帶著 Python 踏上資料科學之路！

科學運算模組基礎

numpy & pandas

在python在處理資料科學相關運用的時候，

有兩個兩個重要的模塊，一個是 numpy,一個是 pandas。

科學運算

Numpy 介紹與應用

Pandas 介紹與應用

單元目的

- 了解科學運算中 numpy & pandas 的重要性
- 了解 numpy 的語法基礎
- 培養執行 numpy 的能力
- 了解 pandas 中 series 與 dataframe 語法基礎
- 培養執行 numpy 的能力

科學運算
numpy & pandas

科學運算當中兩個重要的模塊，
一個是 numpy,一個是 pandas。

為什麼使用 numpy & pandas

- 運算速度快
  - numpy 和 pandas 都是採用 C 語言編寫, pandas 又是基於 numpy, 是 numpy 的升級版本。
- 消耗資源少
  - 採用的是矩陣運算，會比 Python 自帶的字典或者列表快好多。

numpy

可使用在矩陣運算、線性代數、矩陣乘法、轉置、數學運算、統計運算...等

https://docs.scipy.org/doc/numpy/reference/routines.html

Numpy
array

- 處理一維或多維矩陣運算
- 是固定大小的
  - 不像 Python `list` 可以動態增減
- `array` 之中的每一個元素都必須是相同型態。
- NumPy 會讓你程式碼更有效率
  - NumPy 所有元件都需要是相同大小的，因此在記憶體有相同的 Size。更適合用於數學運算與資料較龐大時的運算。

NumPy - 屬性

```
import numpy as np                    #使用numpy 採用np簡寫
array1 = np.array([1,2,3])            #一維的array
array2 = np.array([[1,2,3],[4,5,6]])  #二維的array

print(array2)
print('維度:',array2.ndim)
print('行列:',array2.shape)
print('個數:',array2.size)
```

```
[[1 2 3]
 [4 5 6]]
維度: 2
行列: (2, 3)
個數: 6
```

1D array

	column 0	column 1	column 2
row 0	1	2	**3**

shape(3,)

2D array

	column 0	column 1	column 2
row 0	1	2	**3**
row 1	4	5	**6**

shape(2,3)

NumPy - 建立一維陣列

```
import numpy as np
a = np.array([3,6,9,12,15,18])
b = np.arange(6)                    #建立一個size為6的陣列，數據為0~5
c = np.zeros(6)                     #數據全為0，size 6
d = np.ones(6, dtype = np.int)#數據為1，size 6，型態整數
e = np.random.random(6)             #size為6的陣列，隨機亂數範圍為 [0.0,1.0)

print(a,b,c,d,e, sep="\n")
```

```
[ 3  6  9 12 15 18]
[0 1 2 3 4 5]
[ 0.  0.  0.  0.  0.  0.]
[1 1 1 1 1 1]
[ 0.9038974   0.19673665  0.8186001   0.13724558  0.05995684  0.9762709 ]
```

NumPy - 一維陣列的運算(加減乘除、布林)

```
import numpy as np
a = np.array([3,6,9,12,15,18])
b = np.arange(6)
f = b-a

print(f, sep="\n")
print(f<-9)
print(f==-9)
```

```
[ -3  -5  -7  -9 -11 -13]
[False False False False  True  True]
[False False False  True False False]
```

小提醒
甲 < 乙 若條件成立則得到 True，反之False
甲 ==乙 若條件成立則得到 True，反之False

f = b-a

a	3	6	9	12	15	18
b	0	1	2	3	4	5
f	-3	-5	-7	-9	-11	-13

b[0]-a[0]　b[1]-a[1]　b[2]-a[2]

f < -9

f	-3	-5	-7	-9	-11	-13

f == -9

f	-3	-5	-7	-9	-11	-13

NumPy - 多維陣列的運算(sum)

```
import numpy as np
a = np.array([[3,6,9],[12,15,18]])
b = np.arange(6).reshape(2,3)
c = np.zeros((2,3))                    #數據全為0
d = np.ones((2,3), dtype = np.int)     #數據為1，6個
e = np.random.random((2,3))            #亂數，6個

print(a,b,c,d,e, sep="\n")
print(np.sum(a))
print(np.sum(a, axis=1))
```

```
[[ 3  6  9]
 [12 15 18]]
[[0 1 2]
 [3 4 5]]
[[ 0.  0.  0.]
 [ 0.  0.  0.]]
[[1 1 1]
 [1 1 1]]
[[ 0.59331307  0.94254093  0.06424767]
 [ 0.92768566  0.19110781  0.56701524]]
63
[18 45]
```

sum(a)

