

Week 12

## 數據資料的整理

### 資料整理

#### ■ NumPy

- 安裝 `pip install numpy` (在scripts下)
- 使用 `import numpy`

#### ■ Pandas

- 安裝 `pip install pandas`
- 使用 `import pandas`
- `from pandas import Series`
- `from pandas import DataFrame`

## NumPy

- NumPy是Python語言的一個擴充程式庫，代表 "Numeric Python"。是一個由多維陣列物件和用於處理陣列的常式集合組成的庫，支援高階大量的維度陣列與矩陣運算。
- 針對陣列運算提供大量的數學函數函式庫，在NumPy上只要能被表示為針對陣列或矩陣運算的演算法，其執行效率幾乎都可以與編譯過的等效C語言程式碼一樣快。
- NumPy 主要用於陣列計算，包含：
  - 一個強大的N維陣列物件 ndarray
  - 廣播功能函數
  - 整合 C/C++/Fortran 代碼的工具
  - 線性代數、傅立葉轉換、亂數產生等功能

## NumPy 建立陣列與存取元素

- NumPy的核心功能是"ndarray" (即n-dimensional array，多維陣列) 資料結構。這是一個表示多維度、**同質**並且**固定大小**的陣列物件。
- 可以經由傳送串列(list)或元組(tuple)給np.array，即可建立一維陣列。也可以透過np.arange建立一維NumPy Array，np.linspace則可建立一個間隔相同的陣列。
- 一維陣列以索引(Index)存取元素，與Python索引用法相同。
- 可以經由直接傳送多維串列(list)給np.array，即可建立多維陣列，接著可以透過np.shape取得其外形(shape)。
- 也可以建立一維陣列，接著利用NumPy的reshape可以轉為所需要的各種外形。
- 多維陣列以二維Index存取元素。

## 比較

### Python

```
>>> array1 = [[1,2,3,4],[5,6,7,8]]
>>> array1 * 5
[[1, 2, 3, 4], [5, 6, 7, 8], [1, 2, 3, 4], [5, 6, 7, 8], [1, 2, 3, 4], [5, 6, 7, 8], [1, 2, 3, 4], [5, 6, 7, 8]]
>>> array1+array1
[[1, 2, 3, 4], [5, 6, 7, 8], [1, 2, 3, 4], [5, 6, 7, 8]]
>>> dir(array1)
['_add_', '_class_', '_contains_', '_delattr_', '_delitem_', '_dir_', '_doc_', '_eq_', '_format_', '_ge_', '_getattribute_', '_getitem_', '_hash_', '_iadd_', '_imul_', '_init_', '_iter_', '_le_', '_len_', '_mul_', '_ne_', '_new_', '_reduce_', '_repr_', '_reversed_', '_rmul_', '_setattr_', '_setitem_', '_sizeof_', '_str_', '_subclass_', '_append_', '_clear_', '_copy_', '_count_', '_del_', '_index_', '_insert_', '_pop_', '_remove_', '_reverse_', '_']
>>> array1
[[1, 2, 3, 4], [5, 6, 7, 8]]
>>> array1.append([5,5,5,5])
>>> array1
[[1, 2, 3, 4], [5, 6, 7, 8], [5, 5, 5, 5]]
>>> type(array1)
<class 'list'>
```

```
>>> import numpy as np
argpartition', 'argsort', 'astype', 'base', 'byteswap', 'cholesky', 'compress', 'conj', 'conjugate', 'copy', 'ctypes', 'cumprod', 'data', 'diagonal', 'dot', 'dtype', 'dump', 'dumps', 'fill', 'flat', 'flatten', 'getfield', 'imag', 'item', 'itemset', 'item', 'mean', 'min', 'nbytes', 'ndim', 'newbyteorder', 'nonzero', 'prod', 'ptp', 'put', 'ravel', 'real', 'repeat', 'reshape', 'round', 'searchsorted', 'setfield', 'setflags', 'shape', 'sort', 'squeeze', 'std', 'strides', 'sum', 'swapaxes', 'take', 'tolist', 'tolist', 'tostring', 'trace', 'transpose', 'var',
>>> array2.shape
(2, 4)
>>> array2 * 5
array([[ 5, 10, 15, 20],
       [25, 30, 35, 40]])
>>> array2
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> array2=array2.astype(str)
>>> array2
array([[ '1', '2', '3', '4'],
       [ '5', '6', '7', '8']], dtype='<U11')
>>> array2 = array2.astype(int)
>>> array2
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> array2 + array2
array([[ 2,  4,  6,  8],
       [10, 12, 14, 16]])
```

```
>>> array3=np.array([7,7,7,7])
>>> array4=np.concatenate((array2,array3))
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    array4=np.concatenate((array2,array3))
  File "<_array_function_ internals>", line 5, in concatenate
ValueError: all the input arrays must have same number of dimensions, but the array at index 0 has 2 dimension(s) and the array at index 1 has 1 dimension(s)
>>> array4=np.concatenate((array2,array2))
>>> array4
array([[1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> array4=np.concatenate((array4,array2))
>>> array4
array([[1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> array3
array([7, 7, 7, 7])
>>> array3=np.concatenate((array3,array3))
>>> array3
array([7, 7, 7, 7, 7, 7, 7, 7])
```

```

>>> array5=np.array([[7,7,7,7]])
>>> array6=np.concatenate((array4,array5))
>>> array6
array([[1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8],
       [7, 7, 7, 7]])
>>> array6=np.array(array6).reshape(4,7)
>>> array6
array([[1, 2, 3, 4, 5, 6, 7],
       [8, 1, 2, 3, 4, 5, 6],
       [7, 8, 1, 2, 3, 4, 5],
       [6, 7, 8, 7, 7, 7, 7]])
>>> array4
array([[1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> array6=np.array(array4).reshape(7,4)
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    array6=np.array(array4).reshape(7,4)
ValueError: cannot reshape array of size 24 into shape (7,4)

