(10.0)` is losslessly converted to `randrange(10)`. In the future, this will raise a `TypeError`.

Deprecated since version 3.10: The exception raised for non-integral values such as `randrange(10.5)` or `randrange('10')` will be changed from `ValueError` to `TypeError`.

`random.` **`randint`**(*a*, *b*)

隨機產生範圍內的亂數含b

Return a random integer *N* such that `a <= N <= b`. Alias for `randrange(a, b+1)`.

`random.` **`choice`**(*seq*) ¶

從序列中隨機挑取一個元素

Return a random element from the non-empty sequence *seq*. If *seq* is empty, raises `IndexError`.

`random.` **`shuffle`**(*x*[, *random*]) ¶

將序列打亂(打亂籤筒的概念)，不會回傳值

Shuffle the sequence *x* in place.

The optional argument *random* is a 0-argument function returning a random float in [0.0, 1.0); by default, this is the function `random()`.

To shuffle an immutable sequence and return a new shuffled list, use `sample(x, k=len(x))` instead.

Note that even for small `len(x)`, the total number of permutations of *x* can quickly grow larger than the period of most random number generators. This implies that most permutations of a long sequence can never be generated. For example, a sequence of length 2080 is the largest that can fit within the period of the Mersenne Twister random number generator.

Deprecated since version 3.9, will be removed in version 3.11: The optional parameter *random*.

Openpyxl

1. 可操作excel
2. 工作簿:Workbook　　工作表:Worksheet
3. 方法開頭為大寫 = 建構式 `wb = Workbook()`
4. 每個工作表就是一個二維陣列的概念
5. 使用Excel座標為陣列起始
6. 新增資料是以列新增，可使用append()
7. 讀取工作簿load_workbook(“工作簿名稱”)　儲存工作簿save(“工作簿名稱")

```python
from openpyxl import load_workbook
#1.打開工作簿
wb = load_workbook("成績單.xlsx")

#2.確認工作簿內有多少工作表
print(wb.sheetnames) #將該工作簿子的工作表明稱存成一個list

#3打開指定工作表
sheet1 = wb['學生成績']

#4.將每列資料存放到list內
list2 = []
for row in list(sheet1.rows):
    list1 = []
    for col in row:
        list1.append(col.value)
    list2.append(list1)
print(list2)

#print(sheet1['A1']) <Cell '學生成績'.A1> 格子物件
#print(sheet1['A1'].value) 讀取工作表格內容資料
#print(sheet1.rows) <generator object Worksheet._cells_by_row at 0x000000000293FE58> 生成器物件(可迭代物件)，可轉型list
#print(list(sheet1.rows)) 讀取列資料，並轉型為list
#[print( item.value for item in list(sheet1.rows)] 讀取每列的資料內容，但讀出資料為元組
```

Pytube

Youtube串流資訊

這行解析度最高又有聲音

```
yt = pytube.YouTube("'https://www.youtube.com/watch?v=KGQRrCAM33M'")
print(yt.streams)
```

res 影像檔解析度
abr 純影音檔
acodec有聲音

```
[<Stream: itag="17" mime_type="video/3gpp" res="144p" fps="6fps" vcodec="mp4v.20.3" acodec="mp4a.40.2" progressive="True" type="video">,
 <Stream: itag="18" mime_type="video/mp4" res="360p" fps="25fps" vcodec="avc1.42001E" acodec="mp4a.40.2" progressive="True" type="video">,
 <Stream: itag="22" mime_type="video/mp4" res="720p" fps="25fps" vcodec="avc1.64001F" acodec="mp4a.40.2" progressive="True" type="video">,
 <Stream: itag="313" mime_type="video/webm" res="2160p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="401" mime_type="video/mp4" res="2160p" fps="25fps" vcodec="av01.0.12M.08" progressive="False" type="video">,
 <Stream: itag="271" mime_type="video/webm" res="1440p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="400" mime_type="video/mp4" res="1440p" fps="25fps" vcodec="av01.0.12M.08" progressive="False" type="video">,
 <Stream: itag="137" mime_type="video/mp4" res="1080p" fps="25fps" vcodec="avc1.640028" progressive="False" type="video">,
 <Stream: itag="248" mime_type="video/webm" res="1080p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="399" mime_type="video/mp4" res="1080p" fps="25fps" vcodec="av01.0.08M.08" progressive="False" type="video">,
 <Stream: itag="136" mime_type="video/mp4" res="720p" fps="25fps" vcodec="avc1.4d401f" progressive="False" type="video">,
 <Stream: itag="247" mime_type="video/webm" res="720p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="398" mime_type="video/mp4" res="720p" fps="25fps" vcodec="av01.0.05M.08" progressive="False" type="video">,
 <Stream: itag="135" mime_type="video/mp4" res="480p" fps="25fps" vcodec="avc1.4d401e" progressive="False" type="video">,
 <Stream: itag="244" mime_type="video/webm" res="480p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="397" mime_type="video/mp4" res="480p" fps="25fps" vcodec="av01.0.04M.08" progressive="False" type="video">,
 <Stream: itag="134" mime_type="video/mp4" res="360p" fps="25fps" vcodec="avc1.4d401e" progressive="False" type="video">,
 <Stream: itag="243" mime_type="video/webm" res="360p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="396" mime_type="video/mp4" res="360p" fps="25fps" vcodec="av01.0.01M.08" progressive="False" type="video">,
 <Stream: itag="133" mime_type="video/mp4" res="240p" fps="25fps" vcodec="avc1.4d4015" progressive="False" type="video">,
 <Stream: itag="242" mime_type="video/webm" res="240p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="395" mime_type="video/mp4" res="240p" fps="25fps" vcodec="av01.0.00M.08" progressive="False" type="video">,
 <Stream: itag="160" mime_type="video/mp4" res="144p" fps="25fps" vcodec="avc1.4d400c" progressive="False" type="video">,
 <Stream: itag="278" mime_type="video/webm" res="144p" fps="25fps" vcodec="vp9" progressive="False" type="video">,
 <Stream: itag="394" mime_type="video/mp4" res="144p" fps="25fps" vcodec="av01.0.00M.08" progressive="False" type="video">,
 <Stream: itag="139" mime_type="audio/mp4" abr="48kbps" acodec="mp4a.40.5" progressive="False" type="audio">,
 <Stream: itag="140" mime_type="audio/mp4" abr="128kbps" acodec="mp4a.40.2" progressive="False" type="audio">,
 <Stream: itag="249" mime_type="audio/webm" abr="50kbps" acodec="opus" progressive="False" type="audio">,
 <Stream: itag="250" mime_type="audio/webm" abr="70kbps" acodec="opus" progressive="False" type="audio">,
 <Stream: itag="251" mime_type="audio/webm" abr="160kbps" acodec="opus" progressive="False" type="audio">]
```

Pytube

1. 設定下載目錄
2. 過濾Youtube串流資訊，找到res = 720p的第一個資訊(因為後面的沒有聲音)，並下載到指定目錄

```
download_location = 'd:/YoutubeVideo/'
yt.streams.filter(res='720p').first().download(download_location)
```

作業系統os模組

應用程式控制硬碟中的檔案(檔案總管):

1. os.remove(檔案路徑) ：刪除檔案、os.rmdir(資料夾路徑) ：用來刪除已存在的”空”資料夾
2. 取絕對路徑最後的檔名os.path.basename(r'C:\Users\Administrator\DesktopC:\Users\Administrator\Desktop\1090323切結書.docx')
3. 取絕對路徑檔案目錄os.path.dirname(r'C:\Users\Administrator\DesktopC:\Users\Administrator\Desktop\1090323切結書.docx')
4. 更改檔名os.rename(r'C:\Users\Administrator\Desktop\1090323切結書.docx',r'C:\Users\Administrator\Desktop\1090323切結書.doc')
5. 變更工作目錄os.chdir('D:\YoutubeVideo')
6. os.path.isdir(資料夾路徑)、os.path.isfile(檔案路徑) ：檢查指定的資料夾或檔案是否存在
7. os.path.exists(資料夾路徑/檔案路徑) ：檢查指定的資料夾或檔案是否存在
8. os.mkdir(資料夾路徑) ：在指定的路徑中建立資料夾，若資料夾已經存在會出現錯誤訊息
9. 開檔➔物件變數=open(檔案路徑,’模式’)，關檔➔物件變數.close()，檔案開啟後沒有要用需close，如無close可能造成資料流失
10. with open(檔案路徑,’模式’) as 物件變數，使用with…as…語法來開啟檔案則檔案處理完後會自動關閉

模式	說明
r	讀取模式(預設值)。若資料檔不存在，會出現錯誤。(不可寫入)
w	寫入模式。若資料檔不存在，會建立該名稱的資料檔；若資料檔已存在，則原資料檔的內容會先被刪除，再寫入新的資料。(不可讀取)
a	新增模式。若資料檔不存在，會建立該名稱的資料檔；若資料檔已存在，則新寫入的資料會新增至原資料檔內容的尾端。(不可讀取)
r+	讀寫模式。讀取時，使用方式同 r 模式。寫入時，則寫入的資料會覆蓋原檔案相同位置的內容，若檔案不存在，會出現錯誤。
w+	讀寫模式。寫入時，使用方式同 w 模式。讀取時，需先用 seek() 函式指定讀取指標位址，才能讀取所需資料。seek(0)為檔案開頭。
a+	新增讀寫模式。寫入新增時，使用方式同 a 模式。讀取時，需先用 seek() 函式指定讀取指標位址，才能讀取所需資料。seek(0)為檔案開頭。

函式	說明
flush()	清理緩衝區釋放記憶空間，若緩衝區內還有資料，會全部寫入檔案中。
write(字串)	將字串寫入資料串流。先暫存於緩衝區，再由緩衝區寫入檔案中。
read([size])	從檔案中讀取指定 size 的字元。若未指定 size，則讀取指標位置後面所有字元。
readline([size])	從檔案中讀取所在行指定 size 的字元。若未指定 size，則讀取一整行。
readlines()	從檔案讀取所有行的資料，並回傳一個串列，一個元素放置一個行的內容。
seek()	移動檔案文件讀取指標的位置，如: seek(0)為檔案的開頭。 以byte為移動單位，中文佔2個byte，其他1個

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
王		一		心		,		8	5	,		9	0	\	n	張	

subprocess模組

命令提示字元執行指令

1. subprocess.run(‘指令’,capture_output = True)

Numpy模組

將串列轉為陣列(數字或浮點數使用效果較佳)

1. 物件為N-dimensional array(多維陣列)
2. 將串列轉為陣列x = np.array([[1., 2., 3.], [4, 5, 6]],(np.型態))，如有要指定型態可在後方指定
3. x.shape陣列的大小(2,3) 兩列三行, x.dim陣列的維度=2維,x.size總共有幾個元素=6
4. x.dtype陣列資料內容的型態float64(需一致，如其中一值為浮點數，會全數為浮點數)，預設int dtype為int64； 預設浮點數型態為float64
5. 可使用切片，切片方式使用在同一個中括弧內
6. 二維陣列切片[列切片,行切片]，[0,1] = 2，[:,1] = [2.,5.] (每一列都要，只要第一個索引的數值)
7. Np arrary轉list ➔np_arrary.tolist() list轉Np arrary➔np.arrary(list1)
8. 常用type: int64、uint8(0~255)方便給顏色使用、float64

```
x = np.array([[1,2,3,4,5], [10,20,30,40,50], [11,22,33,44,55], [12,22,32,42,52]])
#np陣列切片

print(x)
print()
#拿一行 [3,30,33,32]
print(x[:,2])
#拿一列 [10,20,30,40,50]
print(x[1,:])
#拿一格 4
print(x[0,3])
#拿一塊 [[20,30,40],[22,33,44]]
print(x[1:3,1:4])
```

```
[[ 1  2  3  4  5]
 [10 20 30 40 50]
 [11 22 33 44 55]
 [12 22 32 42 52]]

[ 3 30 33 32]
[10 20 30 40 50]
4
[[20 30 40]
 [22 33 44]]
```

```
list1 = [[1,2,3,4,5], [10,20,30,40,50], [11,22,33,44,55], [12,22,32,42,52]]
#list切片

print(list1)
print()
#拿一行 [3,30,33,32]
print(list(zip(*list1))[2])
#拿一列 [10,20,30,40,50]
print(list1[1])
#拿一格 4
print(list1[0][3])
#拿一塊 [[20,30,40],[22,33,44]]
print([list1[1][1:4],list1[2][1:4]])
```

```
[[1, 2, 3, 4, 5], [10, 20, 30, 40, 50], [11, 22, 33, 44, 55], [12, 22, 32, 42, 52]]

(3, 30, 33, 32)
[10, 20, 30, 40, 50]
4
[[20, 30, 40], [22, 33, 44]]
```

```
x = np.array([[1,2,3,4,5], [10,20,30,40,50], [11,22,33,44,55], [12,22,32,42,52]])
list1 = [[1,2,3,4,5], [10,20,30,40,50], [11,22,33,44,55], [12,22,32,42,52]]

print(x.shape)  #幾列幾行
print(x.size)  #有幾個元素

print(len(list1),len(list1[0]))
print(len(list1)*len(list1[0]))
```

```
(4, 5)
20
4 5
20
```

參考網址1.https://numpy.org/doc/stable/reference/index.html#reference 參考網址2. https://medium.com/python4u/hello-numpy-b5ebe67a1ada

Numpy模組

生成陣列函式與list生成寫法比對

```
np_arr  =  np.zeros((2,  3))                                  #  建立一個2x3全為0的陣列  [[0.0,  0.0,  0.0],  [0.0,  0.0,  0.0]]
list1  =  [[0.  for  n  in  range(3)]  for  m  in  range(2)]

np_arr  =  np.ones((2,  3,  4))                               #  建立一個2x3x4全為1的陣列  2個3x4全為1的陣列
list1  =  [[[1.  for  n  in  range(4)]  for  m  in  range(3)]  for  i  in  range(2)]

np_arr  =  np.arange(1,  10,  2)                              #  建立一個由1開始，不超過10，間隔值為2的均勻數值陣列  [1,  3,  5,  7,  9]
list1  =[n  for  n  in  range(1,10,2)]

np_arr  =  np.linspace(1,  10,  5)                            #  建立一個0到10之間，均勻的5個數值陣列  [0.0,  2.5,  5.0,  7.5,  10.0]
list1=[n*(10-5)/(5-1)  +1  for  n  in  range(5-1)]

np_arr  =  np.full((3,2),  8)                                 #  建立一個3x2全為8的陣列
list1  =  [[8  for  i  in  range(2)]  for  n  in  range(3)]

np_arr  =  np.eye(5)                                          #  建立一個2x2的單位矩陣，一定是方陣(斜角都是1)[1,0][0,1]
list1  =  [[  1  if  i  ==  n  else  0  for  i  in  range(5)  ]  for  n  in  range(5)]

np_arr  =  np.random.random((2,3))                            #  建立一個2x3  0到1中的隨機值矩陣
list1  =  [[random.random()  for  n  in  range(3)]  for  m  in  range(2)]
```

Numpy模組

```
import  numpy  as  np
np_arr  =  np.array([[1,2,3,4,5],  [10,200,300,400,500],  [11,22,33,44,55],  [12,22,32,42,52]],np.uint8)
print(np_arr)
```

```
[[  1   2   3   4   5]
 [ 10 200  44 144 244]
 [ 11  22  33  44  55]
 [ 12  22  32  42  52]]
```