整個陣列的值加起來

a

3	6	9
12	15	18

sum(a)
= 3+6+9+12+15+18
= 63

sum(a, axis=1)

逐行的值相加

a

3	6	9
12	15	18

sum(a, axis=1)
=[(3+6+9) (12+15+18)]
=[18 45]

NumPy - 多維陣列的運算(加減乘除法 *非矩陣乘法*)

```
import numpy as np
a=np.array([[2,4,6],[8,10,12]])
b=np.arange(6).reshape((2,3))

print(a)
print(b)
print(a-b)
print(a*b)
```

```
[[ 2  4  6]
 [ 8 10 12]]
[[0 1 2]
 [3 4 5]]
[[2 3 4]
 [5 6 7]]
[[ 0  4 12]
 [24 40 60]]
```

	0	1	2
0	2	4	6
1	8	10	12

a

	0	1	2
0	0	1	2
1	3	4	5

b

a-b

a-b	a[0][0]-b[0][0]	a[0][1]-b[0][1]	a[0][2]-b[0][2]
	a[1][0]-b[1][0]	a[1][1]-b[1][1]	a[1][2]-b[1][2]

a*b 注意!這裡的a*b非矩陣乘法

a-b	a[0][0]*b[0][0]	a[0][1]*b[0][1]	a[0][2]*b[0][2]
	a[1][0]*b[1][0]	a[1][1]*b[1][1]	a[1][2]*b[1][2]

NumPy - 矩陣乘法

```
#矩陣乘法
import numpy as np
a=np.array([[2,4,6],[8,10,12]])
b=np.arange(6).reshape((2,3))

print(a)
print(b.T) #transpose

print(np.dot(a,b.T))
```

			0	3
			1	4
			2	5
2	4	6		
8	10	12		

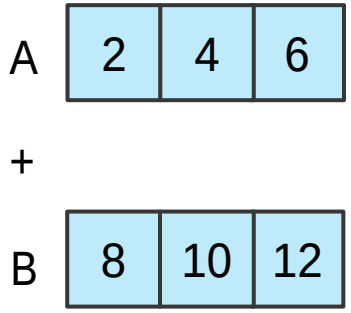
```
[[ 2  4  6]
 [ 8 10 12]]
[[0 3]
 [1 4]
 [2 5]]
[[ 16  52]
 [ 34 124]]
```

NumPy - array 合併

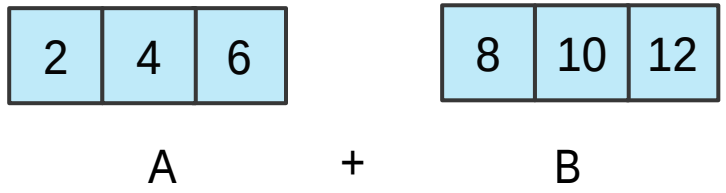
```
import numpy as np
A = np.array([1,1,1])
B = np.array([2,2,2])

print(np.vstack((A,B))) #vertical
print(np.hstack((A,B))) #horizontal
```

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

vstack(A,B)



hstack(A,B)



NumPy - array分割(1)

```
import numpy as np
a = np.arange(12).reshape((3,4))

print(a)
print(np.vsplit(a,3))
print(np.hsplit(a,2))
```

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

NumPy - array分割(2)

a

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

vsplit(a,3)

a

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

hsplit(a,2)

a

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

帶著 Python 踏上資料科學之路！

科學運算模組基礎

numpy & pandas

在python在處理資料科學相關運用的時候，

有兩個兩個重要的模塊，一個是 numpy,一個是 pandas。

科學運算

Numpy 介紹與應用

Pandas 介紹與應用

單元目的

- 了解科學運算中 numpy & pandas 的重要性
- 了解 numpy 的語法基礎
- 培養執行 numpy 的能力
- 了解 pandas 中 series 與 dataframe 語法基礎
- 培養執行 numpy 的能力