```

```

>>> array7=np.transpose(array6)
>>> array7
array([[1, 8, 7, 6],
       [2, 1, 8, 7],
       [3, 2, 1, 8],
       [4, 3, 2, 7],
       [5, 4, 3, 7],
       [6, 5, 4, 7],
       [7, 6, 5, 7]])
>>> np.sort([4,2,6,4,5,7])
array([2, 4, 4, 5, 6, 7])
>>> np.sort([[8,4],[2,5],[6,3]])
array([[4, 8],
       [2, 5],
       [3, 6]])
>>> np.save('nptest',array6)
>>> np.load('nptest.npy')
array([[1, 2, 3, 4, 5, 6, 7],
       [8, 1, 2, 3, 4, 5, 6],
       [7, 8, 1, 2, 3, 4, 5],
       [6, 7, 8, 7, 7, 7, 7]])

```

```

>>> array7
array([[1, 8, 7, 6],
       [2, 1, 8, 7],
       [3, 2, 1, 8],
       [4, 3, 2, 7],
       [5, 4, 3, 7],
       [6, 5, 4, 7],
       [7, 6, 5, 7]])

>>> np.shape(array7)
(7, 4)

>>> array7.shape
(7, 4)

>>> np.size(array7)
28

>>> array7.size
28

>>> array7.ndim
2

>>> array7.dtype
dtype('int32')

>>> array7.itemsize
4

```

```

>>> np.zeros
<built-in function zeros>

>>> array0=np.zeros(5)
>>> array0
array([0., 0., 0., 0., 0.])

>>> array1=np.ones(6)
>>> array1
array([1., 1., 1., 1., 1., 1.])

>>> array1.astype(int)
array([1, 1, 1, 1, 1, 1])

>>> array1
array([1., 1., 1., 1., 1., 1.])

>>> array1=array1.astype(int)
>>> array1
array([1, 1, 1, 1, 1, 1])

>>> array7.max()
8

>>> array7.mean()
4.857142857142857

>>> array7.min()
1

>>> array7.sum()
136

>>> array7.std()
2.294625486315573

>>> var=array7.std()
>>> var
2.294625486315573