Uint8只允許0-255的數值，超過會直接改為 % 256(元素除256餘數)的數值
300➔44
400➔144
500➔244

強制轉型 arr.tolist(), tostring(), toflaot(), toint()

```
a  =  np.array([1,  2,  3])
b  =  np.array([(2.5,  1,  4.5),  (5,  6,  7,  8)])
a.shape                                  #  Array  dimensions  (3,)
b.shape                                  #(2,)  因二維陣列內容長度不同，故直接識別為兩個物件，所以還是一為陣列

len(a)                                     #  Length  of  array  3
len(b)                                     #  Length  of  array  2
b.ndim                                     #  Number  of  array  dimensions  幾為陣列
a.size                                     #  Number  of  array  elements  3  總共有幾個元素
b.size                                     #  Number  of  array  elements
b.dtype                                    #  Data  type  of  array  elements  dtype('O')資料型態
a.dtype                                    #  Data  type  of  array  elements  dtype('int64')
b.dtype.name                             #  Name  of  data  type  object
a.dtype.name                             #  Name  of  data  type  int64
a1  =  a.astype(uint8)                     #  Convert  an  array  to  a  different  type  強制轉型，不會變更原本的型態，會回傳新的陣列
a.dtype                                    #  dtype('int64')
a1.dtype                                   #dtype('uint8')
```

Numpy模組

陣列運算

```
import  numpy  as  np
a  =  np.array([(1,  2,  3),(4,  5,  6)])
b  =  np.array([(1,  3,  3),(4,  7,  6)])

a  ==  b  #array([[  True,  False,    True],  比較每一個元素哪裡不一樣  element-wise
          #       [  True,  False,    True]])

a<3  #array([[  True,    True,  False],
      #       [False,  False,  False]])
```

```
np.array_equal(a,  b)  #False  array-wise  整體比較
```

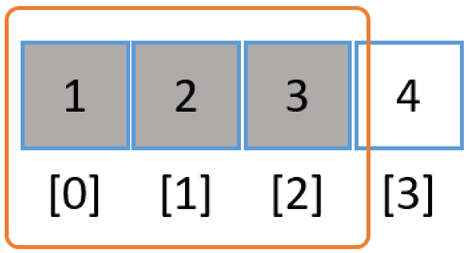
```
a  =  np.array([2,  5,  7])
b  =  np.array([(0,  1),  (2,  3)])
a.sum()                                          #  Array-wise  sum  14
#a.max()                                         #  Array-wise  maximum  value  7
#a.min()                                         #  Array-wise  minimum  value  2
#b.max(axis=0)                                   #  Maximum  value  of  an  array  row  array([2,  3])
#b.max(axis=1)                                   #  Maximum  value  of  an  array  column  array([1,  3])
#np.median(a)                                    #  Median  5.0
#np.mean(a)                                      #  Mean  4.666666666666667
#np.std(a)                                       #  Standard  deviation  2.0548046676563256
```

Numpy模組

布林切片:把所有符合條件的元素取出成為一個一維陣列

- Boolean Indexing

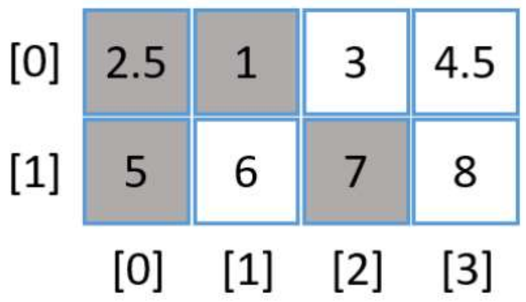
```
>>> a[a < 4]
array([1, 2, 3])
```



前衛索引:以陣列位置作為索引，並輸出，第一個陣列為列索引，第二陣列為行索引

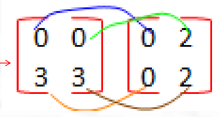
- Fancy Indexing

```
>>> b[[1, 1, 0, 0], [0, 2, 1, 0]]
array([5. , 7. , 1. , 2.5])
```



(1,0),(1,2),(0,1),(0,0) ➔ arrary([5., 7., 1., 2.5])

```
rows = np.array([[0,0],[3,3]])
cols = np.array([[0,2],[0,2]])
y = x[rows,cols]
```




```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

x

```
[[ 0  2]
 [ 9 11]]
```

y

(0,0),(0,2),(3,0),(3,2) ➔ arrary([0,2]
[9,11])

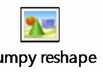
參考網址 numpy的高級切片https://www.runoob.com/numpy/numpy-advanced-indexing.html

Numpy模組

矩陣轉置np.transpose():行變列，列變行

```
#Transposing  Array  矩陣轉置  np.transpose(array)  or  array.T
import  numpy  as  np
a  =  np.array([(1,  3,  5,  7),  (2,  4,  6,  8)])
b  =  np.transpose(a)
print(a)
print(b)
b.T
print()

list1  =  [[1,  3,  5,  7],  [2,  4,  6,  8]]
list2  =  list(zip(*list1))
print(list1)
print(list2)
```

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



np.reshape():改變陣列形狀元素數量需與元陣列相同(只打一個整數代表變更為一維，可打-1忽略行數/列數)

```
#Changing  Array  Shape
a  =  np.array([(1,  3,  5,  7),  (2,  4,  6,  8)])
a.shape  #(2,  4)
a.ravel()  #打平成一維陣列  array([1,  3,  5,  7,  2,  4,  6,  8])
a.reshape(4,  2)  #變更為
                  #  array([[1,  3],
                  #         [5,  7],
                  #         [2,  4],
                  #         [6,  8]])
```

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

```
>>> a = np.arange(6).reshape((3, 2))
>>> a
array([[0, 1],
       [2, 3],
       [4, 5]])
```

預設排序為C型

C型排序，橫向打平，橫向放入資料

012345➔(0,1,2) (3,4,5)

F型排序，直向打平，直向放入資料

024135➔ (0,4,3)

(2,1,5)

```
>>> np.reshape(a, (2, 3)) # C-like index ordering
array([[0, 1, 2],
       [3, 4, 5]])
>>> np.reshape(np.ravel(a), (2, 3)) # equivalent to C ravel then C reshape
array([[0, 1, 2],
       [3, 4, 5]])
>>> np.reshape(a, (2, 3), order='F') # Fortran-like index ordering
array([[0, 4, 3],
       [2, 1, 5]])
>>> np.reshape(np.ravel(a, order='F'), (2, 3), order='F')
array([[0, 4, 3],
       [2, 1, 5]])
```

Numpy模組

np.append():新增資料

```
import  numpy  as  np
a  =  np.array([1,  2,  3,  4])
b  =  np.array([6,  7,  8,  9])
list1  =  [1,  2,  3,  4]
list2  =  [6,  7,  8,  9]

np.append(a,  5)  #array([1,  2,  3,  4,  5])
np.append(a,  b)  #array([1,  2,  3,  4,  6,  7,  8,  9])

list1  +  list2  #[1,  2,  3,  4,  6,  7,  8,  9]
list1.append(list2)  #[1,  2,  3,  4,  [6,  7,  8,  9]]
```

Numpy模組

Broadcast運算 + np.where(條件,True回傳,False回傳)

np.ogrid[]網格生成陣列(連號)

```
x, y = np.ogrid[:3, :4]
print(x)
print(y)
np.where(x < y, x, 10 + y)    # both x and 10+y are broadcast廣播  陣列形狀不同時，自動觸發廣播機機制，會將矩陣大小相乘
```

```
[[0]
 [1]
 [2]]
[[0 1 2 3]]
array([[10,  0,  0,  0],
       [10, 11,  1,  1],
       [10, 11, 12,  2]])
```

x broadcast			
0	0	0	0
1	1	1	1
2	2	2	2

y broadcast			
0	1	2	3
0	1	2	3
0	1	2	3

np.where(x < y, x, y+10)			
10 (0<0 = False) (y+10 = 0+ 10 = 10	0 (0<1 = True) (x = 0)	0 (0<2 = True) (x = 0)	0 (0<3 = True) (x = 0)
10 (0<0 = False) (y+10 = 0+ 10 = 10	11 (0<0 = False) (y+10 = 1+ 10 = 11	1 (1<2 = True) (x = 1)	1 (1<3 = True) (x = 1)
10 (0<0 = False) (y+10 = 0+ 10 = 10	11 (0<1 = False) (y+10 = 1+ 10 = 11	12 (0<0 = False) (y+10 = 2+ 10 = 12	2 (2<3 = True) (x = 2)

Numpy模組

np.ix_[]網格生成陣列(可以自己指定號碼)

```
import numpy as np

x=np.arange(32).reshape((8,4))
print (x[np.ix_([1,5,7,2],[0,3,1,2])])
```

x

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23],
       [24, 25, 26, 27],
       [28, 29, 30, 31]])
```

	0	1	2	3
	np.ix_([1,5,7,2],[0,3,1,2])			
0	4 [1,0]	7 [1,3]	5 [1,1]	6 [1,2]
1	20 [5,0]	23 [5,3]	21 [5,1]	22 [5,2]
2	28 [7,0]	31 [7,3]	29 [7,1]	30 [7,2]
3	8 [2,0]	11 [2,3]	9 [2,1]	10 [2,2]

Numpy模組

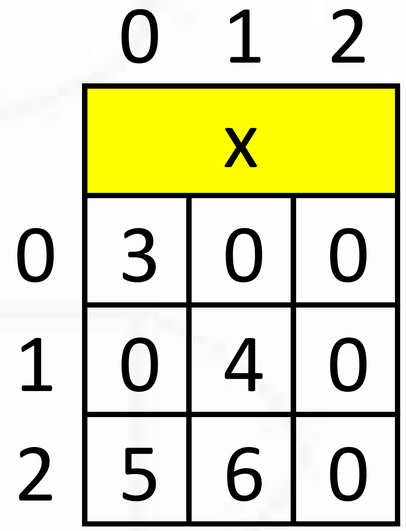
Np.nonzero(陣列):回傳不是0的索引位置(垂直放置)

不是0的位置➔3(0,0)、4(1,1)、5(2,0)、6(1,2)

```
x = np.array([[3, 0, 0], [0, 4, 0], [5, 6, 0]])
np.nonzero(x)
```

```
(array([0, 1, 2, 2]), array([0, 1, 0, 1]))
```

輸出結果: (array([0, 1, 2, 2)],
array([0, 1, 0, 1)])



陣列[np.nonzero(陣列)]會回傳不是0的元素(3,4,5,6)

```
>>> x[np.nonzero(x)]
array([3, 4, 5, 6])
>>> np.transpose(np.nonzero(x))
array([[0, 0],
       [1, 1],
       [2, 0],
       [2, 1]])
```

False就是0，故np.nonzero('條件')會回傳符合條件的元素位置

```
>>> a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> a > 3
array([[False, False, False],
       [ True,  True,  True],
       [ True,  True,  True]])
>>> np.nonzero(a > 3)
(array([1, 1, 1, 2, 2, 2]), array([0, 1, 2, 0, 1, 2]))
```

```
>>> a[np.nonzero(a > 3)]
array([4, 5, 6, 7, 8, 9])
>>> a[a > 3]  # prefer this spelling
array([4, 5, 6, 7, 8, 9])
```

Numpy模組

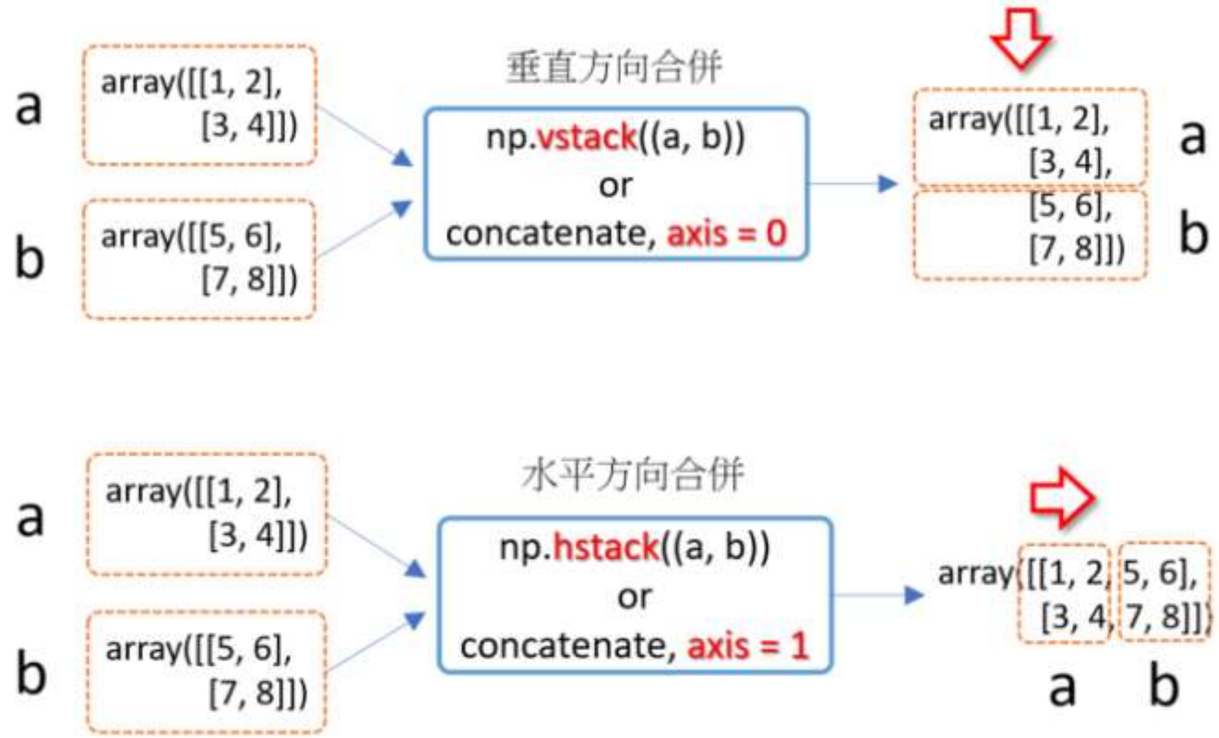
陣列合併

np.v/hstack(陣列,陣列)

np.concatenate((陣列,陣列),axis = 0/1)

- vstack：垂直方向合併
- hstack：水平方向合併

```
>>> a = np.array([(1, 2), (3, 4)])
>>> b = np.array([(5, 6), (7, 8)])
>>> np.vstack((a, b))                # 垂直方向合併
array([[1, 2],
 [3, 4],
 [5, 6],
 [7, 8]])
>>> np.hstack((a, b))                # 水平方向合併
array([[1, 2, 5, 6],
 [3, 4, 7, 8]])
```



- concatenate (axis = 0)：沿垂直方向合併
- concatenate (axis = 1)：沿水平方向合併

```
>>> np.concatenate((a, b), axis = 0)    # axis=0，沿垂直方向合併
array([[1, 2],
 [3, 4],
 [5, 6],
 [7, 8]])
>>> np.concatenate((a, b), axis = 1)    # axis=1，沿水平方向合併
array([[1, 2, 5, 6],
 [3, 4, 7, 8]])
```