Pandas

Pandas建構於Numpy上，Python做資料分析常用的套件

支援

- Numpy的矩陣運算功能
- SQL-like function
- 資料預處理
- 視覺化

Pandas 基本介紹

- **Numpy 和 Pandas 有什麼不同?**
  - Numpy 是 array 形式的，沒有欄位名稱或標籤。
  - Pandas 基於 Numpy 構建的，有列表的標籤。
    - 主要兩個數據結構：Series 和 DataFrame。

Series

Series是有index的一維array

```
import pandas as pd
import numpy as np
s = pd.Series([1,"abc","6",np.nan,44,1])

print(s)
```

```
0      1
1    abc
2      6
3    NaN
4     44
5      1
dtype: object
```

想一想：如果是numpy的array呢?

DataFrame - 建立資料集(矩陣)

DataFrame 用來處理結構化(Table like)的資料，有列索引與欄標籤的二維資料集，可以透過 Dictionary 或是 Array 來建立，但也可以利用外部的資料來讀取後來建立，例如： CSV 檔案、資料庫等等。

建立一個DataFrame的資料集，資料請亂數(random)出7*3筆

```
import pandas as pd
import numpy as np

#生成方式一：用array的矩陣(1)
df = pd.DataFrame(np.random.randn(7,3))
print(df)
```

```
          0         1         2
0  1.238052 -1.034129  0.222476
1 -0.457043 -1.252934  0.955214
2 -0.476281 -1.509945  0.573486
3  1.046944  0.143649 -0.138385
4 -1.268997  0.033301 -0.989308
5 -0.572510 -0.950910 -0.086820
6  0.250895 -0.889284 -0.149622
```

DataFrame - 建立資料集(矩陣)

建立一個DataFrame資料集，並且加上列索引與欄標籤

```
import pandas as pd
import numpy as np

#生成方式一：用array的矩陣(2)
eat = np.random.randint(10,size=(7,3))*5+50
dates = pd.date_range('20170812',periods=7)
df0 = pd.DataFrame(eat)

#加上欄位
df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐'])
print(df1)
```

```
            早餐  午餐  晚餐
2017-08-12  65  75  65
2017-08-13  65  55  70
2017-08-14  70  55  80
2017-08-15  70  85  95
2017-08-16  65  85  60
2017-08-17  60  80  60
2017-08-18  95  95  50
```

DataFrame - 選擇數據

取得DataFrame資料集的內容

df1['欄位名稱'] 取得該欄位底下的值
df1[起始索引:結束索引] 取得起始索引到結束索引的資料

```
print(df1['午餐'])
print(df1[0:3])
```

```
2017-08-12    75
2017-08-13    55
2017-08-14    55
2017-08-15    85
2017-08-16    85
2017-08-17    80
2017-08-18    95
Freq: D, Name: 午餐, dtype: int32

            早餐   午餐   晚餐
2017-08-12  65    75    65
2017-08-13  65    55    70
2017-08-14  70    55    80
```

```
 早餐   午餐   晚餐
2017-08-12  65   75   65
2017-08-13  65   55   70
2017-08-14  70   55   80
2017-08-15  70   85   95
2017-08-16  65   85   60
2017-08-17  60   80   60
2017-08-18  95   95   50
```

原始資料集

DataFrame - 選擇數據 loc、iloc、ix

```
# select by label
print(df1.loc['20170812'])
print(df1.loc[:,['早餐','晚餐']])

#select by position
print(df1.iloc[3,1])
print(df1.iloc[3:5,1:3])

#select by ix
print(df1.ix[:3,['午餐','晚餐']])

# 判斷
print(df1[df1.午餐>80])
```

	早餐	午餐	晚餐
2017-08-12	65	75	65
2017-08-13	65	55	70
2017-08-14	70	55	80
2017-08-15	70	85	95
2017-08-16	65	85	60
2017-08-17	60	80	60
2017-08-18	95	95	50