```

函 式	說 明	範 例
add(A, B)	參數 A 加參數 B	>>> np.add(array1, array2) array([ 6, 8, 10, 12])
subtract(A, B)	參數 A 減參數 B	>>> np.subtract(array2, array1) array([4, 4, 4, 4])
multiply(A, B)	參數 A 乘以參數 B	>>> np.multiply(array1, array2) array([ 5, 12, 21, 32])
divide(A, B)	參數 A 除以參數 B	>>> np.divide(array2, array1) array([5., 3., 2.33333333, 2.])
mod(A, B)	參數 A 除以參數 B 的餘數	>>> np.mod(array2, array1) array([0, 0, 1, 0], dtype=int32)
power(A, B)	參數 A 的參數 B 次方	>>> np.power(array2, array1) array([ 5, 36, 343, 4096], dtype=int32)
square(A)	參數 A 的平方	>>> np.square(array1) array([ 1, 4, 9, 16], dtype=int32)
sqrt(A)	參數 A 的開根號	>>> np.sqrt([1,2,3,4]) array([1. , 1.41421356, 1.73205081, 2. ]) )
isfinite(A)	回傳參數 A 是否為有限	>>> np.isfinite(array1) array([ True, True, True, True])
isinf(A)	回傳參數 A 是否為無限	>>> np.isinf(array1) array([False, False, False, False])
isnan(A)	回傳參數 A 是否「不是數值」(not a number)	>>> np.isnan(array1) array([False, False, False, False])
sign(A)	回傳參數 A 的正負號。 -1 表示參數 A 小於 0， 0 表示參數 A 等於 0， 1 表示參數 A 大於 0。	>>> np.sign(array1) array([1, 1, 1, 1])

函 式	說 明	範 例
absolute(A)	回傳參數 A 的絕對值	>>> np.absolute([1,2,-3,-4]) array([1, 2, 3, 4])
negative(A)	回傳參數 A 的負數	>>> np.negative([1,2,-3,-4]) array([-1, -2, 3, 4])
rint(A)	回傳最接近參數 A 的整數	>>> np.rint([1.33, 2.45, 3, 4.68]) array([1., 2., 3., 5.])
floor(A)	回傳比參數 A 小 1 的整數	>>> np.floor([1.33, 2.45, 3, 4.68]) array([1., 2., 3., 4.])
ceil(A)	回傳比參數 A 大的整數	>>> np.ceil([1.33, 2.45, 3, 4.68]) array([2., 3., 3., 5.])
exp(A)	取參數 A 的指數值	>>> np.exp([1,2,3,4]) array([ 2.71828183, 7.3890561, 20.08553692, 54.59815003])
log(A)	取參數 A 的自然對數值	>>> np.log([1, 2, 3, 4]) array([0. , 0.69314718, 1.09861229, 1.38629436])
log10(A)	取參數 A 的基底 10 對數值	>>> np.log10([1, 2, 3, 4]) array([0. , 0.30103 , 0.47712125, 0.60205999])
log2(A)	取參數 A 的基底 2 對數值	>>> np.log2([1, 2, 3, 4]) array([0. , 1. , 1.5849625, 2. ]) )

函 式	說 明	範 例
sum(A)	回傳參數 A 元素的總和	>>>np.sum([1, 2, 3, 4.68]) 10.68
max(A)	回傳參數 A 的最大值	>>>np.max([1, 2, 3, 6]) 6.0
maximum(A, B)	比較參數 A 和參數 B 相對應位置的最大值	>>> np.maximum([5,1,8], [4,9,2]) array([5, 9, 8])
minimum(A, B)	比較參數 A 和參數 B 相對應位置的最小值	>>> np.minimum([5,1,8], [4,9,2]) array([4, 1, 2])
min(A)	回傳參數 A 的最小值	>>>np.min([1, 2, 3, 6]) 1.0
mean(A)	計算參數 A 的平均數	>>>np.mean([1,2,3,4]) 2.5
median(A)	計算參數 A 的中位數	>>>np.median([1,2,3,4,5]) 3.0
std(A)	計算參數 A 的標準差 (standard deviation)	>>>np.std([1,2,3,4]) 1.118033988749895
var(A)	計算參數 A 的變異數 (variance)	>>> np.var([1,2,3,4]) 1.25
sort(A)	針對參數 A 進行排序。 如果 A 是多維陣列，則是對每一列進行排序。	>>>np.sort([8,4,1,6,5,7]) array([1, 4, 5, 6, 7, 8])  >>>np.sort([[8,4],[1,6],[7,5]]) array([[4, 8], [1, 6], [5, 7]])
cos(A)、sin(A)、tan(A)、acos(A)、asin(A)、atan(A) 等三角函式	分別回傳參數 A 的餘弦值 (cosine)、正弦值 (sine)、正切值 (tangent)、反餘弦值 (arccosine)、反正弦值 (arcsine)、反正切值 (arctan)	>>>np.cos(np.array([0, 30, 45]) * (np.pi) / 180.) array([1.00000000, 0.8660254 , 0.70710678])