Numpy模組

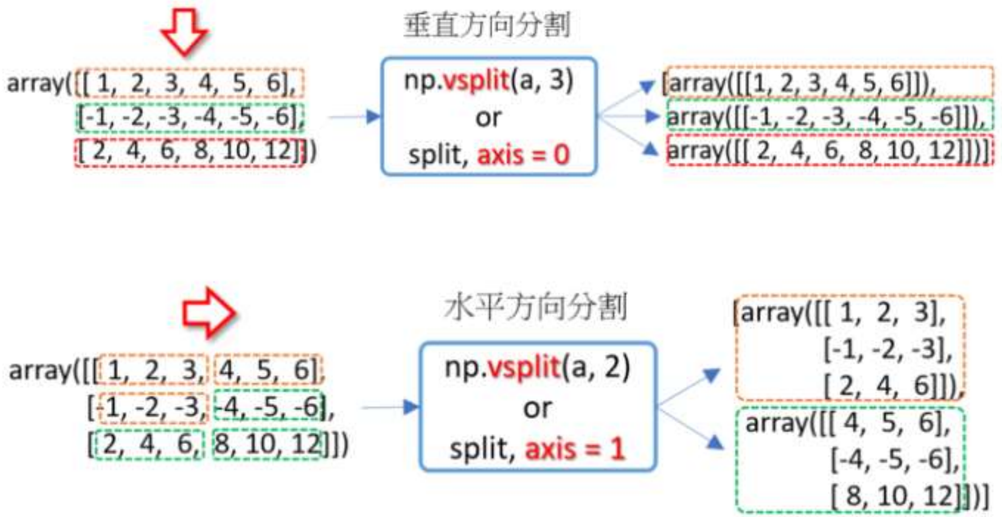
陣列切割
np.h/vsplit(陣列, 整數/一維陣列)

- Splitting Arrays

跟Combining Arrays一樣，NumPy提供一些在不同軸向的切割方式，如下說明：

- vsplit：垂直方向分割
- hsplit：水平方向分割

```
>>> a = np.array([(1, 2, 3, 4, 5, 6), (-1, -2, -3, -4, -5, -6), (2, 4,
6, 8, 10, 12)])
>>> a
array([[ 1, 2, 3, 4, 5, 6],
 [-1, -2, -3, -4, -5, -6],
 [ 2, 4, 6, 8, 10, 12]])
>>> np.vsplit(a, 3)                  # 垂直方向分割
[array([[1, 2, 3, 4, 5, 6]]), array([[-1, -2, -3, -4, -5, -6]]),
array([[ 2, 4, 6, 8, 10, 12]])]
>>> np.hsplit(a, 2)                  # 水平方向分割
[array([[ 1, 2, 3],
 [-1, -2, -3],
 [ 2, 4, 6]]), array([[ 4, 5, 6],
 [-4, -5, -6],
 [ 8, 10, 12]])]
>>> np.split(a, 3, axis=0)           # axis=0，沿垂直方向分割
[array([[1, 2, 3, 4, 5, 6]]), array([[-1, -2, -3, -4, -5, -6]]),
array([[ 2, 4, 6, 8, 10, 12]])]
>>> np.split(a, 2, axis=1)           # axis=1，沿水平方向分割
[array([[ 1, 2, 3],
 [-1, -2, -3],
 [ 2, 4, 6]]), array([[ 4, 5, 6],
 [-4, -5, -6],
 [ 8, 10, 12]])]
```




```
>>> x = np.arange(9.0)
>>> np.split(x, 3)
[array([0.,  1.,  2.]), array([3.,  4.,  5.]), array([6.,  7.,  8.])]
```

```
>>> x = np.arange(8.0)
>>> np.split(x, [3, 5, 6, 10])
[array([0.,  1.,  2.]),
 array([3.,  4.]),
 array([5.]),
 array([6.,  7.]),
 array([], dtype=float64)]
```

Numpy模組

聚合函式
Aggregate and Statistical Functions

	Functions	Description
平均值	np.mean()	Compute the arithmetic mean along the specified axis.
標準差	np.std()	Compute the standard deviation along the specified axis.
變異數	np.var()	Compute the variance along the specified axis.
	np.sum()	Sum of array elements over a given axis.
乘積和	np.prod()	Return the product of array elements over a given axis.
累積和	np.cumsum()	Return the cumulative sum of the elements along a given axis.
累積乘	np.cumprod()	Return the cumulative product of elements along a given axis.
	np.min(), np.max()	Return the minimum / maximum of an array or minimum along an axis.
最大/小值位置	np.argmin(), np.argmax()	Returns the indices of the minimum / maximum values along an axis
	np.all()	Test whether all array elements along a given axis evaluate to True.
	np.any()	Test whether any array element along a given axis evaluates to True.

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

一維陣列Series

※所有的二維資料最終都會回到一維資料來操作，資料要盡量做成tidy data(每一列只有一個觀察值/每一欄型態相同/只講一類事)

把資料分三塊，列索引(index)，行索引(欄位column)，以及中間的Data

	ID	name	sex	email	第1次平時考	第2次平時考	第3次平時考	第4次平時考	第5次平時考
0	1080001.0	丁軒軒	男	1080001@sun.tc.edu.tw	86.0	88.0	82.0	87.0	92.0
1	1080002.0	王倫樺	女	1080002@sun.tc.edu.tw	92.0	100.0	95.0	99.0	96.0
2	1080003.0	何宜敏	女	1080003@sun.tc.edu.tw	82.0	87.0	86.0	82.0	82.0
3	1080004.0	何志陞	男	1080004@sun.tc.edu.tw	91.0	92.0	99.0	91.0	92.0
4	1080005.0	吳一歌	女	1080005@sun.tc.edu.tw	93.0	NaN	92.0	97.0	98.0

pd.Series(序列) ：Series物件建構式(Constructor)，會自動生成列索引，如傳入資料為一個字典，會自動將字典的key值轉為索引

```
c  =  ['國文','英文','數學','資訊科技']
s  =  [84,92,88,95]
cs  =  pd.Series(c)
ss  =  pd.Series(s)
print(cs)
print(ss)
```

```
0      國文
1      英文
2      數學
3    資訊科技
dtype: object
0    84
1    92
2    88
3    95
dtype: int64
```

```
c = ['國文','英文','數學','資訊科技']
s = [84,92,88,95]
cs = pd.Series(c)
ss = pd.Series(s)
print(cs)
print(ss)
```

```
# 傳入一個字典
S3 = pd.Series({"a":1,"b":2,"c":3,"d":4})
S3
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

一維陣列Series

pd.Series(s, index=序列)：可以自己定義列索引的編號，也可以用字串(資料就會像Dict模式)

```
import  pandas  as  pd
c  =  ['國文','英文','數學','資訊科技']
s  =  [84,92,88,95]
sc  =  pd.Series(s,  index=c)
print(sc)
```

```
國文      84
英文      92
數學      88
資訊科技    95
dtype: int64
```

程式碼：

```
import pandas as pd
c = ['國文','英文','數學','資訊科技']
s = [84,92,88,95]
sc = pd.Series(s, index=c)
print(sc)
```

```
#使用內定索引
print(sc[1])
sc[1]  =  100
print(sc[1])

#使用自定索引
print(sc['英文'])
sc['英文']  =  80
print(sc['英文'])
```

```
92
100
100
80
```

可使用切片pd.Series[索引:索引] ，可跳著拿資料[1,3]，也可以從後面的資料開始拿[3,1]

```
#使用內定索引存取多個元素
ss  =  pd.Series([84,92,88,95])
print(ss[0:3])
print(ss[[1,3]])
nums=[1,3]
print(ss[nums])
```

```
0    84
1    92
2    88
dtype: int64
1    92
3    95
dtype: int64
1    92
3    95
dtype: int64
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

一維陣列Series

```
S1 = pd.Series([10,20,30,40], index = ['a','b','c','d'], name = 'SS')
S1
```

S1 = pd.Series([10,20,30,40], index = ['a','b','c','d'], name = 'SS')

```
a    10
b    20
c    30
d    40
Name: SS, dtype: int64
```

S1.index：回傳自訂索引陣列，如無自訂，則輸出結果為RangeIndex(start=0, stop=4, step=1)

雖然可以跳著建立索引，但盡量不要這樣做 index = [n for n in range(1,8,2)] ➔ Int64Index([1, 3, 5, 7], dtype='int64')

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

S1.name：回傳序列名稱，當資料型態轉為Dataframe時，此值會自動轉變為欄位名稱

```
'SS'
```

S1.values：回傳Data區內容，型態為ndarray

```
array([10, 20, 30, 40], dtype=int64)
```

S1.size：回傳序列長度，可以用BIF的len(S1)

```
4
```

S1.attrs：回傳屬性，型態為字典

```
{}
```

S1.dtype：回傳Data屬性

```
dtype('int64')
```

S1.hasnans：回傳資料內容有沒有numpy.nan(nan➔not a number，不知道要填什麼資料時可以先填這個)，不是檢查有沒有空字串或0

```
False
```

S1.shape：回傳Series的大小，型態為Tuple

```
(4,)
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

一維陣列Series

累積密度寫法

第二個值等於第一個值+第二個值
第三個值等於第二個值+第一個值
.
.
.

```
#list 累積密度
ls1 = [1,2,3,4,5]
ls2 =[]

total = 0
for i in range(len(ls1)):
    total += ls1[i]
    ls2.append(total)
ls2
```

```
[1, 3, 6, 10, 15]
```

```
#numpy 累積密度
import numpy as np
n1 = np.arange(1,50,2)
n2 = np.array([],dtype = np.int64)
total = 0
for item in n1:
    total += item
    n2 = np.append(n2,total)
n2
```

```
[]
array([  1,   4,   9,  16,  25,  36,  49,  64,  81, 100, 121, 144, 169,
       196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625])
```

```
#Series 累積密度
import pandas as pd
import numpy as np
S1 = pd.Series([n for n in range(1,50,2)])
S2 = pd.Series([],dtype=np.int64)
total = 0
for item in S1:
    total += item
    S2 = S2.append(pd.Series([total]))
S2
```

```
0      1
0      4
0      9
0     16
0     25
0     36
0     49
0     64
0     81
0    100
0    121
0    144
0    169
0    196
0    225
0    256
0    289
0    324
0    361
0    400
0    441
0    484
0    529
0    576
0    625
dtype: int64
```

新增資料用append
但索引編號不會續編
如果要重編
需打上ignore_index = True

```
#Series 累積密度
import pandas as pd
import numpy as np
S1 = pd.Series([n for n in range(1,50,2)])
S2 = pd.Series([],dtype=np.int64)
total = 0
for item in S1:
    total += item
    S2 = S2.append(pd.Series([total]), ignore_index = True)
S2
```

```
0      1
1      4
2      9
3     16
4     25
5     36
6     49
7     64
8     81
9    100
10   121
11   144
12   169
13   196
14   225
15   256
16   289
17   324
18   361
19   400
20   441
21   484
22   529
23   576
24   625
dtype: int64
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame(建立DataFrame)與資料查看

建立方法pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)，列索引=index，行索引=columns

```
df  =  pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

df.info()：查看表格結構

```
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, 丁軒軒 to 何志陞
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   國文      4 non-null      int64
 1   英文      4 non-null      int64
 2   數學      4 non-null      int64
 3   社會      4 non-null      int64
 4   自然      4 non-null      int64
dtypes: int64(5)
memory usage: 192.0+ bytes
```

df.head(行數)　：查看表格前幾行，不寫預設為5行
df.tail(行數)　：查看表格後幾行，不寫預設為5行
df.describe()　：將表格的數值型資料做統計
(總數、平均值、標準差、最小值、最大值和百分位數)
df.describe()　：將表格的數值型資料做統計
df['欄位'].dtype ()　：查看特定欄位的資料類型
df['欄位'].astype ()　：修改指定欄位的資料型態
pd.set_option()　:設置顯示的行列數，
顯示所有欄:('display.max_columns',None)
顯示所有列:('display.max_rows',None)

新增資料行：如果欄位名稱不存在，會動態新增資料，資料會直行新增，所以資料拿取方式要df[columns][index]，len(df)是回傳列索引的長度

```
df['科目']  =  c
df['分數']  =  s
df
```

	科目	分數
0	國文	84
1	英文	92
2	數學	88
3	資訊科技	95

```
for  i  in  range(len(df)):
        print(df['科目'][i],df['分數'][i])
```

```
國文 84
英文 92
數學 88
資訊科技 95
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame(建立DataFrame)

將字典變成df：以行為主填入資料

```
df=  pd.DataFrame({"a"  :[4,  5  ,  6],
                   "b"  :  [7,  8,  9],
                   "c"  :  [10,  11,  12]}
                   ,index  =  [1,  2,  3])  #將字典變成df
df
```

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df= pd.DataFrame({"a" :[4, 5 , 6],
       "b" : [7, 8, 9],
       "c" : [10, 11, 12]}
       ,index = [1, 2, 3]) #將字典變成df
df
```

將串列變成df：以列為主填入資料

```
df=  pd.DataFrame([[4,  5  ,  6],
                   [7,  8,  9],
                   [10,  11,  12]]
                   ,index  =  [1,  2,  3])  #將二維串列變為df
columns  =  ['a',  'b',  'c']
df
```

	0	1	2
1	4	5	6
2	7	8	9
3	10	11	12

```
df= pd.DataFrame([[4, 5 , 6],
       [7, 8, 9],
       [10, 11, 12]]
       ,index = [1, 2, 3]) #將串列變為df
columns = ['a', 'b', 'c']
df
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame(建立DataFrame)

自訂Index： df.index = ['第1科','第2科','第3科','第4科']

```
import  pandas  as  pd
c  =  ['國文','英文','數學','資訊科技']
s  =  [84,92,88,95]
df  =  pd.DataFrame()
df['科目']  =  c
df['分數']  =  s
df.index  =  ['第1科','第2科','第3科','第4科']
df
```

	科目	分數
第1科	國文	84
第2科	英文	92
第3科	數學	88
第4科	資訊科技	95

DatetimeIndex日期索引：pd.date_range(start = ‘2022-01-01’, end = ‘2022-01-10’)➔變成日期索引後資料很方便操作

```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08',
               '2022-01-09', '2022-01-10'],
              dtype='datetime64[ns]', freq='D')
```

data[“2022”]：取得2022年的資料

data[“2022-01”]：取得2022年1月的資料

df[df[“成交時間”] == datetime(2022,8,8)]：取得成交時間為2022年8月8號日的訂單

df[df[“成交時間”] > datetime(2022,8,8)]：取得成交時間2022年8月8號日後的訂單

data[“2022-01-01”:“2022-01-05”]：取得2022年1月1日到2022年1月5日的資料

data[“2022-01-01”:“2022-01-01”]：一天的資料要這樣打

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame(建立DataFrame)

MultiIndex多重索引df：

```
#多維索引df
df  =pd.DataFrame({"a"  :  [4  ,5,  6],
                   "b"  :  [7,  8,  9],
                   "c"  :  [10,  11,  12]},
                   index  =  pd.  MultiIndex.from_tuples([('d',  1),  ('d',2),  ('e',  2)],  names=['n',  'v']))
df
```

n	v	
d	1	4 7 10
	2	5 8 11
e	2	6 9 12

```
df =pd.DataFrame({"a" : [4 ,5, 6],
      "b" : [7, 8, 9],
      "c" : [10, 11, 12]},
      index = pd. MultiIndex.from_tuples([('d', 1), ('d',2), ('e', 2)], names=['n', 'v']))
df
```