原始資料集

```
早餐    65
午餐    75
晚餐    65
Name: 2017-08-12 00:00:00, dtype: int32
            早餐  晚餐
2017-08-12  65  65
2017-08-13  65  70
2017-08-14  70  80
2017-08-15  70  95
2017-08-16  65  60
2017-08-17  60  60
2017-08-18  95  50
85
            午餐  晚餐
2017-08-15  85  95
2017-08-16  85  60
            午餐  晚餐
2017-08-12  75  65
2017-08-13  55  70
2017-08-14  55  80
 早餐  午餐  晚餐
2017-08-15  70  85  95
2017-08-16  65  85  60
2017-08-18  95  95  50
```

DataFrame - 建立資料集(字典)

```
import pandas as pd
import numpy as np

#生成方式二，字典方式
df2 = pd.DataFrame({'小數':pd.Series(1,index=list(range(4)),dtype='float32'),
                    '整數':np.array([3] * 4,dtype='int32'),
                    '時間':pd.Timestamp('20170812'),
                    '類別資料':pd.Categorical(["test","train","test","train"]),
                   })

#dtype指定資料格式
```

想一想：字典的key代表甚麼?

DataFrame - 屬性 dtypes、index、describe

dtypes 查看資料型態
index 查看資料集的索引
describe() 查看數字資料描述

```
#DataFrame的屬性
print(df2)
print(df2.dtypes)
print(df2.index) #print(df2.columns), print(df2.values)
print(df2.describe()) #有甚麼要注意的呢?
```

```
   小數  整數          時間   類別資料
0  1.0   3 2017-08-12   test
1  1.0   3 2017-08-12  train
2  1.0   3 2017-08-12   test
3  1.0   3 2017-08-12  train
小數              float32
整數                int32
時間       datetime64[ns]
類別資料           category
dtype: object
Int64Index([0, 1, 2, 3], dtype='int64')
         小數   整數
count  4.0  4.0
mean   1.0  3.0
std    0.0  0.0
min    1.0  3.0
25%    1.0  3.0
50%    1.0  3.0
75%    1.0  3.0
max    1.0  3.0
```

DataFrame - 轉置、排序資料

```
print(df2.T)
print(df2.sort_index(axis=1, ascending=False))
print(df2.sort_values(by='類別資料'))
```

```
0          ...                                3
小數                       1          ...                              1
整數                       3          ...                              3
時間     2017-08-12 00:00:00          ...            2017-08-12 00:00:00
類別資料                  test           ...                          train

[4 rows x 4 columns]
    類別資料         時間  整數   小數
0   test 2017-08-12   3  1.0
1  train 2017-08-12   3  1.0
2   test 2017-08-12   3  1.0
3  train 2017-08-12   3  1.0
    小數  整數         時間   類別資料
0  1.0   3 2017-08-12   test
2  1.0   3 2017-08-12   test
1  1.0   3 2017-08-12  train
3  1.0   3 2017-08-12  train
```

Pandas - 輸入值

利用loc、iloc、判斷來修改 dataframe 資料集的值

```
import numpy as np
import pamdas as pd

eat = np.random.randint(10,size=(7,3))*5+50
dates = pd.date_range('20170812',periods=7)
df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐'])

df1.iloc[2,2] = 95
df1.loc['20170818','晚餐'] = 60
df1.晚餐[df1.早餐>80] = 40
df1.loc['20170814','午餐'] = np.nan

print(df1)
```

```
            早餐    午餐  晚餐
2017-08-12  95  65.0  40
2017-08-13  70  60.0  95
2017-08-14  55   NaN  95
2017-08-15  85  75.0  40
2017-08-16  75  80.0  75
2017-08-17  80  85.0  75
2017-08-18  70  90.0  60
```

Pandas - 檔案讀取、儲存

讀取一個csv檔案並將讀出的結果儲存成新的csv檔

```
import pandas as pd #加載模組

#讀取csv
data = pd.read_csv('XXX.csv')

#列印data
print(data)

#存成csv
data.to_csv('student.csv')

#說明 http://pandas.pydata.org/pandas-docs/stable/io.html
```