## 你可以做的練習

- 請使用者輸入任意個數數字，將資料存入numpy陣列中。  
如[1, 2, 3, 4]
- 將每一筆資料依照輸入順序倍數成長後存入  
如變成[1, 4, 9, 16]
- 將資料內容累加存到後面去，將資料存入檔案中  
如變成[1, 5, 14, 30]
- 再將檔案資料讀取出來



## 可以再試試NumPy 排序

- 在NumPy中快速排序方式：`np.sort`和`np.argsort`。
- 要在不修改輸入的情況下返回陣列的排序版本，可以使用`np.sort`；`argsort`返回已排序元素的索引(indices)。
- NumPy排序演算法的一個有用特性是能夠使用`axis`參數對多維陣列的特定行或列進行排序。
- `np.partition`取一個陣列和一個數字K；結果是一個新陣列，在分區左邊有最小的K個值，剩下的值任意顯示在右邊。

## Pandas 概念

- Pandas 是基於NumPy 的一種工具，該工具是為了解決資料分析任務而創建的。
- Pandas 納入了大量庫和一些標準的資料模型，提供了高效地操作大型資料集所需的工具。
- Pandas 可以刪除或插入列，資料可以自動對齊，具有靈活強大的分組功能，可對資料集進行拆分組合操作。
- 可以方便的將其他Python和NumPy資料結構中不同類索引的資料轉換為DataFrame物件，且能進行直觀的合併，連接資料集，並且輕易的重新定義資料集形狀和轉置。



## Pandas 物件概念

- Pandas提供Series與DataFrame兩種主要的數據結構，以下分別介紹這二種結構的操作方式。
- Pandas Series是索引資料的一維陣列，可以從串列或陣列創建，包含一系列值和一系列索引，可使用values和index屬性來訪問。
- Pandas Series可被看一般化的NumPy陣列，也可以當作特殊的字典。
- Pandas的DataFrame可以被視為字典的特化。當字典將鍵映射到值時，DataFrame將行名稱映射到行資料的Series。

## Pandas 索引與資料選取

- Pandas Series物件可用於一維NumPy陣列與標準的Python字典。像字典一樣，Series物件提供從一組鍵到一組值的映射，甚至可以用類似字典的語法修改。
- 若將Series當作一維陣列，通過與NumPy陣列相同的基本機制提供陣列樣式的項目選擇，即slices、masking和fancy indexing。索引採用指令loc、iloc、和ix。
- DataFrame類似二維或結構化陣列與共享相同索引的Series結構字典。
- 可將DataFrame視為增強的二維陣列。可以使用values屬性檢查原始底層資料陣列。
- Pandas使用iloc索引器，可以將底層陣列索引成好像它是一個簡單的NumPy陣列（使用隱式的Python樣式索引），但結果中保留了DataFrame索引和行標籤；使用loc索引器，我們可以使用顯式索引和行名稱，以類似陣列的樣式索引基礎資料。

```

>>> from pandas import series
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    from pandas import series
ImportError: cannot import name 'series' from 'pandas' (C:\Users\Li-Ling\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\_init_.py)
>>> from pandas import Series
>>> ser_obj=Series([1,3,5,7])
>>> ser_obj
0    1
1    3
2    5
3    7
dtype: int64
>>> ser_obj[2]
5
>>> ser_obj.values
array([1, 3, 5, 7], dtype=int64)
>>> ser_obj.index
RangeIndex(start=0, stop=4, step=1)
>>> ser_obj.index=('a','b','c','d')
>>> ser_obj
a    1
b    3
c    5
d    7
dtype: int64