多重索引的表格概念

		國文(a)	英文(b)	數學(c)
性別(n)	姓名(v)			
男(d)	YCC(1)	4	7	10
	ABB(2)	5	8	11
女(v)	CDD(2)	6	9	12

df.loc[“d”]：查看d群組

	a	b	c
v			
1	4	7	10
2	5	8	11

df.loc[“d”,1,:]：查看d之1群

		a	b	c
n	v			
d	1	4	7	10

df.loc[“d”,1]：查看d之1群(沒寫冒號回傳格式不同)

```
a     4
b     7
c    10
Name: (d, 1), dtype: int64
```

df.reset_index(level=0/1)：將指定的索引列(level)變為行欄位，進而使多重索引變為單一索引

Level = 0

	n	a	b	c
v				
1	d	4	7	10
2	d	5	8	11
2	e	6	9	12

Level = 1

	v	a	b	c
n				
d	1	4	7	10
d	2	5	8	11
e	2	6	9	12

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame (資料拿取df.loc[])

df.loc[0,'期別']：列索引"0"，行索引"期別" = 108年1月，資料型態為Scalar(原本內容的資料型態，此結果為String)

```
'108年1月'
```

df.loc[:,'十八尖山']：十八尖山的每列資料，資料型態為Series(一維)，也可寫成df['十八尖山'](只有要找整欄資料才可以)

```
0    138881
1    137512
2    101301
3    186020
4    160695
5    169084
```

df.loc[:,['十八尖山']]：十八尖山的每列資料，但資料型態為Dataframe(二維)

	十八尖山
0	138881
1	137512
2	101301
3	186020
4	160695
5	169084

df

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame (資料拿取df.loc[])

df.loc[:,['期別',' 十八尖山']]：行索引“期別” & “十八尖山”的每一列資料(可以跳著拿)，資料型態為Dataframe

	期別	十八尖山
0	108年1月	138881
1	108年2月	137512
2	108年3月	101301
3	108年4月	186020
4	108年5月	160695

	期別	賞蟹步道
0	108年1月	8574
1	108年2月	20629

df.loc[:,['期別',' 賞蟹步道']]

df.loc[:,'期別':'青草湖']：行索引“期別” 到 “青草湖”的每一列資料，資料型態為Dataframe

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖
0	108年1月	138881	22226	186932	226935	8574	14436
1	108年2月	137512	40034	251404	219437	20629	16169
2	108年3月	101301	32227	157317	241011	20187	12116
3	108年4月	186020	64880	205830	272600	29660	17830
4	108年5月	160695	30172	212146	309255	24648	15267

df

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame (資料拿取df.loc[])

df.loc[3,:]：列索引3的每一行資料，資料型態為Series(一維)

```
期別              108年4月
十八尖山            186020
青青草原             64880
城隍廟             205830
新竹漁港            272600
賞蟹步道             29660
青草湖              17830
十七公里腳踏車道        18440
新竹公園             26210
Name: 3, dtype: object
```

df.loc[[3],:]：列索引3的每一行資料，資料型態為DataFrame(二維)

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

df.loc[[3,5],:]：列索引3跟5的每一行資料，資料型態為DataFrame(二維)

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210
5	108年6月	169084	31216	187465	331952	28083	20814	25789	20129

df

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame (資料拿取df.iloc[] - 用index拿取)

df.iloc[1,2]：列索引位置1,行索引位置2的資料，資料型態為Scalar(原本內容的資料型態，此結果為numpy.int64)

```
40034
```

df.iloc[:,2]：行索引2的每列資料，資料型態Series(一維陣列)

```
0    22226
1    40034
2    32227
3    64880
4    30172
5    31216
```

df.iloc[:,[2]]：行索引2的每列資料，資料型態Dataframe(二維陣列)

	青青草原
0	22226
1	40034
2	32227
3	64880
4	30172
5	31216

df

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame (資料拿取df.iloc[] - 用index拿取)

df.iloc[:,[2,1]]：行索引2跟1的每列資料，資料型態Dataframe(二維陣列)

	青青草原	十八尖山
0	22226	138881
1	40034	137512
2	32227	101301
3	64880	186020
4	30172	160695
5	31216	169084

df.iloc[:,:3]：行索引0~2的每列資料，資料型態Dataframe(二維陣列)

	期別	十八尖山	青青草原
0	108年1月	138881	22226
1	108年2月	137512	40034
2	108年3月	101301	32227
3	108年4月	186020	64880
4	108年5月	160695	30172
5	108年6月	169084	31216

df

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame (資料拿取df.iloc[] - 用index拿取)

df.iloc[1,:]：列索引1的每行資料，資料型態Series(一維陣列)

期別	108年2月
十八尖山	137512
青青草原	40034
城隍廟	251404
新竹漁港	219437
賞蟹步道	20629
青草湖	16169
十七公里腳踏車道	17081
新竹公園	42939

df.iloc[[1],:]：列索引1的每行資料，資料型態Dataframe(二維陣列)

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939

df.iloc[[3,1],:]：列索引3跟1的每行資料，資料型態Dataframe(二維陣列)

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939

df.iloc[1:3,:]：列索引1~2的每行資料，資料型態Dataframe(二維陣列)

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910

df

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame，操作列索引(index)

df.index：查看索引

```
Index(['丁軒軒', '王倫樺', '何宜敏', '何志陞'], dtype='object')
```

各種索引型態：

RangeIndex(start = 0, stop = 5, step =1)

Index(['A', 'B', 'C', 'D'])

Int64Index([1001,1002,1003,1004,1005])

df.reset_index()：重新用數值索引為列索引，只留下新的索引加上drop = True

Before

	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫樺	92	100	95	99	96
何宜敏	82	87	86	82	82
何志陞	91	92	99	91	92

reset_index()

	index	國文	英文	數學	社會	自然
0	丁軒軒	86	88	82	87	92
1	王倫樺	92	100	95	99	96
2	何宜敏	82	87	86	82	82
3	何志陞	91	92	99	91	92

reset_index(drop = True)

	國文	英文	數學	社會	自然
0	86	88	82	87	92
1	92	100	95	99	96
2	82	87	86	82	82
3	91	92	99	91	92

df.index.name = '指定名稱'：指定列索引的名字

學生姓名
丁軒軒
王倫樺
何宜敏
何志陞

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame，操作列索引(index)

df.set_index('欄位')：將欄資料設定為列索引

	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫樺	92	100	95	99	96
何宜敏	82	87	86	82	82
何志陞	91	92	99	91	92

	國文	數學	社會	自然
英文				
88	86	82	87	92
100	92	95	99	96
87	82	86	82	82
92	91	99	91	92

df.set_index(['欄位','欄位'])：將多個欄位變成多重索引

		英文	社會	自然
國文	數學			
86	**82**	88	87	92
92	**95**	100	99	96
82	**86**	87	82	82
91	**99**	92	91	92

df.sort_index()：將列索引資料由小排到大

	國文	數學	社會	自然
英文				
87	82	86	82	82
88	86	82	87	92
92	91	99	91	92
100	92	95	99	96

df.sort_values('欄位')：依照指定欄位資料將列索引小排到大

	國文	英文	數學	社會	自然
何宜敏	82	87	86	82	82
丁軒軒	86	88	82	87	92
何志陞	91	92	99	91	92
王倫樺	92	100	95	99	96

df.sort_values(['欄位,'欄位'])：多欄排序

	國文	英文	數學	社會	自然
何宜敏	82	87	86	82	82
丁軒軒	86	88	82	87	92
何志陞	91	92	99	91	92
王倫樺	92	100	95	99	96

●空值會排在最後，如果要排在最前面，加上參數 na_position = 'first'

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

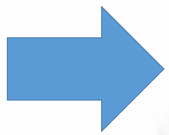
二維陣列DataFrame，操作列索引(index)

df.index.format：修改列索引資料型態

(常用於股票資料抓取，因為一開始抓下來的列索引是日期格式，但是台灣股票六日不開盤，繪圖時資料會不連續，所以要改成字串型態)

```
df_2330.index = df_2330.index.format(formatter=lambda x: x.strftime('%Y-%m-%d'))
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 654 entries, 2020-01-02 to 2022-09-07
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   High       654 non-null    float64
 1   Low        654 non-null    float64
 2   Open       654 non-null    float64
 3   Close      654 non-null    float64
 4   Volume     654 non-null    int64
 5   Adj Close  654 non-null    float64
dtypes: float64(5), int64(1)
memory usage: 35.8 KB
```



```
<class 'pandas.core.frame.DataFrame'>
Index: 654 entries, 2020-01-02 to 2022-09-07
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   High       654 non-null    float64
 1   Low        654 non-null    float64
 2   Open       654 non-null    float64
 3   Close      654 non-null    float64
 4   Volume     654 non-null    int64
 5   Adj Close  654 non-null    float64
dtypes: float64(5), int64(1)
memory usage: 35.8+ KB
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

二維陣列DataFrame，操作行欄位(columns)

df.columns：查看行欄位(一維陣列)

```
Index(['國文', '英文', '數學', '社會', '自然'], dtype='object')
```

df.columns.name = '名稱'：指定行欄位名稱

```
df.columns.name  = '科目'
df
```

科目	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫樺	92	100	95	99	96
何宜敏	82	87	86	82	82
何志陞	91	92	99	91	92

df.columns = ['a', 'b'] ：等號後方的名稱數量跟行欄位一樣多的話相等於修改欄位名稱

```
>>> df = pd.DataFrame({'$a':[1,2], '$b': [10,20]})
>>> df
   $a  $b
0   1  10
1   2  20

>>> df.columns = ['a', 'b']
>>> df
   a   b
0  1  10
1  2  20
```

df.rename(columns={'舊欄位名稱' : '新欄位名稱'}) ：修改指定欄位名稱，

```
df1 = df.rename(columns = {'國文':'文科','數學':'理科'})
df1
```

	文科	英文	理科	社會	自然
丁軒軒	86	88	82	87	92
王倫樺	92	100	95	99	96
何宜敏	82	87	86	82	82
何志陞	91	92	99	91	92

df.drop(columns = '行欄位', index ='列索引') ：刪除指定索引和欄位

	新訂單編號	新客戶姓名	唯一識別碼	成交時間
一	A1	張通	101	2018-08-08
二	A2	李谷	102	2018-08-09
三	A3	孫鳳	103	2018-08-10
4	A3	孫鳳	103	2018-08-10
5	A4	趙桓	104	2018-08-11
6	A5	趙桓	104	2018-08-12

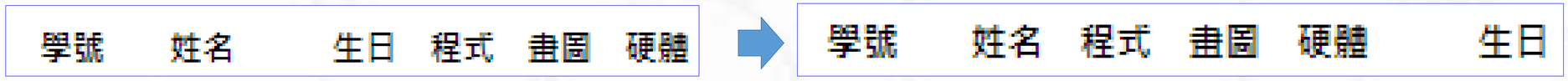
```
df.rename(columns = {"訂單編號":"新訂單編號",
                     "客戶姓名":"新客戶姓名"},
                index = {1:"一",
                         2:"二",
                         3:"三"})
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

行欄位(columns)重新安排

df.loc[:,[“學號”,“姓名”,“程式”,“畫圖”,“硬體”,“生日”]]：利用loc指定欄位把順序調換並存為另外一個df



df.T ：轉置➔行變列，列變行

	0	1	2	3
學號	S17	S09	S06	S07
姓名	劉家銘	王舒琦	溫孟芳	張馨之
生日	82-02-20	84-04-05	86-08-27	85-05-20
程式	95	94	93	92
畫圖	90	76	77	78
硬體	90	81	82	83

	學號	姓名	程式	畫圖	硬體	生日
0	S17	劉家銘	95	90	90	82-02-20
1	S09	王舒琦	94	76	81	84-04-05
2	S06	溫孟芳	93	77	82	86-08-27
3	S07	張馨之	92	78	83	85-05-20
4	S10	張家翊	91	79	84	83-03-26
5	S19	陳佳鈴	90	80	85	82-06-18

索引重塑

df.stack() ：把資料拉值打散變成多重索引，索引(0,學號) =S17, 索引(15,姓名) = 陳竹儀

```
0   學號          S17
    姓名          劉家銘
    生日     82-02-20
    程式           95
    畫圖           90
             ...
15  姓名          陳竹儀
    生日     84-08-20
    程式           50
    畫圖           40
    硬體           81
Length: 96, dtype: object
```

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

rank排序(將數據排名，數值最小的排名最高)

df['名次']=df['平均成績'].rank(method = 'max'):同分則並列排名，但依照最小的排名排，2,3名同分，兩個排名皆為3

	學號	姓名	生日	程式	畫圖	硬體	平均成績	名次
0	S17	劉家銘	82-02-20	95	75	80	83.333333	1.0
1	S09	王舒琦	84-04-05	94	76	81	83.666667	3.0
15	S20	陳竹儀	84-08-20	80	90	81	83.666667	3.0

df['名次']=df['平均成績'].rank(method = 'min'):同分則並列排名，但依照最小的排名排，2,3名同分，兩個排名皆為2

	學號	姓名	生日	程式	畫圖	硬體	平均成績	名次
0	S17	劉家銘	82-02-20	95	75	80	83.333333	1.0
1	S09	王舒琦	84-04-05	94	76	81	83.666667	2.0
15	S20	陳竹儀	84-08-20	80	90	81	83.666667	2.0

df['名次']=df['平均成績'].rank(method = 'average'):同分取排名平均，2,3名同分則排名變為(2+3)/2=2.5

	學號	姓名	生日	程式	畫圖	硬體	平均成績	名次
0	S17	劉家銘	82-02-20	95	75	80	83.333333	1.0
1	S09	王舒琦	84-04-05	94	76	81	83.666667	2.5
15	S20	陳竹儀	84-08-20	80	90	81	83.666667	2.5

df['名次']=df['平均成績'].rank(method = 'first'):同分依照索引值順序排名

	學號	姓名	生日	程式	畫圖	硬體	平均成績	名次
0	S17	劉家銘	82-02-20	95	75	80	83.333333	1.0
1	S09	王舒琦	84-04-05	94	76	81	83.666667	2.0
15	S20	陳竹儀	84-08-20	80	90	81	83.666667	3.0

df['名次']=df['平均成績'].rank(method = 'min', ascending = False):數值最大的排名優先加上ascending = False

	學號	姓名	生日	程式	畫圖	硬體	平均成績	名次
0	S17	劉家銘	82-02-20	95	75	80	83.333333	16.0
1	S09	王舒琦	84-04-05	94	76	81	83.666667	14.0
15	S20	陳竹儀	84-08-20	80	90	81	83.666667	14.0

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

更改資料

df2 = df.copy()：產生副本

df	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫華	92	100	95	99	96
何宜敏	82	87	86	82	82
陳志昇	91	92	99	91	92

df2	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫華	92	100	95	99	96
何宜敏	82	87	86	82	82
陳志昇	91	92	99	91	92

df2.loc[‘丁軒軒’,‘英文’] = 100：一次改一格資料

	國文	英文	數學	社會	自然
丁軒軒	86	100	82	87	92
王倫華	92	100	95	99	96
何宜敏	82	87	86	82	82
陳志昇	91	92	99	91	92

df2.loc[[‘丁軒軒’,‘王倫華’],‘自然’] = 59：一次改多格資料(同一欄)

	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	59
王倫華	92	100	95	99	59
何宜敏	82	87	86	82	82
陳志昇	91	92	99	91	92

用替換值來更改資料

df.replace(82,101)：針對全部

	國文	英文	數學	社會	自然
丁軒軒	86	88	101	87	92
王倫華	92	100	95	99	96
何宜敏	101	87	86	101	101
陳志昇	91	92	99	91	92

df.replace(82,101)：針對某一欄

	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫華	92	100	95	99	96
何宜敏	82	87	86	82	101
陳志昇	91	92	99	91	92

df['平均成績']= (df['程式']+df['畫圖']+df['硬體'])/3 ：針對某一欄用運算式更改資料(欄位不存在會自動新增)