Pandas - 合併 dataframe (concat)

合併多個dataframe

```
#concat
import pandas as pd
import numpy as np

#定義資料集
df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'])
df2 = pd.DataFrame(np.ones((3,4))*1, columns=['a','b','c','d'])
df3 = pd.DataFrame(np.ones((3,4))*2, columns=['a','b','c','d'])

#concat縱向合併
res = pd.concat([df1, df2, df3], axis=0, ignore_index=True)
print(res)
```

	a	b	c	d
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	2.0	2.0	2.0	2.0
7	2.0	2.0	2.0	2.0
8	2.0	2.0	2.0	2.0

Pandas - 合併 dataframe (join、append)

```
#concat

#join, ['inner','outer']
df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'], index=[1,2,3])
df2 = pd.DataFrame(np.ones((3,4))*1, columns=['b','c','d','e'], index=[2,3,4])
res = pd.concat([df1, df2], axis=0, join='outer', ignore_index=True )

#append
#res = df1.append(df2, ignore_index=True)
```

想一想：用 append() 要注意甚麼呢??

	a	b	c	d	e
0	0.0	0.0	0.0	0.0	NaN
1	0.0	0.0	0.0	0.0	NaN
2	0.0	0.0	0.0	0.0	NaN
3	NaN	1.0	1.0	1.0	1.0
4	NaN	1.0	1.0	1.0	1.0
5	NaN	1.0	1.0	1.0	1.0

Pandas - 合併 dataframe (merge)

```
# merge
import pandas as pd

left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key': ['K1', 'K2', 'K3', 'K4'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
res = pd.merge(left, right, on='key')
```

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

	key	C	D
0	K1	C0	D0
1	K2	C1	D1
2	K3	C2	D2
3	K4	C3	D3

merge

	key	A	B	C	D
0	K1	A1	B1	C0	D0
1	K2	A2	B2	C1	D1
2	K3	A3	B3	C2	D2

Pandas - 合併 dataframe(merge)

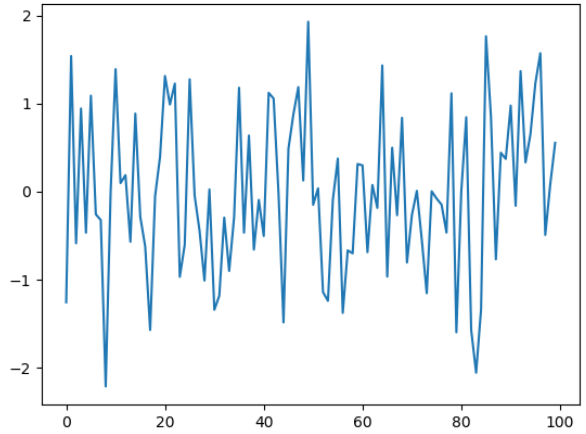
```
# merge
import pandas as pd

#定義資料集並列印出
left = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                     'key2': ['K0', 'K0', 'K0', 'K0'],
                     'C': ['C0', 'C1', 'C2', 'C3'],
                     'D': ['D0', 'D1', 'D2', 'D3']})

#依key1與key2 進行合併，並印出四種結果['left', 'right', 'outer', 'inner']
res = pd.merge(left, right, on=['key1', 'key2'], how='inner')
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

pandas & matplotlib.pyplot

將數據視覺化

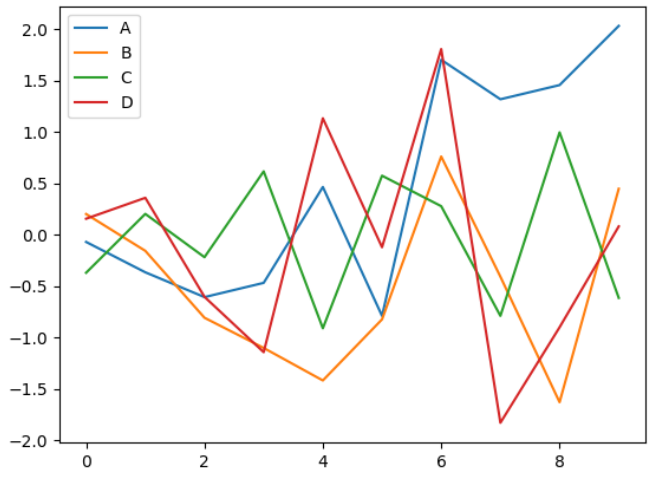
```
#Series
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#隨機生成100個數據
data = pd.Series(np.random.randn(100),index=np.arange(100))