```

改變index

```

>>> game_dic={1:"棒",2:"老虎",3:"雞",4:"蟲"}
>>> ser_obj1=Series(game_dic)
>>> ser_obj1
1    棒
2   老虎
3    雞
4    蟲
dtype: object
>>> ser_obj1.index
Int64Index([1, 2, 3, 4], dtype='int64')
>>> ser_obj1.values
array(['棒', '老虎', '雞', '蟲'], dtype=object)
>>> dir(ser_obj1)
Squeezed text (86 lines).
>>> ser_obj1.name='sertest'
>>> ser_obj1.index.name='GameNumber'
>>> ser_obj1
GameNumber
1    棒
2   老虎
3    雞
4    蟲
Name: sertest, dtype: object

```

直接與dict接軌

索引

資料內容

物件命名

index命名

```

>>> ser_obj1
GameNumber
1    棒
2    老虎
3    雞
4    蟲
Name: sertest, dtype: object

>>> ser_obj2=Series([2,4,6,8])
>>> ser_obj2
0    2
1    4
2    6
3    8
dtype: int64
>>> ser_obj3=Series(ser_obj1,index=ser_obj2)  整合兩個series ·
>>> ser_obj3  以其中一個值作為index
2    老虎
4    蟲
6    NaN
8    NaN
Name: sertest, dtype: object
>>> ser_obj3.isnull()  檢查那些符合空的條件
2    False
4    False
6    True
8    True
Name: sertest, dtype: bool

```

```

>>> ser_obj2.index=['a','b','c','d']
>>> ser_obj2
a    2
b    4
c    6
d    8
dtype: int64
>>> ser_obj4=Series([1,2,3,4],index=['a','b','c','d'])
>>> ser_obj4
a    1
b    2
c    3
d    4
dtype: int64

>>> ser_obj2+ser_obj4  Index相同時的運作
a    3
b    6
c    9
d   12
dtype: int64
>>> ser_obj2*ser_obj4
a    2
b    8
c   18
d   32
dtype: int64

```

```
>>> ser_obj2
a    2
b    4
c    6
d    8
dtype: int64
```

```
>>> ser_obj3
2    老虎
4    蟲
6    NaN
8    NaN
Name: ser_obj3, dtype: object
```

Index不同時的運作

```
>>> ser_obj2+ser_obj3
```

```
a    NaN
b    NaN
c    NaN
d    NaN
2    NaN
4    NaN
6    NaN
8    NaN
dtype: object
```

```
>>> ser_obj2.append(ser_obj3)
```

```
a    2
b    4
c    6
d    8
2    老虎
4    蟲
6    NaN
8    NaN
dtype: object
```

```
>>> ser_obj2
a    2
b    4
c    6
d    8
dtype: int64
```

```
>>> ser_obj2[a]
Traceback (most recent call last):
  File "<pyshell#55>", line 1, in <module>
    ser_obj2[a]
NameError: name 'a' is not defined
>>> ser_obj2['a']
2
```

## 針對索引(index)的運算: union、delete、reindex

```
>>> idx1=ser_obj1.index
>>> idx2=ser_obj2.index
>>> idx3=ser_obj3.index
>>> idx1
Int64Index([1, 2, 3, 4], dtype='int64', name='GameNumber')
>>> idx2
Index(['a', 'b', 'c', 'd'], dtype='object')
>>> idx3
Int64Index([2, 4, 6, 8], dtype='int64')
>>> idx2.union(idx3)
Index(['a', 'b', 'c', 'd', 2, 4, 6, 8], dtype='object')
>>> idx2
Index(['a', 'b', 'c', 'd'], dtype='object')
>>> idx1.union(idx3)
Int64Index([1, 2, 3, 4, 6, 8], dtype='int64')
>>> idx2.delete(3)
Index(['a', 'b', 'c'], dtype='object')
>>> ser_obj2.index.delete(3)
Index(['a', 'b', 'c'], dtype='object')
>>> ser_obj2
a      2
b      4
c      6
d      8
dtype: int64
```

```
>>> ser_obj3
2      老虎
4      蟲
6      NaN
8      NaN
Name: sertest, dtype: object
>>> ser2=ser_obj3.reindex([1,2,3,4],method = 'ffill')
>>> ser2
1      NaN
2      老虎
3      老虎
4      蟲
Name: sertest, dtype: object
>>> ser1=ser_obj3.reindex([1,2,3,4],method='bfill')
>>> ser1
1      老虎
2      老虎
3      蟲
4      蟲
Name: sertest, dtype: object
>>> ser2=ser_obj3.reindex([1,2,3,4])
>>> ser2
1      NaN
2      老虎
3      NaN
4      蟲
Name: sertest, dtype: object
```

```

>>> from pandas import DataFrame as DF
>>> stud_data={'grade':['Freshman','Sophomore','Junior','Senior','Graduate'],'year':[2019,2018,2017,2016], 'number':[78,79,68,88]}
>>> stud_frame=DF(stud_data)
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    stud_frame=DF(stud_data)
  File "C:\Users\Li-Ling\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\core\frame.py", line 435, in __init__
    mgr = init_dict(data, index, columns, dtype=dtype)
  File "C:\Users\Li-Ling\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\core\internals\construction.py", line 254, in init_dict
    return arrays_to_mgr(arrays, data_names, index, columns, dtype=dtype)
  File "C:\Users\Li-Ling\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\core\internals\construction.py", line 64, in arrays_to_mgr
    index = extract_index(arrays)
  File "C:\Users\Li-Ling\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\core\internals\construction.py", line 365, in extract_index
    raise ValueError("arrays must all be same length")
ValueError: arrays must all be same length
>>> stud_data={'grade':['Freshman','Sophomore','Junior','Senior','Graduate'],'year':[2019,2018,2017,2016,2018], 'number':[78,79,68,88,10]}
>>> stud_frame=DF(stud_data)
>>> stud_frame
   grade  year  number
0  Freshman  2019     78
1  Sophomore  2018     79
2    Junior  2017     68
3    Senior  2016     88
4  Graduate  2018     10