	學號	姓名	生日	程式	畫圖	硬體	平均成績
0	S17	劉家銘	82-02-20	95	75	80	83.333333
1	S09	王舒琦	84-04-05	94	76	81	83.666667
2	S06	溫孟芳	86-08-27	93	77	82	84.000000
3	S07	張馨之	85-05-20	92	78	83	84.333333
4	S10	張家翊	83-03-26	91	79	84	84.666667

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

利用邏輯索引更改資料

s_bool =(df2["英文"] > 90) & (df2["自然"] > 90)
df2.loc[s_bool, “數學”] = 0：利用布林陣列修改資料

	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫華	92	100	0	99	96
何宜敏	82	87	86	82	82
陳志昇	91	92	0	91	92

把英文跟自然分數都超過90分的數學分數改為0

df2[df2 > 95] = 101：改所有符合條件的格子

	國文	英文	數學	社會	自然
丁軒軒	86	88	82	87	92
王倫華	92	101	95	101	101
何宜敏	82	87	86	82	82
陳志昇	91	92	101	91	92

※必須所有資料型態都相同才可用

新增欄位來更改資料

df2['健康'] = 60：新增欄位來更改資料

	國文	英文	數學	社會	自然	健康
丁軒軒	86	88	82	87	92	60
王倫華	92	100	95	99	96	60
何宜敏	82	87	86	82	82	60
陳志昇	91	92	99	91	92	60

pandas的邏輯運算			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(vlaues)	Group membership(多條件)
==	Equals	pd.isnull(obj)	Is NaN
<=	Less than or equals	pd.notnull(obj)	Is not NaN
>=	Greater than or equals	&, |, ~, ^, df.any(), df.all()	Logical and, or, not, xor, any, all

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

重複資料的處理

	訂單編號	客戶姓名	唯一識別碼	成交時間
0	A1	張通	101	2018-08-08
1	A2	李谷	102	2018-08-09
2	A3	孫鳳	103	2018-08-10
3	A3	孫鳳	103	2018-08-10
4	A4	趙桓	104	2018-08-11
5	A5	趙桓	104	2018-08-12

df.drop_duplicates()：移除重複資料

	訂單編號	客戶姓名	唯一識別碼	成交時間
0	A1	張通	101	2018-08-08
1	A2	李谷	102	2018-08-09
2	A3	孫鳳	103	2018-08-10
4	A4	趙桓	104	2018-08-11
5	A5	趙桓	104	2018-08-12

df.drop_duplicates(subset = “唯一識別碼”)：移除指定欄位重複資料

	訂單編號	客戶姓名	唯一識別碼	成交時間
0	A1	張通	101	2018-08-08
1	A2	李谷	102	2018-08-09
2	A3	孫鳳	103	2018-08-10
4	A4	趙桓	104	2018-08-11

df.drop_duplicates(subset = [“客戶姓名”,“唯一識別碼”])：
指定欄位的資料需都重複資料才會刪除

	訂單編號	客戶姓名	唯一識別碼	成交時間
0	A1	張通	101	2018-08-08
1	A2	李谷	102	2018-08-09
2	A3	孫鳳	103	2018-08-10
4	A4	趙桓	104	2018-08-11

df.drop_duplicates(subset = [“客戶姓名”,“唯一識別碼”], keep = “last”)：
保存最後一個值，移除其餘重複資料

df.drop_duplicates(subset = [“客戶姓名”,“唯一識別碼”], keep = False)：
不保存任何重複的資料

	訂單編號	客戶姓名	唯一識別碼	成交時間
0	A1	張通	101	2018-08-08
1	A2	李谷	102	2018-08-09
3	A3	孫鳳	103	2018-08-10
5	A5	趙桓	104	2018-08-12

keep = “last”

	訂單編號	客戶姓名	唯一識別碼	成交時間
0	A1	張通	101	2018-08-08
1	A2	李谷	102	2018-08-09

keep = False

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

重複資料的處理

df.[“店名”].is_unique：檢查欄位內的資料是否獨立，回傳True/False

df.[“店名”].unique：列出欄位內到底有幾種資料(回傳np陣列)

```
array([' 四維', ' 南吉', ' 科鈺', ' 陽明', ' 新創', ' 崇德'], dtype=object)
```

df.[“銷售ID”].duplicated()：回傳一個布林陣列，True的就是有重複資料的值

```
0    False
1    False
2     True
3     True
4    False
Name: 銷售ID, dtype: bool
```

df[“銷售ID”].value_counts()：不同的資料共有幾筆(有值才會列出)

```
2    2
1    2
3    1
Name: 銷售ID, dtype: int64
```

	訂單編號	客戶姓名	唯一識別碼	年齡	成交時間	銷售ID
0	A1	張通	101	31	2018-08-08	1
1	A2	李谷	102	45	2018-08-09	2
2	A3	孫凰	103	23	2018-08-10	1
3	A4	趙桓	104	36	2018-08-11	2
4	A5	王娜	105	21	2018-08-11	3

df[“銷售ID”].value_counts(normalize = True,sort = False)：不同的資料共有幾筆(比例)，索引值由小到大排序

```
1    0.4
2    0.4
3    0.2
Name: 銷售ID, dtype: float64
```

np.bincount：列出不同資料的資料個數(會把資料範圍的最大最小都列出)

```
#1~10隨機出100次，並確認每個數字各出幾次
import random
s1= pd.Series(random.randint(1,10) for n in range(100))
x1 = np.bincount(s1, minlength = 10) #minlenth參數可指定bincount陣列的索引個數
x1 = pd.DataFrame(data = x1)#將資料轉成pandas以便察看結果
```

	0
0	0
1	9
2	4
3	9
4	11
5	15
6	7
7	9
8	15
9	11
10	10

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

pd.value_count & np.bincount用法

1~10隨機出100次，並確認每個數字各出幾次

s1

0	4
1	5
2	4
3	3
4	5
..	..
95	3
96	3
97	6
98	3
99	7

s1.value_counts()

3	23
5	22
7	20
4	19
6	16

np.bincount(s1)

	0
0	0
1	0
2	0
3	23
4	19
5	22
6	16
7	20

np.bincount(s1,minlengh=11)

	0
0	0
1	0
2	0
3	23
4	19
5	22
6	16
7	20
8	0
9	0
10	0

pd.value_counts():亂數中沒出現的數字不會顯示出來

np.bincount():會列出0~亂數中最大數字的資料➔最大7，則只會顯示索引值為0->7的所有數字資料；可以額外增加minlenth參數將最大值擴充

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

Handling Missing Data處理空格資料

df.dropna()：只要有空格的，整列移除，可加上how = all參數值，整列都為空才會移除

df.isna()：產生布林二維陣列，有空格的格子為True

df.fillna(0)：將空格資料填為0

NaN　　：數值空值(Not a number)
pd.NaT ：日期格式空值
None　 ：文字空值

np.nap == np.nan ➔False
numpy的nan都會內定為一個不一樣的浮點數，所以會得到False

None == None ➔ True

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

集合運算

df1

	name	age	sex
0	a	10	男
1	b	11	男
2	c	11	女
3	a	10	女
4	c	11	男

df2

	name	age	sex
1	b	11	男
2	c	11	女
3	a	10	女
4	c	11	男

df1.merge (df1,df2, on= ['name', 'age', 'sex'])：取交集資料(兩個表格完全一樣的資料)

	name	age	sex
0	a	10	男

df1.merge(df2, on= ['name', 'age', 'sex'], how = 'outer')：取對稱差集資料 (兩個表格合併，並去除交集資料)

	name	age	sex
0	a	10	男
1	b	11	男
2	c	11	女
3	a	10	女
4	c	11	男
5	b	11	女

pd.concat([df1, df2], axis=0)：表格連集 (可多個)，可以指定行/列合併

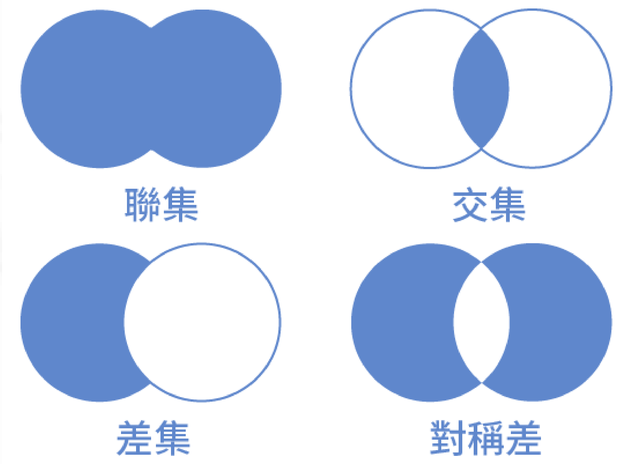
(df.merge()➔以行為主上下合併，df.join()➔以列為主，左右合併)，merge跟join只能操作兩個表格

	name	age	sex
0	a	10	男
1	b	11	男
2	c	11	女
3	a	10	女
4	c	11	男
0	a	10	男
1	b	11	女

df3 = pd.concat([df1,df2,df2], axis = 0)

df3.drop_duplicates(keep =False)：取df1差集df2資料 (df1+df2+df2後去除重複項)

	name	age	sex
1	b	11	男
2	c	11	女
3	a	10	女
4	c	11	男



Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

區間切分：將資料分割為指定區塊，資料型態為category(分類型別)

pages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32, 101]

bins = [18, 25, 35, 60, 100]：

pd.cut(ages, bins)：超出指定範圍就變成NaN

0	(18.0, 25.0]
1	(18.0, 25.0]
2	(18.0, 25.0]
3	(25.0, 35.0]
4	(18.0, 25.0]
5	(18.0, 25.0]
6	(35.0, 60.0]
7	(25.0, 35.0]
8	(60.0, 100.0]
9	(35.0, 60.0]
10	(35.0, 60.0]
11	(25.0, 35.0]
12	NaN

pd.cut(ages, 5)：也可以指定分成幾塊，
資料會依照數值範圍作分割

```
Categories (4, interval[int64]): [(18, 25] < (25, 35] < (35, 60] < (60, 100]]
```

0	(19.919, 36.2]
1	(19.919, 36.2]
2	(19.919, 36.2]
3	(19.919, 36.2]
4	(19.919, 36.2]
5	(19.919, 36.2]
6	(36.2, 52.4]
7	(19.919, 36.2]
8	(52.4, 68.6]
9	(36.2, 52.4]
10	(36.2, 52.4]
11	(19.919, 36.2]
12	(84.8, 101.0]

value_counts結果：
每個區間會差不多範圍

```
(19.919, 36.2]    8
(36.2, 52.4]      3
(84.8, 101.0]     1
(52.4, 68.6]      1
(68.6, 84.8]      0
```

pd.qcut(ages, 5)：

0	(19.999, 22.4]
1	(19.999, 22.4]
2	(22.4, 26.6]
3	(26.6, 33.0]
4	(19.999, 22.4]
5	(22.4, 26.6]
6	(33.0, 43.4]
7	(26.6, 33.0]
8	(43.4, 101.0]
9	(43.4, 101.0]
10	(33.0, 43.4]
11	(26.6, 33.0]
12	(43.4, 101.0]

value_counts結果：
每個區間的個數會差不多

```
(43.4, 101.0]     3
(26.6, 33.0]      3
(19.999, 22.4]    3
(33.0, 43.4]      2
(22.4, 26.6]      2
```

cut　：用數值區間將數值分類，可以自行指定怎麼分，如不指定則每個區間會差不多範圍

qcut ：用分位數將數值分類，每個區間分到的數量會比較平均

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

pivot(Pandas的樞紐分析:寬表格變為長表格)

df.pivot_table(index = [“日期”],values = “小計”, aggfunc= sum):aggfunc預設為mean(平均值)

日期	小計
2014-01-01	147384
2014-01-02	137300
2014-01-03	86847
2014-01-04	125190
2014-01-05	76345

df.pivot_table(index = df[“日期”].dt.month,values = “小計”, aggfunc= sum):by月份資料統計(Pandas datetime的格式取得年月pd.dt.year pd.dt.month)

日期	小計
1	2650012
2	2503070
3	2643644
4	1993056

df.pivot_table(index = df["日期"].dt.month,columns = '店名', values = "小計", aggfunc= sum):在by不同店資料分開看，加上指定columns=店名

店名	南吉	四維	崇德	新創	科廷	陽明
日期						
1	359315	188865	447320	368917	662395	623200
2	465882	462070	281550	334446	555022	404100
3	162510	487410	568317	479880	453368	492159
4	158730	584747	597830	217990	230240	203519
5	458586	323780	547470	277700	445687	763319

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

melt(取消樞紐分析:長表格變為寬表格)

```
df.melt(id_vars = "期別",
        var_name = "據點",
        value_name = "來客人數")
```

	期別	十八尖山	青青草原	城隍廟	新竹漁港	賞蟹步道	青草湖	十七公里腳踏車道	新竹公園
0	108年1月	138881	22226	186932	226935	8574	14436	7538	16902
1	108年2月	137512	40034	251404	219437	20629	16169	17081	42939
2	108年3月	101301	32227	157317	241011	20187	12116	8621	26910
3	108年4月	186020	64880	205830	272600	29660	17830	18440	26210
4	108年5月	160695	30172	212146	309255	24648	15267	26268	21465
5	108年6月	169084	31216	187465	331952	28083	20814	25789	20129

	期別	據點	來客人數
0	108年1月	十八尖山	138881
1	108年2月	十八尖山	137512
2	108年3月	十八尖山	101301
3	108年4月	十八尖山	186020
4	108年5月	十八尖山	160695

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

Series.map (map的參數要放一個函式，用法跟BIF的map一樣)

```
df['備註'] = df['平均'].map(lambda x:"及格" if(x >= 60) else "不及格" )

def grade(x):
    if x >= 90:
        return 'A'
    elif x>= 80:
        return 'B'
    elif x>= 70:
        return 'C'
    elif x>= 60:
        return 'D'
    else:
        return 'E'
df[‘等級’] = df[‘平均’].map(grade)#放在Series.map的函式不需要放參數，本來資料就會一筆一筆傳入比對
```

	學號	姓名	生日	程式	畫圖	硬體	平均	名次	備註	等級
0	S17	劉家銘	82-02-20	95	90	90	91.67	1	及格	A
1	S09	王舒琦	84-04-05	94	76	81	83.67	13	及格	B
2	S06	溫孟芳	86-08-27	93	77	82	84.00	12	及格	B
3	S07	張譽之	85-05-20	92	78	83	84.33	11	及格	B
4	S10	張家翊	83-03-26	91	79	84	84.67	9	及格	B
5	S19	陳佳鈴	82-06-18	90	80	85	85.00	7	及格	B
6	S16	吳芝儀	83-12-25	89	81	86	85.33	5	及格	B
7	S12	林千雅	84-11-10	88	82	87	85.67	3	及格	B
8	S11	張慶輝	85-10-10	87	83	88	86.00	2	及格	B
9	S08	王正豪	86-09-04	86	84	87	85.67	3	及格	B
10	S15	黃羽庭	87-07-13	85	85	86	85.33	5	及格	B
11	S18	林雅惠	88-06-23	84	86	85	85.00	7	及格	B
12	S14	楊梅	87-05-20	83	87	84	84.67	9	及格	B
13	S03	陳祐綸	86-06-20	72	78	73	74.33	14	及格	C
14	S05	楊宗源	85-07-15	60	69	62	63.67	15	及格	D
15	S20	陳竹儀	84-08-20	50	40	81	57.00	16	不及格	E