#pandas 數據可以直接觀看其可視化形式
data.plot()
plt.show()
```



pandas & matplotlib.pyplot

將數據視覺化

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Dataframe
data = pd.DataFrame(
    np.random.randn(10,4),
    index=np.arange(10),
    columns=list("ABCD")
    )
# bar, hist, box, scatter, pie,…..
data.plot()
#ax = data.plot.scatter(x='A', y='B', label='class1',
color='lightgreen')
#data.plot.scatter(x='C', y='D', label='class1', color='darkred', ax=ax)
plt.show()
```



帶著 Python 踏上資料科學之路！

資料預處理實作

檔案匯入匯出

資料格式整理、遺漏值處理、重複值處理

資料表合併、聯集

推動大學程式設計教學計畫。分項六：資料分析領域教學研發推廣團隊（國立中央大學網路學習科技研究所洪暉鈞老師主編）

資料預處理流程

- 檔案讀取、資料匯入
- 處理遺漏值
- 移除重複值
- 資料對應\取代\分箱
- 資料表合併

單元目的

- 具備使用python將檔案讀取與存檔的能力
- 遇到遺漏值可活用進行條件式刪除、插補
- 遇到重複值可依照條件進行刪除
- 針對資料格式及不同格式欄位進行對應\取代\分箱操作
- 不同資料來源或資料表的合併

1.檔案匯入匯出

兩種方式進行檔案匯入匯入

檔案匯入匯出

- 兩種方式進行檔案匯入匯入

使用原本Python的open()方法

使用pandas的套件直接讀成Dataframe

檔案匯入匯出

- 檔案開啟
- Python使用 open() 打開檔案
  - f = open('檔案', '模式')
- 模式
  - r　　讀取(檔案需存在)
  - w　新建檔案寫入(檔案可不存在，若存在則清空)
  - a　　資料附加到舊檔案後面(游標指在EOF)

檔案匯入匯出

- 檔案讀取
- Python使用 `read()` 讀取
  - `f.read()`
- 其他模式
  - `f.read(size)`
    - `size` 可以指定要讀進來的字串長度
  - `f.readline()`
    - 讀取一行，加上一個換行符號 \n
  - `f.readlines()`
    - 讀取成一個 list，每一行為一項
    - 讀取一行，加上一個換行符號 \n

檔案匯入匯出

- 檔案寫入
- Python使用 open() 打開，並且使用 write() 寫入
  - f = open('檔案', 'a')
  - f.write("要添加的內容....")
  - f.close()

檔案匯入匯出

- 使用 pandas 的套件直接讀成 dataframe
  - `import pandas as pd`
  - `df = pd.read_csv("開啟的檔名.csv")`
  - `df.to_csv("要存檔的檔名.csv")`
    - 中文要注意轉碼問題，有可能要編碼，如
    - `df = pd.read_csv("檔名.csv", encoding='big5')`

檔案匯入匯出

```
import pandas as pd

df1 = pd.read_csv("testc.csv",encoding='big5')
df1[:5]
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

檔案匯入匯出

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

http://pandas.pydata.org/pandas-docs/stable/io.html

2.處理遺漏值

討論：遇到遺漏值怎麼辦?

處理遺漏值

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

- 先判斷是否有缺失數據 NaN
- 以及遺漏值的分布情形

```
#判斷是否有缺失數據 NaN, 為 True 表示缺失數據:

df1.isnull()
```

	ID	dept	gender	test01	test02	test03
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	True	False	False
4	False	False	False	False	False	False

處理遺漏值

- 先判斷是否有缺失數據 NaN
- 以及遺漏值的分布情形

```
df1.isnull().sum()
```

	ID	dept	gender	test01	test02	test03
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	True	False	False
4	False	False	False	False	False	False