```

```

>>> DF(stud_data, column=['year','grade','number'])
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    DF(stud_data, column=['year','grade','number'])
TypeError: __init__() got an unexpected keyword argument 'column'
指定欄位順序

>>> DF(stud_data, columns=['year','grade','number'])
   year  grade  number
0  2019  Freshman     78
1  2018  Sophomore     79
2  2017    Junior     68
3  2016    Senior     88
4  2018  Graduate     10
多一個欄位

>>> stud_frame2=DF(stud_data, columns=['year','grade','number', 'registered'], index=['one', 'two', 'three', 'four', 'five'])
>>> stud_frame2
   year  grade  number  registered
one  2019  Freshman     78        NaN
two  2018  Sophomore     79        NaN
three 2017    Junior     68        NaN
four  2016    Senior     88        NaN
five  2018  Graduate     10        NaN
指定索引內容

```

```

>>> stud_frame2
   year  grade  number registered
one   2019  Freshman    78      NaN
two   2018  Sophomore    79      NaN
three 2017   Junior    68      NaN
four  2016   Senior    88      NaN
five  2018  Graduate    10      NaN
>>> stud_frame2['grade']
one      Freshman
two      Sophomore
three     Junior
four      Senior
five     Graduate
Name: grade, dtype: object
>>> stud_frame2.grade
one      Freshman
two      Sophomore
three     Junior
four      Senior
five     Graduate
Name: grade, dtype: object
>>> ser_obj2['a']
2
>>> ser_obj2.a
2
>>> ser_obj2
a    2
b    4
c    6
d    8

```

查詢某一欄資料

比對Series查詢某項資料

```

>>> stud_frame2.index=[1,2,3,4,5]
>>> stud_frame2
   year  grade  number registered
1  2019  Freshman    78      NaN
2  2018  Sophomore    79      NaN
3  2017   Junior    68      NaN
4  2016   Senior    88      NaN
5  2018  Graduate    10      NaN
>>> stud_frame2.index[2]
3
>>> stud_frame2.values[2]
array([2017, 'Junior', 68, nan], dtype=object)
>>> stud_frame2.index=['one','two','three','four','five']
>>> stud_frame2
   year  grade  number registered
one   2019  Freshman    78      NaN
two   2018  Sophomore    79      NaN
three 2017   Junior    68      NaN
four  2016   Senior    88      NaN
five  2018  Graduate    10      NaN
>>> stud_frame2.index == 'two'
array([False,  True, False, False, False])
>>> stud_frame2.values[1]
array([2018, 'Sophomore', 79, nan], dtype=object)
>>> stud_frame2.values[2]
array([2017, 'Junior', 68, nan], dtype=object)