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

Dataframe.apply (函式,axis=0/1)

```
#apply的傳入值是一排一排的資料，所以函式本身的參數值要是陣列
def grade(x):
    if x['平均'] >= 90:
        return 'A'
    elif x['平均']>= 80:
        return 'B'
    elif x['平均']>= 70:
        return 'C'
    elif x['平均']>= 60:
        return 'D'
    else:
        return 'E'

def pass_(s):
    if s['平均']<60:
        return '不及格'
    else:
        return '及格'

df2 = df.copy()
df2['平均']= df2.loc[:,'程式':'硬體'].apply(np.mean,axis=1)
df2['名次']= df2['平均'].rank(method = 'min', ascending = False).astype("int64")
df2['備註']= df2.apply(pass_,axis=1)
df2['等級']= df2.apply(grade,axis=1)
```

	學號	姓名	生日	程式	畫圖	硬體	平均	名次	備註	等級
0	S17	劉家銘	82-02-20	95	90	90	91.666667	1	及格	A
1	S09	王舒琦	84-04-05	94	76	81	83.666667	13	及格	B
2	S06	溫孟芳	86-08-27	93	77	82	84.000000	12	及格	B
3	S07	張馨之	85-05-20	92	78	83	84.333333	11	及格	B
4	S10	張家翊	83-03-26	91	79	84	84.666667	9	及格	B
5	S19	陳佳鈴	82-06-18	90	80	85	85.000000	7	及格	B
6	S16	吳芝儀	83-12-25	89	81	86	85.333333	5	及格	B
7	S12	林千雅	84-11-10	88	82	87	85.666667	3	及格	B
8	S11	張慶輝	85-10-10	87	83	88	86.000000	2	及格	B
9	S08	王正豪	86-09-04	86	84	87	85.666667	3	及格	B

Pandas模組

(Series(序列：一維資料) DataFrame(框架：二維資料))

Dataframe.applymap (函式)

```
#applymap可以把二維陣列的值一個一個取出
def color_(v, f_color, bg_color):
   return f"color: {f_color}; background-color:{bg_color};" if v == 'B' else None

df2.style.applymap(color_, f_color = 'white', bg_color = 'red')
```

	學號	姓名	生日	程式	畫圖	硬體	平均	名次	備註	等級
0	S17	劉家銘	82-02-20	95	90	90	91.666667	1	及格	A
1	S09	王舒琦	84-04-05	94	76	81	83.666667	13	及格	B
2	S06	溫孟芳	86-08-27	93	77	82	84.000000	12	及格	B
3	S07	張譽之	85-05-20	92	78	83	84.333333	11	及格	B
4	S10	張家翊	83-03-26	91	79	84	84.666667	9	及格	B
5	S19	陳佳鈴	82-06-18	90	80	85	85.000000	7	及格	B
6	S16	吳芝儀	83-12-25	89	81	86	85.333333	5	及格	B
7	S12	林千雅	84-11-10	88	82	87	85.666667	3	及格	B
8	S11	張慶輝	85-10-10	87	83	88	86.000000	2	及格	B

Pandas模組

(檔案讀取)

pd.read_excel(Excel檔名):讀取Excel檔；；存檔df.to_excel(‘路徑’)

```
str1 = '/content/MyGoogleDrive/MyDrive/Colab Notebooks/Python-for-Titanic/Ch03/新竹市重要遊憩據點遊客人次統計.xlsx'
df = pd.read_excel(str1) #也可以用usecols只讀指定欄位➔ pd.read_excel(str1, usecols = [0,1,3])
```

pd. read_csv (csv檔名):讀取csv檔，中文編碼常有問題，要確認文件編碼是什麼並去設定encoding；存檔df.to_csv(‘路徑’)

```
str1 = '/content/MyGoogleDrive/MyDrive/Colab Notebooks/Python-for-Titanic/Ch03/新竹市重要遊憩據點遊客人次統計.csv'
df = pd.read_csv(str1,encoding = ‘Big5’,sep=“;”, nrows = 2,header=, skipfooter =5,thousands =‘,’)#讀取後如字串格式為數值格式，會自動轉換，sep指定分隔符號，預設是使用逗號做切割，nrows指定讀取列數，header以指定列資料當索引,skipfooter濾除不要的尾端資料，thousands可將數值的千分位逗號移除(有千分位讀進來會被判定為object型態)
```

pd. read_json(json檔名):讀取json檔

```
str1 = '/content/MyGoogleDrive/MyDrive/Colab Notebooks/Python-for-Titanic/Ch03/新竹市重要遊憩據點遊客人次統計.json'
df = pd.read_json(str1)#讀進來會都是string，要記得將數值換成正確的type
```

pd. read_xml (xml檔名):讀取xml檔中文標籤常有問題，改用xml.etree.ElementTree模組讀取

```
str1 = '/content/MyGoogleDrive/MyDrive/Colab Notebooks/Python-for-Titanic/Ch03/新竹市重要遊憩據點遊客人次統計.xml’
df = pd.read_xml(str1)
```

```
import xml.etree.ElementTree as ET
import pandas as pd
tree = ET.parse('/content/MyGoogleDrive/MyDrive/Colab Notebooks/Python-for-Titanic/Ch03/新竹市重要遊憩據點遊客人次統計.xml')
root = tree.getroot() #取得樹根Datas

list_2D = [] #建立空串列，後面用來儲存資料
columns_name = [ child.tag for child in root[0]]
#root[0]就是第一個Data標籤，用裡面的標籤名稱(期別、十八尖山……)當作欄位

for child in root:
  list_1D = [];
  for grandson in child:
    list_1D.append(grandson.text);
  list_2D.append(list_1D)

df = pd.DataFrame(data = list_2D,
      columns = columns_name)
```

```
▼<Datas>
  ▼<Data>
      <期別>108年1月</期別>
      <十八尖山>138881</十八尖山>
      <青青草原>22226</青青草原>
      <城隍廟>186932</城隍廟>
      <新竹漁港>226935</新竹漁港>
      <賞蟹步道>8574</賞蟹步道>
      <青草湖>14436</青草湖>
      <十七公里腳踏車道>7538</十七公里腳踏車道>
      <新竹公園>16902</新竹公園>
    </Data>
```

Pandas模組

(檔案讀取)

pd.read_html(網址/html格式字串):讀取網頁上的”表格”

```
url = "https://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list"
dfs = pd.read_html(url)
```

	Bank NameBank	CityCity	StateSt	CertCert	Acquiring InstitutionAI	Closing DateClosing	FundFund
0	Almena State Bank	Almena	KS	15426	Equity Bank	October 23, 2020	10538
1	First City Bank of Florida	Fort Walton Beach	FL	16748	United Fidelity Bank, fsb	October 16, 2020	10537
2	The First State Bank	Barboursville	WV	14361	MVB Bank, Inc.	April 3, 2020	10536
3	Ericson State Bank	Ericson	NE	18265	Farmers and Merchants Bank	February 14, 2020	10535
4	City National Bank of New Jersey	Newark	NJ	21111	Industrial Bank	November 1, 2019	10534

Bank Name	City	State	Cert	Acquiring Institution	Closing Date	Fund
Almena State Bank	Almena	KS	15426	Equity Bank	October 23, 2020	10538
First City Bank of Florida	Fort Walton Beach	FL	16748	United Fidelity Bank, fsb	October 16, 2020	10537
The First State Bank	Barboursville	WV	14361	MVB Bank, Inc.	April 3, 2020	10536
Ericson State Bank	Ericson	NE	18265	Farmers and Merchants Bank	February 14, 2020	10535
City National Bank of New Jersey	Newark	NJ	21111	Industrial Bank	November 1, 2019	10534
Resolute Bank	Maumee	OH	58317	Buckeye State Bank	October 25, 2019	10533

Pandas模組

(檔案讀取)

pd.read_html(網址/html格式字串):讀取網頁上的”表格”，會將讀到的表格塞到一個串列內[[df1], [df2], [df3]……..]

```
import pandas as pd
url='https://www.taiwanlottery.com.tw/Lotto/Lotto649/history.aspx'
df = pd.read_html(url)
```

網頁表格與網頁程式碼對照

大樂透

期別	開獎日	兌獎截止(註6)	銷售金額	獎金總額
111000078	111/08/26	111/11/28	88,354,950	120,791,842

獎號						特別號
開出順序	16	30	42	38	04	17	05
大小順序	04	16	17	30	38	42	05

獎金分配

項目	頭獎	貳獎	參獎	肆獎	伍獎	陸獎	柒獎	普獎
對中獎號數	6個	任5個＋特別號	任5個	任4個＋特別號	任4個	任3個＋特別號	任2個＋特別號	任3個
中獎注數	0	2	30	72	1,437	2,106	22,409	26,900
單注獎金	0	805,193	57,808	15,484	2,000	1,000	400	400
累至次期獎金	91,628,711	0	0	0				

共11列

```
<table id="Lotto649Control_history_dlQuery" cellspacing="0" style="border-collapse:collapse;">
  <tbody>
    <tr>  tr = table row
      <td>  td = table data
        <itemtemplate></itemtemplate>
        <table class="table_org td_hm">
          <tbody>
1.          <tr>…</tr>
2.          <tr>…</tr>
3.          <tr>…</tr>
4.          <tr> == $0
              <td colspan="2" class="td_org2"> 開出順序 </td>
              <td width="80" class="td_w font_black14b_center">…</td>
              <td class="td_w font_black14b_center">…</td>
              <td width="71" class="td_w font_black14b_center">…</td>
              <td width="69" class="td_w font_black14b_center">…</td>
              <td width="74" class="td_w font_black14b_center">…</td>
              <td width="72" class="td_w font_black14b_center">…</td>
              <td colspan="2" class="td_w font_red14b_center">…</td>
            </tr>
5.          <tr>…</tr>
6.          <tr>…</tr>
7.          <tr>…</tr>
8.          <tr>…</tr>
9.          <tr>…</tr>
10.         <tr>…</tr>
11.       </tbody>
        </table>
```

11列<tr>標籤

Pandas模組

(檔案讀取)

pd.read_html(網址/html格式字串):讀取網頁上的”表格”，會將讀到的表格塞到一個串列內[[df1], [df2], [df3]……..]

```
import pandas as pd
url='https://www.taiwanlottery.com.tw/Lotto/Lotto649/history.aspx'
df = pd.read_html(url)

#期別 = 從df[2]開始，每個df的[0][1]
#中獎號碼 = 每個df的[4][2]~[4][8]
#特別號 = 每個df的[4][9]
ls = []
dict1 = { str(n+1).zfill(2):0 for n in range(49)}
for n in range(2,len(df)-1):
  date = df[n].iloc[1,0]
  bingonumber = list(df[n].iloc[4,2:8]) #出來的資料是Series
  specialnumber = df[n].iloc[4,9]
  print(f'期別:{date}, 中獎號碼 : {bingonumber}, 特別號:{specialnumber}')
  ls += bingonumber


for item in ls:
  dict1[item] += 1
sortlist = sorted(dict1.items(), key = lambda x:(x[1],x[0]), reverse = True) #統計1~49的出現次數，並由大排到小
sortlist1 = sorted(dict1.items(), key = lambda x:(x[1],(-int(x[0]))), reverse = True) #統計1~49的出現次數，先由出現次數大排到小，如出現次數重複，再由數字小排到大
```

網業表格與網頁程式碼對照

Xml.etree.ElementTree模組(產生樹狀結構，建立xml檔)

Element物件、ElementTree物件、XPath語法、模組方法

```
<a><b /><c><d /></c></a>
```

```
import xml.etree.ElementTree as ET
a = ET.Element('a')
b = ET.SubElement(a, 'b')
c = ET.SubElement(a, 'c')
d = ET.SubElement(c, 'd')
ET.dump(a) #拆解讀取到的內容，將裡面的結構倒出
ET.indent(a) #自動排好看，但3.9版本才可以用
#底下沒有資料的標籤會加上”/”
```

```
<a>(root祖父)
    <b/>(爸爸)
    <c>
        <d/>(兒子)
    </c>
</a>
```

```
record
D-0 text--> 期別
D-A text--> 108年1月
D-B text--> 108年二月

record
D-0 text--> 十八尖山
D-A text--> 138881
D-B text--> 13888

record
D-0 text--> 青青草原
D-A text--> 22226
D-B text--> 2222
```

```
<Datas>
  <record>
    <D-0>期別</D-0>
    <D-A>108年1月</D-A>
    <D-B>108年二月</D-B>
  </record>
  <record>
    <D-0>十八尖山</D-0>
    <D-A>138881</D-A>
    <D-B>13888</D-B>
  </record>
  <record>
    <D-0>青青草原</D-0>
    <D-A>22226</D-A>
    <D-B>2222</D-B>
  </record>
</Datas>
```

```
import xml.etree.ElementTree as ET
datas = ET.Element('Datas')
record1 = ET.SubElement(datas, 'record')
ET.SubElement(record1, 'D-0').text = '期別'
ET.SubElement(record1, 'D-A').text = '108年1月'
ET.SubElement(record1, 'D-B').text = '108年二月'

record2 = ET.SubElement(datas, 'record')
ET.SubElement(record2, 'D-0').text = '十八尖山'
ET.SubElement(record2, 'D-A').text = '138881'
ET.SubElement(record2, 'D-B').text = '13888'

record3 = ET.SubElement(datas, 'record')
ET.SubElement(record3, 'D-0').text = '青青草原'
ET.SubElement(record3, 'D-A').text = '22226'
ET.SubElement(record3, 'D-B').text = '2222'

#element.tag 取得標籤，element.text取得標籤內容
for child in datas:
  print(child.tag)
  for grandson in child:
    print(grandson.tag, 'text-->',grandson.text)
  print()
```