```
ID          0
dept        0
gender      0
test01      1
test02      1
test03      0
dtype: int64
```

處理遺漏值

- 處理方法一：

```
df2 = df1.fillna(value=0)
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	0.0	85
3	4	牙醫學系(106)	m	0.0	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

處理遺漏值

- 處理方法二：
  - 將 NaN 值刪除

```
df3 = df1.dropna()
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

ID=3,4被刪除了

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	5	牙體技術學系(104)	f	95.0	85.0	80

處理遺漏值

- 處理方法二-參數：

```
#其他用法
df1.dropna(subset=['欄位名稱']) #針對特定欄位名稱
df1.dropna(axis=0) # 0: 對資料列進行操作; 1: 對該欄位進行操作
df1.dropna(how='any') # 'any': 存在 NaN 就 drop 掉;
                      # 'all': 全是 NaN 才 drop
df1.dropna(thresh=3) #至少要有 thresh 個非NaN值
```

處理遺漏值

- 處理方法二-參數範例：

```
#其他用法
df3 = df1.dropna(subset=["test02"],axis=0,how='any')
#針對test02列存在NaN就drop掉
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

ID=3被刪除了

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	4	牙醫學系(106)	m	NaN	95.0	90
3	5	牙體技術學系(104)	f	95.0	85.0	80

處理遺漏值

- 處理方法三：
  - 差補法 (使用sklearn)

```
from sklearn.preprocessing import Imputer
imr = Imputer(missing_values='NaN', strategy=_______, axis=0)
imr = imr.fit(df1)
imputed_data = imr.transform(df1.values)
imputed_data

# strategy='mean', 'median', 'most_frequent'
```

處理遺漏值

- 處理方法三：
  - 差補法 (使用sklearn)

```
from sklearn.preprocessing import Imputer
imr = Imputer(missing_values='NaN', strategy=mean, axis=0)
imr = imr.fit(df1)
imputed_data = imr.transform(df1.values)
imputed_data
```

	ID	test01	test02	test03
0	1	95.0	85.0	95
1	2	95.0	95.0	95
2	3	95.0	NaN	85
3	4	NaN	95.0	90
4	5	95.0	85.0	80

	ID	test01	test02	test03
0	1	95.0	85.0	95
1	2	95.0	95.0	95
2	3	95.0	90.0	85
3	4	95.0	95.0	90
4	5	95.0	85.0	80

3. 移除重複值

重複輸入，或是有一模一樣的值

移除重複值

- 使用 pandas
- 檢查是否有重複
  - `df.duplicated()`
- 移除重複
  - `df.drop_duplicates()`
    - 針對特定欄位：可加上欄位名稱與保留順序
    - 例如：`df.drop_duplicates(['col1'], keep='last')`

移除重複值

```
import pandas as pd
df = pd.DataFrame({'col1':['A','A','A','A','B','B','C','C'],
                   'col2':[1,2,2,3,3,4,5,5]})
df_d = df.drop_duplicates(['col1'], keep='last')
#指定欄位col1重複就移除，保留最後一個
```

	col1	col2
0	A	1
1	A	2
2	A	2
3	A	3
4	B	3
5	B	4
6	C	5
7	C	5

	col1	col2
3	A	3
5	B	4
7	C	5

帶著 Python 踏上資料科學之路！

資料預處理實作

檔案匯入匯出

資料格式整理、遺漏值處理、重複值處理

資料表合併、聯集

推動大學程式設計教學計畫。分項六：資料分析領域教學研發推廣團隊（國立中央大學網路學習科技研究所洪暉鈞老師主編）

資料預處理流程

- 檔案讀取、資料匯入
- 處理遺漏值
- 移除重複值
- 資料對應\取代\分箱
- 資料表合併

單元目的

- 具備使用python將檔案讀取與存檔的能力
- 遇到遺漏值可活用進行條件式刪除、插補
- 遇到重複值可依照條件進行刪除
- 針對資料格式及不同格式欄位進行對應\取代\分箱操作
- 不同資料來源或資料表的合併