```

第幾筆資料(從零開始算)



```

>>> stud_frame2
   year  grade  number  registered
one  2019  Freshman    78         NaN
two  2018  Sophomore    79         NaN
three 2017   Junior    68         NaN
four  2016   Senior    88         NaN
five  2018  Graduate    10         NaN
>>> stud_frame2['registered']=(76,75,68,80,10)
>>> stud_frame2
   year  grade  number  registered
one  2019  Freshman    78         76
two  2018  Sophomore    79         75
three 2017   Junior    68         68
four  2016   Senior    88         80
five  2018  Graduate    10         10
>>> dir()
['DF', 'Series', '_annotations_', '_builtins_', '_doc_', '_loader_', '_name',
'_package_', '_spec_', '_array_', 'game_die', 'ser_obj', 'ser_obj1', 'ser_obj2',
'_ser_obj3', 'ser_obj4', 'stud_data', 'stud_frame', 'stud_frame2']
>>> NewRegister=Series([140,120,45],index=['two','four','five'])
>>> stud_frame2['registered']=NewRegister
>>> stud_frame2
   year  grade  number  registered
one  2019  Freshman    78         NaN
two  2018  Sophomore    79        140.0
three 2017   Junior    68         NaN
four  2016   Senior    88        120.0
five  2018  Graduate    10         45.0

```

增加某一欄位的資料

宣告某個series

DataFrame新增一欄位，並將一Series的資料塞進此欄位裡

沒有資料的地方補空

增加一欄位

欄位裡的資料

```

>>> stud_frame2['enrol'] = stud_frame2.grade == 'Graduate'
>>> stud_frame2
   year  grade  number  registered  enrol
one  2019  Freshman    78         NaN  False
two  2018  Sophomore    79        140.0  False
three 2017   Junior    68         NaN  False
four  2016   Senior    88        120.0  False
five  2018  Graduate    10         45.0   True
>>> del stud_frame2['enrol']
>>> stud_frame2
   year  grade  number  registered
one  2019  Freshman    78         NaN
two  2018  Sophomore    79        140.0
three 2017   Junior    68         NaN
four  2016   Senior    88        120.0
five  2018  Graduate    10         45.0
>>> stud_frame2.T
   year  grade  number  registered
one  2019  Freshman    78         NaN
two  2018  Sophomore    79        140.0
three 2017   Junior    68         NaN
four  2016   Senior    88        120.0
five  2018  Graduate    10         45.0

```

```
>>> stud_frame2
      year  grade  number  registered
one  2019  Freshman    78         NaN
two  2018  Sophomore    79        140.0
three 2017   Junior    68         NaN
four  2016   Senior    88        120.0
five  2018  Graduate    10         45.0
>>> stud_frame3=stud_frame2.T
>>> stud_frame3['two']
year          2018
grade      Sophomore
number          79
registered      140
Name: two, dtype: object
```

```
>>> stud_frame2.index.name='lable'
>>> stud_frame2.columns.name='field'
>>> stud_frame2
field year  grade  number  registered
lable
one  2019  Freshman    78         NaN
two  2018  Sophomore    79        140.0
three 2017   Junior    68         NaN
four  2016   Senior    88        120.0
five  2018  Graduate    10         45.0
>>>
```

一樣可以幫欄位  
和index命名

## 你可以做的練習

1. 請開啟範例10-1.py檔執行
2. 請加一個計算平均的欄位
3. 請針對分數高低進行排序
4. 將所有成績資料寫進一檔案

## 再加一個練習:請把下列資料寫入一 DataFrame 中

- 請以西瓜價遞減順序印出
- 將台北一、二合併價格是取總價不變的金額
- 三重市改三重區
- 輸出最便宜的三個地區價格
- 寫入檔案

某日各果菜批發市場之西瓜與香瓜之拍賣行情（價與量）如下表：

	西瓜價	西瓜量	香瓜價	香瓜量
三重市	9.00	203674	13.20	18894
台中市	11.70	180785	12.30	54894
台北一	10.10	127802	14.70	18563
台北二	11.80	28604	14.90	21963
台東市	13.20	600	13.10	900
板橋區	6.90	38071	9.60	3555
高雄市	12.10	35660	10.60	9005
嘉義市	12.00	15000	13.00	12000
鳳山區	11.70	48770	9.10	14370
豐原區	9.84	6100	11.89	8980