Xml.etree.ElementTree模組(產生樹狀結構)
write()輸出成xml檔

讀取Excel資料並輸出成xml檔

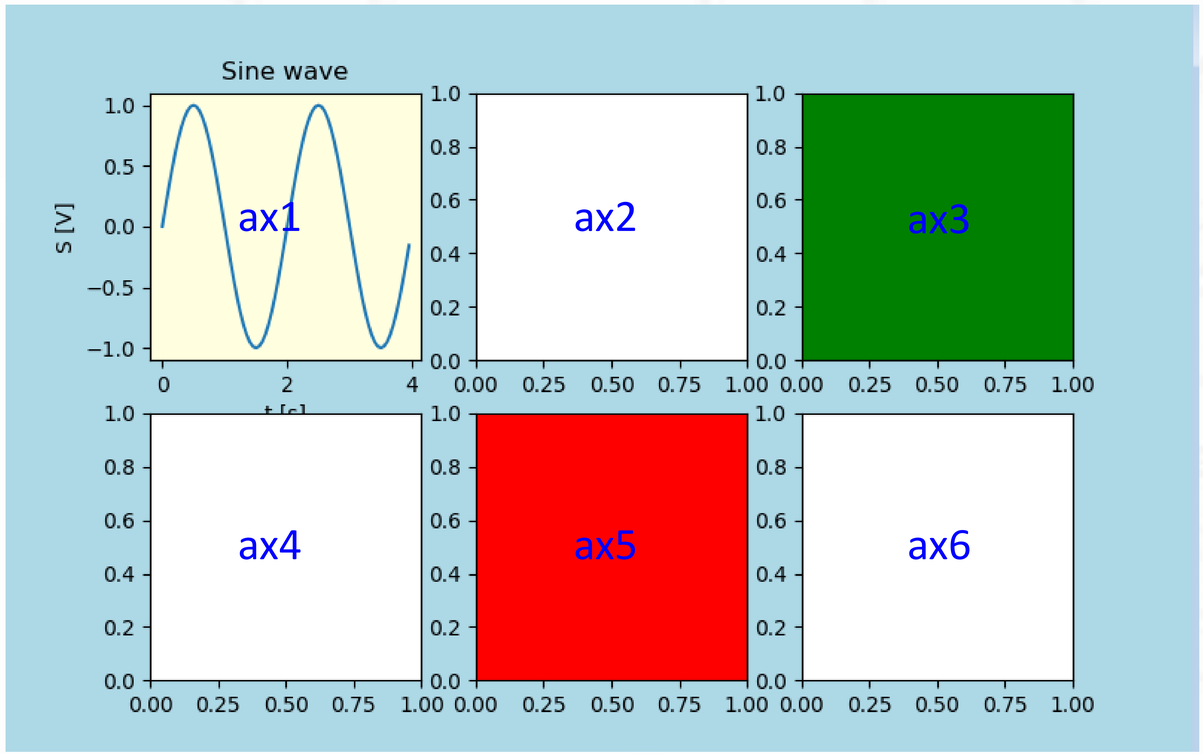
```
import xml.etree.ElementTree as ET
import pandas as pd
str1 = '/content/MyGoogleDrive/MyDrive/Colab Notebooks/Python-for-Titanic/Ch03/新竹市重要遊憩據點遊客人次統計.xlsx'
df = pd.read_excel(str1)

datas = ET.Element('Datas') #產生樹根標籤Datas
list_1D = [] #建立空串列，用來儲存record標籤

for i, col in enumerate(df.columns): #行標籤 期別、十八尖山.....
  list_1D.append(ET.SubElement(datas, 'data')) #將data標籤存入list1，並當作Datas下層標籤
  ET.SubElement(list_1D[i], 'Item' + str(0)).text = col #在data下層新增標籤為Item0，資料內容為列標籤
  for j, row in enumerate(df[col], start = 1): #每一行的每一列資料 108年1月、108年2月
    ET.SubElement(list_1D[i], 'Item' + str(j)).text = str(row) #在data下層新增每行的data
datas = ET.ElementTree(datas) #轉型為ElementTree，ElementTree物件才可使用write()函式
datas.write('新竹市重要遊憩據點遊客人次統計.xml')
```

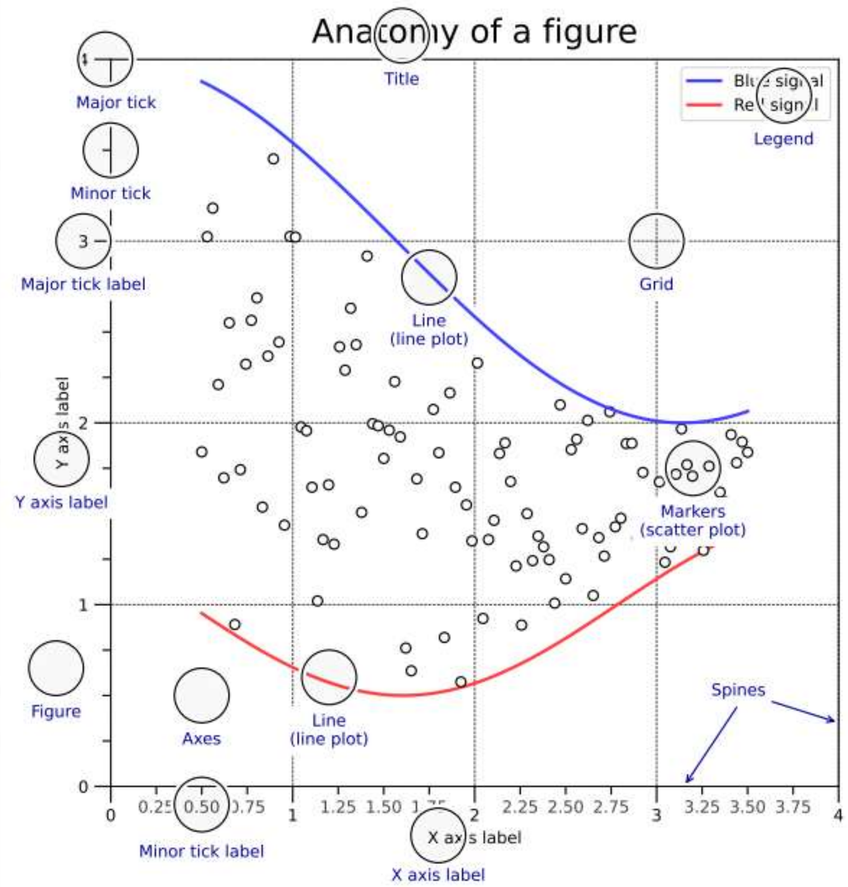
```
▼<Datas>
 ▼<data>
    <Item0>期別</Item0>
    <Item1>108年1月</Item1>
    <Item2>108年2月</Item2>
    <Item3>108年3月</Item3>
    <Item4>108年4月</Item4>
  </data>
 ▼<data>
    <Item0>十八尖山</Item0>
    <Item1>138881</Item1>
    <Item2>137512</Item2>
    <Item3>101301</Item3>
    <Item4>186020</Item4>
```

	Item0	Item1	Item2	Item3	Item4	Item5	It
0	期別	108年1月	108年2月	108年3月	108年4月	108年5月	108
1	十八尖山	138881	137512	101301	186020	160695	169
2	青青草原	22226	40034	32227	64880	30172	31
3	城隍廟	186932	251404	157317	205830	212146	187
4	新竹漁港	226935	219437	241011	272600	309255	331

matplotlib模組(畫圖)

```
fig, ((ax1,ax2,ax3),(ax4,ax5,ax6)) = matplotlib.pyplot.subplots(nrows=2, ncols=3 ) #return fig : Figure & ax : axes.Axes or array of Axes
                                                                                    #因為有2*3個小圖表，回傳的ax為2列3行的二維陣列，用解包方式
                                                                                    #將回傳的資料用前面的變數儲存，fig = 底圖，ax1~6分別為小圖表
fig.set_facecolor("lightblue")
fig.set_size_inches(10, 10,forward = True)
ax1.set_facecolor("lightyellow")
ax3.set_facecolor("green")
ax5.set_facecolor("red")

x = np.arange(0, 4, 0.05)
y = np.sin(x*np.pi)
ax1.plot(x, y) #畫圖
ax1.set_xlabel('t [s]') #Y座標標題
ax1.set_ylabel('S [V]') #X座標標題
ax1.set_title('Sine wave') #圖表標題
```





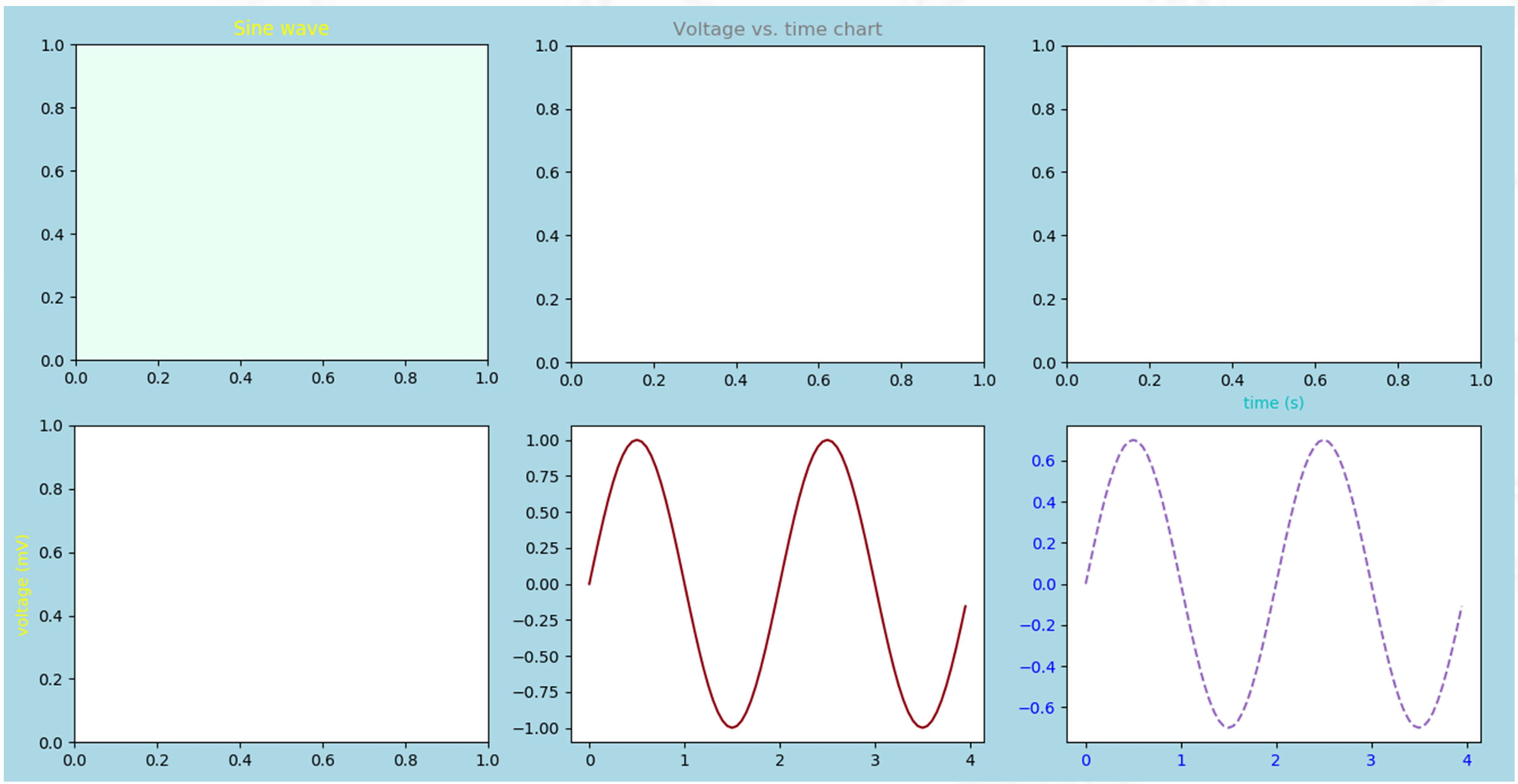
matplotlib item name

matplotlib cheat sheets

參考網址https://matplotlib.org/stable/api/index.html

matplotlib模組(畫圖)

```
ax1.set_facecolor('#eafff5)                                  #hex string 可使用16進位顏色
ax1.set_title(‘Sine wave’, color = ((255/255),(255/255),(1/255))) #也可以用r,g,b方式設置，但matplotlib是用比例去算，所以設定的數值要除255

ax2.set_title('Voltage vs. time chart', color='0.5')         #gray level string 灰階顏色只需要打一個數值即可

ax3.set_xlabel(‘time (s)’, color=‘c’)                        #single letter color string   只有[r , g, b, c, m, y, k ,w]可以用

ax4.set_ylabel('voltage (mV)', color='yellow')               # a named color 直接使用顏色名稱

ax5.plot(x, y, 'xkcd:crimson')                               # a named xkcd color

ax6.plot(x, .7*y, color=‘C4’, linestyle=‘--’)                #Cn notation matplot定義好的顏色，linestyle = ‘--’畫虛線

ax6.tick_params(labelcolor=‘blue’)                           #設置坐標軸顏色
```

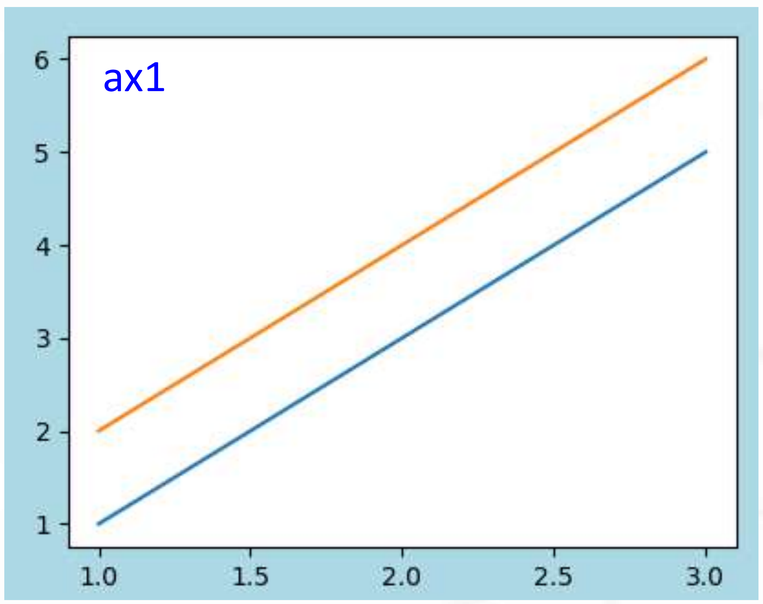


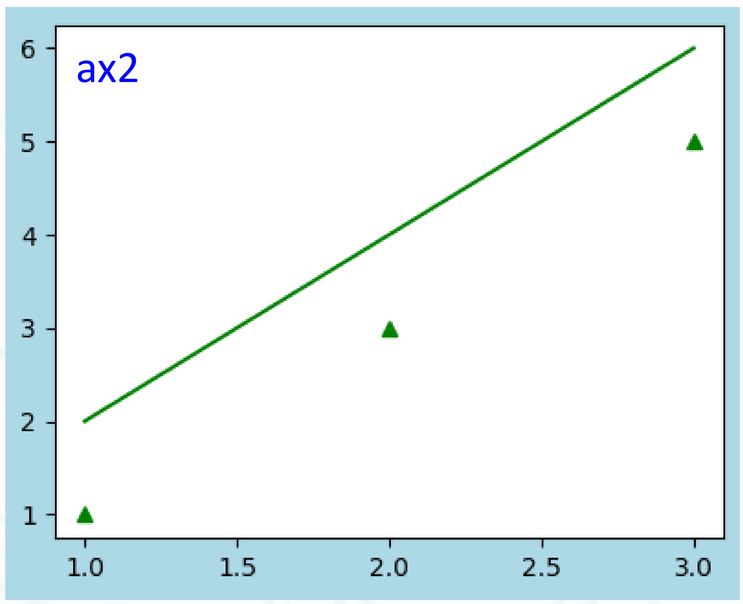
顏色參考網址https://matplotlib.org/stable/gallery/color/color_demo.html?highlight=color%20demo

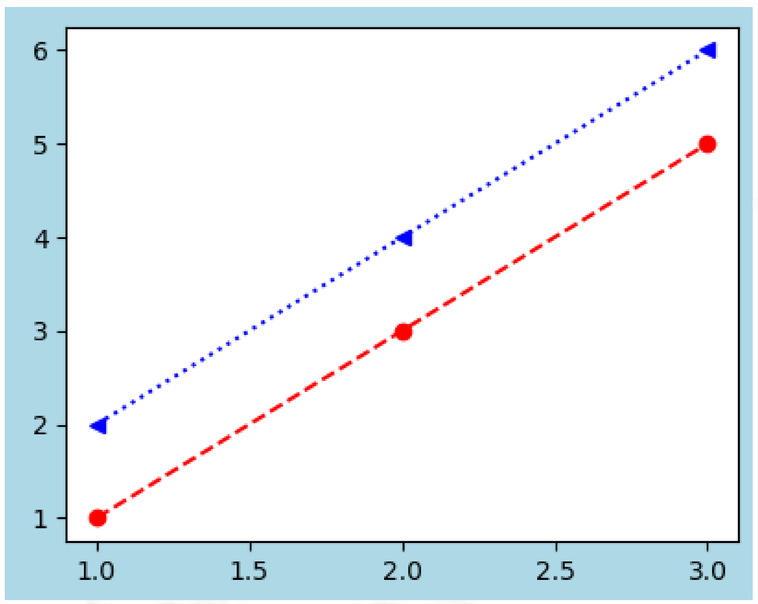
colorsmap網址https://matplotlib.org/stable/tutorials/colors/colormaps.html