4. 資料對應\取代\分箱

想把資料分組，或是特定去取代某一個值

資料對應

- 新建一個字典
  - {key01 : value01,
    key02 : value02,…..}
  - 用字典裡的value取代原本的key(原本資料表的值)
- 在特定欄位用 map() 執行
  - df["欄位"].map({dict})
- 例如：將性別欄位 (女、男) 變成 (0、1)
  - `gender_to_boolean = {"female":0,"male":1} #字典`
  - `df['gender'].map(sex_to_boolean)`

資料對應

```
df = pd.DataFrame({'gender':['male','male','female','male','female','female']})

gender_to_boolean = {'female':0,'male':1} #dict
df['code'] = df['gender'].map(gender_to_boolean)
```

	gender
0	male
1	male
2	female
3	male
4	female
5	female

	gender	code
0	male	1
1	male	1
2	female	0
3	male	1
4	female	0
5	female	0

取代

- 取代請愛用 `replace()`
- df.replace(原本的值，新的值）
  - `df['col2'].replace("-",0)`
- df.replace(也可以用字典方式）
  - `df.replace({"NULL":0 , "-":-1})`

取代

```
df = pd.DataFrame({'col1':['c01','c02','c03','c04','c05'],
                   'col2':[65,'NULL','-','-',78],
                   'col3':[321,34,'NULL','-',34]})
df['col2'] = df['col2'].replace('-',0)
```

	col1	col2	col3
0	c01	65	321
1	c02	NULL	34
2	c03	-	NULL
3	c04	-	-
4	c05	78	34

	col1	col2	col3
0	c01	65	321
1	c02	NULL	34
2	c03	0	NULL
3	c04	0	-
4	c05	78	34

分箱

- 建立兩個list
  - bins
    - 分箱的間隔點 list
  - labels
    - 各區間對應的 labels

  - `bins = [0, 60, 70, 80, 90, 100]`
  - `labels = ['E','D','C','B','A']`
  - `pd.cut(df['score'], bins, right=False, labels=labels)`

分箱

```
df = pd.DataFrame({'id':['John','Mary','Tom','Nick','Alice'],
                   'score':[90,59,68,77,60]})
bins = [0, 60, 70, 80, 90, 100]
labels = ['E','D','C','B','A']

df['label'] = pd.cut(df['score'],bins, right=False, labels=labels)
```

	id	score
0	John	90
1	Mary	59
2	Tom	68
3	Nick	77
4	Alice	60

	id	score	label
0	John	90	A
1	Mary	59	E
2	Tom	68	D
3	Nick	77	C
4	Alice	60	D

5.資料表合併

不同表格的合併與聯集

Pandas 合併dataframe

```
#concat
import pandas as pd
import numpy as np

#定義資料集
df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'])
df2 = pd.DataFrame(np.ones((3,4))*1, columns=['a','b','c','d'])
df3 = pd.DataFrame(np.ones((3,4))*2, columns=['a','b','c','d'])

#concat縱向合併
res = pd.concat([df1, df2, df3], axis=0, ignore_index=True)
```

	a	b	c	d
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	2.0	2.0	2.0	2.0
7	2.0	2.0	2.0	2.0
8	2.0	2.0	2.0	2.0

Pandas 合併dataframe

```
# merge
import pandas as pd

left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key': ['K1', 'K2', 'K3', 'K4'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
res = pd.merge(left, right, on='key')
```

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

	key	C	D
0	K1	C0	D0
1	K2	C1	D1
2	K3	C2	D2
3	K4	C3	D3

merge →

	key	A	B	C	D
0	K1	A1	B1	C0	D0
1	K2	A2	B2	C1	D1
2	K3	A3	B3	C2	D2

Pandas 合併dataframe

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

```
# merge
import pandas as pd

#定義資料集並打印出
left = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                     'key2': ['K0', 'K0', 'K0', 'K0'],
                     'C': ['C0', 'C1', 'C2', 'C3'],
                     'D': ['D0', 'D1', 'D2', 'D3']})

#依key1與key2 進行合併，並印出四種結果['left', 'right', 'outer', 'inner']
res = pd.merge(left, right, on=['key1', 'key2'], how='inner')
```