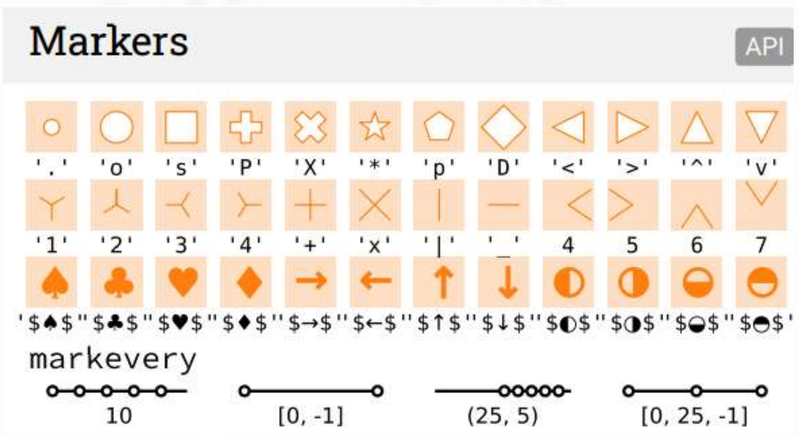
matplotlib模組(畫圖)

```
x = [1,2,3]
y = np.array([[1,2],[3,4],[5,6]])
fig, ((ax1,ax2,ax3),(ax4,ax5,ax6)) = plt.subplots(nrows = 2, ncols= 3)
fig.set_facecolor("lightblue")
fig.set_size_inches(16, 8,forward = True)

ax1.plot(x,y)                                  #因為y是二維陣列，所以會畫一條[(1,1),(2,3),(3,5)]以及一條[(1,2),(2,4),(3,6)]
ax2.plot(x, y[:,0], 'g^', x, y[:,1], 'g-')     #在對應位置畫出綠色的指定符號
ax3.plot(x, y[:,0], 'ro--', x, y[:,1], 'b<:')  #[color,mark,line]Foramt String設置要畫的圖案、圖案顏色 & 線條顏色、格式
```





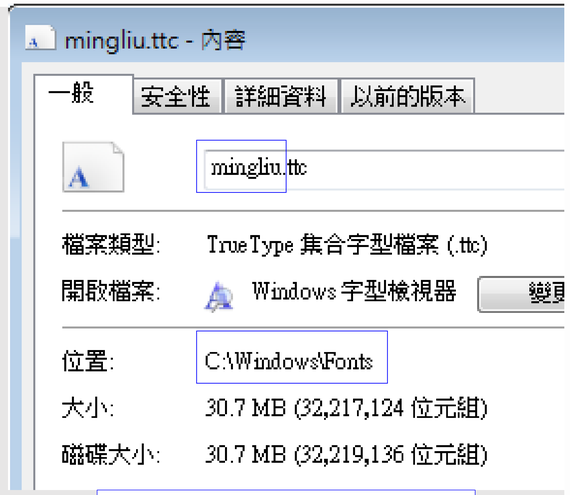




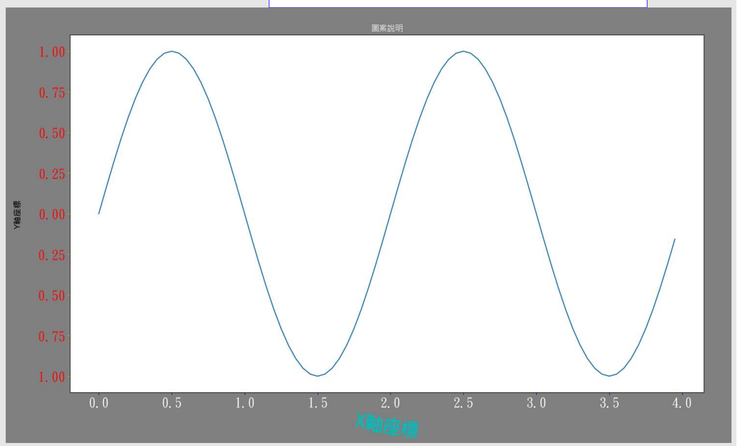
Format String參考網址https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.plot.html

matplotlib模組(畫圖)

```
#Step1. 設定整體參數，可以print(mpl.rcParams.keys())看目前設定
mpl.rcParams['font.family'] = 'DFKai-SB' #設定整體字型，電腦必須要有，包括座標軸標題字型
mpl.rcParams['font.size'] = 20 #設定整體字型大小，包括座標軸標題字型
mpl.rcParams['text.color'] = 'w' #設定標題顏色
mpl.rcParams['xtick.color'] = 'b' #設定x坐標軸刻度線顏色
mpl.rcParams['xtick.labelcolor'] = 'w' #設定x坐標軸顏色
mpl.rcParams['ytick.color'] = 'y' #設定y坐標軸刻度線顏色，預設會連坐標軸一起修改，如果要分開需再去設定labelcolor
mpl.rcParams['ytick.labelcolor'] = 'r' #設定y坐標軸刻度線顏色

#設定字體，並用變數儲存，以便後面使用
微軟正黑 = mpl.font_manager.FontProperties(fname=r'C:\Windows\Fonts\msjh.ttf',size = 12)
粉圓 = mpl.font_manager.FontProperties(fname=r'C:\Windows\Fonts\Kingnam-Maiyuan-2.otf',size = 12)
梅圓 = mpl.font_manager.FontProperties(fname=r'C:\Windows\Fonts\jf-openhuninn-1.1.ttf',size = 12)

#Step2. 產生圖紙及小圖案，然後做圖紙的設定
fig, ax1 = plt.subplots(nrows = 1, ncols= 1)
fig.set_facecolor("gray")
fig.set_size_inches(16, 8,forward = True)

#畫圖在a1小圖表
x = np.arange(0,4,0.05)
y = np.sin(x*np.pi)
ax1.plot(x,y)
ax1.set_xlabel('X軸座標',fontproperties = 粉圓, color = 'c', fontsize = 30, rotation = -10)#可以再次將先前設定的文字顏色大小角度進行修改
ax1.set_ylabel('Y軸座標',fontproperties = 梅圓)
ax1.set_title('圖案說明',fontproperties = 微軟正黑)
plt.show()
```



字型的位置與名字



整體參數設定參考網址https://matplotlib.org/stable/api/matplotlib_configuration_api.html#matplotlib.rcParams

matplotlib模組(畫圖)

```
#Step1. 設定整體參數，可以print(mpl.rcParams.keys())看目前設定
mpl.rcParams['font.family'] = 'DFKai-SB' #設定整體字型，電腦必須要有，包括座標軸標題字型
mpl.rcParams['font.size'] = 20 #設定整體字型大小，包括座標軸標題字型
mpl.rcParams['text.color'] = 'w' #設定標題顏色
mpl.rcParams['xtick.color'] = 'b' #設定x坐標軸刻度線顏色
mpl.rcParams['xtick.labelcolor'] = 'w' #設定x坐標軸顏色
mpl.rcParams['ytick.color'] = 'y' #設定y坐標軸刻度線顏色，預設會連坐標軸一起修改，如果要分開需再去設定labelcolor
mpl.rcParams['ytick.labelcolor'] = 'r' #設定y坐標軸刻度線顏色
```

yfinance模組(Yahoo財經)、mplfinance模組(財經統計圖)

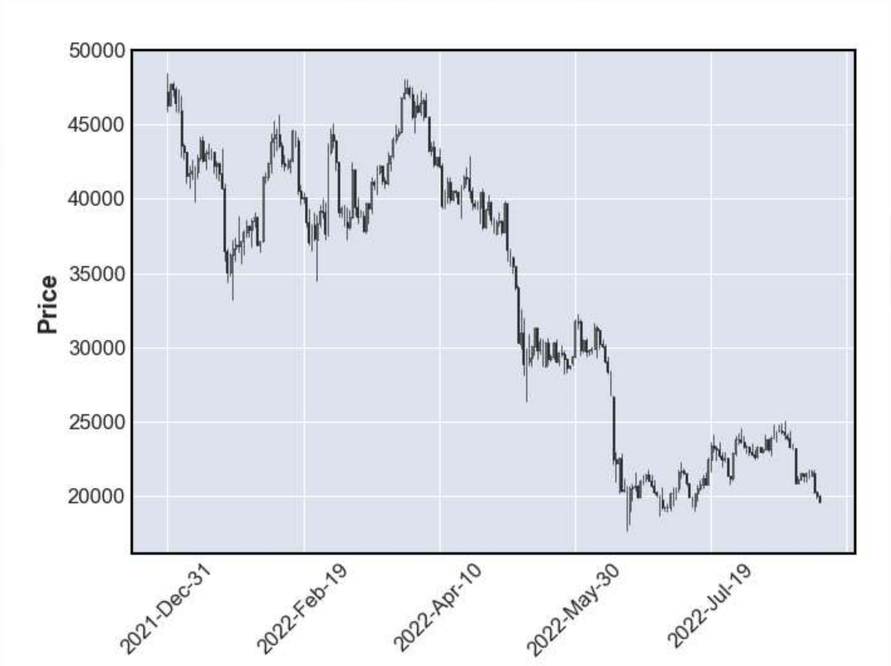
```
import yfinance as yf
import mplfinance as mpf
股票 = {"比特幣":"BTC-USC","以太幣":"ETH-USD"}
data = yf.download(tickers=股票[“比特幣”],start=“2022-01-01”,end=“2022-08-29”)  #會得到一個DataFrame，故取得的資料可以直接用DataFrame方式操作
```

yf.download(tickers = ‘雅虎財經股票代碼’, start = ‘開始日期 格式yyyy-mm-dd’, end = ‘結束日期 格式yyyy-mm-dd’

```
DatetimeIndex: 241 entries, 2021-12-31 to 2022-08-28
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       241 non-null    float64
 1   High       241 non-null    float64
 2   Low        241 non-null    float64
 3   Close      241 non-null    float64
 4   Adj Close  241 non-null    float64
 5   Volume     241 non-null    int64
dtypes: float64(5), int64(1)
```

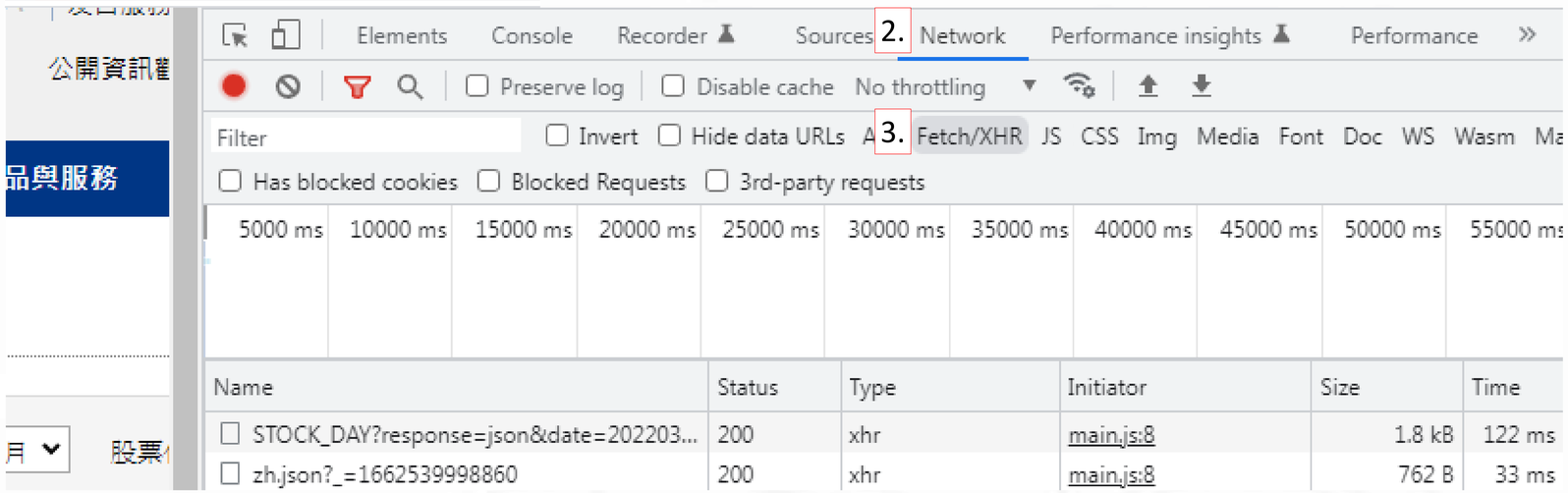
開
高
低
收
不重要
成交量

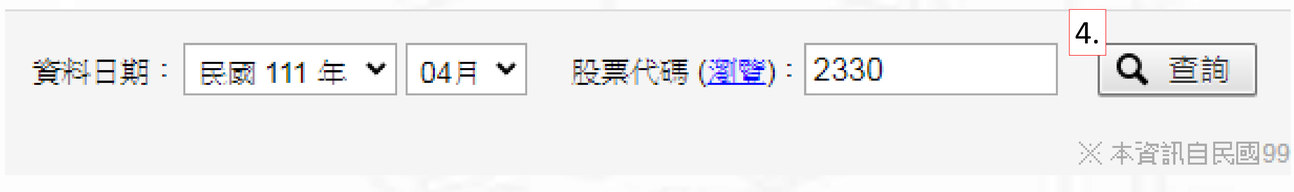
mpf.plot(data, type = ‘candle’) ：將資料畫成蠟燭圖

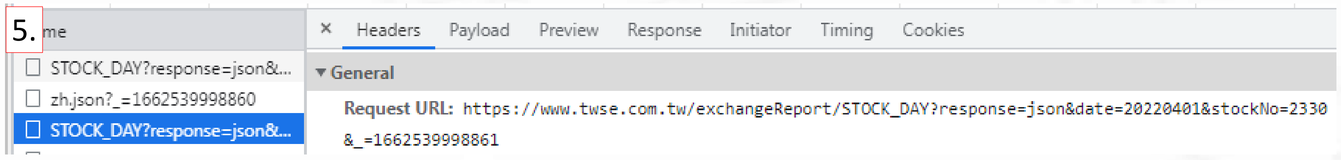


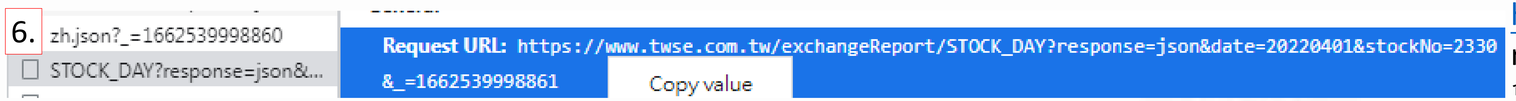
不要一次爬太多
電腦會被鎖!!!

網路爬蟲模組(Requests)









1.右鍵檢查
2.點選Network
3.查看Fetch/XHR
4.再重新用網頁查詢一次
5.得到查詢的結果
6.找到關鍵request指令並複製

https://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date=20220401&stockNo=2330&_=1662539998861
會變動的只有=後方➔後續只要修改藍字就可以得到別的資料