

Data Structure in Python

## Tuple

- 不可更動內容、功能少、占空間小
- 創建:使用小括號()
  - e.g. x = (10, 12)
- ·轉換成list
  - e.g. y = list(x)

### 串列 (list) - 創建

- 創建空串列
  - sample\_list = []
  - sample\_list = list()
- 初始項目
  - ▶ sample\_list = [10, 'test', 3.14] (以逗號隔開,資料型態不拘)
  - ▶ sample\_list = [['A', 'B', 'C'], [1, 2, 3]] (串列中也可以放串列等資料結構)

### 串列 (list) - 新增

- · 在串列最後增加項目:list.append(item)
  - e.g. sample\_list.append('A')
- · 在串列最後增加串列:list.extend(list)
  - e.g. sample\_list.extend(['D', 'E', 'F'])
- · 指定位置插入項目:list.insert(index, item)
  - e.g. sample\_list.insert(1, 'A')

### 串列 (list) - 删除

- · 删除整個串列:del
  - e.g. del sample\_list
- · 删除指定位置項目:del
  - e.g. del sample\_list[1]
- 移出 (回傳) 並刪除串列項目: pop(index)
  - ▶ 預設pop()是移出最後一項,相同於pop(-1),若pop(o)則是移出第一項
  - e.g. sample\_list.pop()

### 串列 (list) - 項目位置與排序

- · 找出第一次出現的位置:list.index(value)
  - ▶ e.g. sample\_list.index('A') (回傳位置index)
- · 判斷值是否在串列中:'value' in list
  - ▶ e.g. if('A' in sample\_list) (回傳True/False)
- · 根據值的內容來排序(sorting):sorted(list)
  - e.g. sorted(['D','A','F','C','E','D']) => ['A', 'C', 'D', 'D', 'E', 'F']
  - e.g. sorted([4,23,1,3,2,98,3]) => [1, 2, 3, 3, 4, 23, 98]

# 字典 (Dictionary) - 創建

- · 以key-value (鍵-值) pair形式儲存的資料結構
- · key必須獨一無二 (unique) ,若有重複key會自動覆蓋原key-value pair
- 創建:使用大括號 {},內容可存多種資料型態和結構
  - e.g. dic = {'Jack':84,'Ben':63,'Cathy':'Hey', 'Bob': 83.3}
  - e.g. dic = {'Jack':[84,23,34],'Ben':[63,12,74],'Cathy':(12,43,76),'Bob': {83,81,90}}
- 取值:dictionary[key]
  - e.g. dic['Jack']

# 字典 (Dictionary) - 新增與刪除

- · 新增字典內容:update(dictionary)
  - e.g. dic.update({'William':[23,43,84],'Eric':[93,31,32]})
- 删除項目:del
  - e.g. del dic['William']
  - ▶ e.g. del dic (刪除整個字典)

# 字典 (Dictionary) - 取值

- · 判斷項目 (key) 是否存在於字典: in
  - e.g. 'Eric' in dic
- · 獲取字典中的資料(皆非list,可用list()轉換)
  - 取得所有的key:dictionary.keys()
  - 取得所有的value: dictionary.values()
  - 取得所有的key-value pair: dictionary.items()

## 集合 (Set) - 創建

- 內容不可重複,若有重複創建會自動刪除
- 創建:使用大括號 {}
  - e.g.  $set1 = \{1,2,3,4,5\}$
- 項目數:len(set)
- 最大值:max(set)
- 最小值:min(set)

## 集合 (Set) - 新增與刪除

- · 新增:add(item)
  - e.g. set1.add(6)
- 删除:remove(item)
  - e.g. set1.remove(6)

## 集合 (Set) - 運算

- · 交集:set1 & set2 或 set1.intersection(set2)
- ・ 聯集:set1 | set2 或 set1.union(set2)
- · 差集:set1 set2 或 set1.difference(set2)
- set1是set2的真子集:set1 < set2
- set1是set2的子集:set1 <= set2 或 set1.issubset(set2)
- set1是set2的超集合(superset):set1 >= set2
- · 判斷是否"無"交集:set1.isdisjoint(set2)

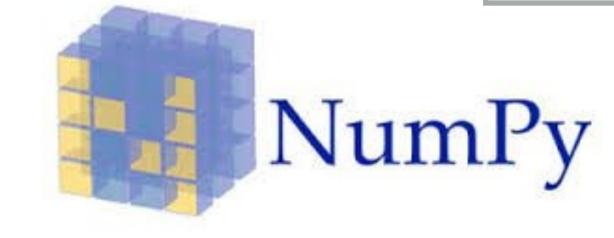


# ndarray

- · Numpy:處理一維或多維矩陣(array-like)存取和運算
  - e.g. import numpy as np
  - ▶ s = np.array([3,4,5,6]) #一維矩陣
  - s = np.array([[1,2,4,8],[1,3,5,7],[3,6,9,12]]) #多維矩陣 (3\*4)
- 矩陣運算
  - ▶ 線性代數(矩陣乘法、轉置、eigenvalues...等)
  - ▶ 數學運算(加減乘除、約數、平方、平方根、取log、絕對值...等)
  - 統計運算(最大值、最小值、平均、中位數、標準差、相關性...等)
  - ▶ Numpy提供的功能:<a href="https://docs.scipy.org/doc/numpy/reference/routines.html">https://docs.scipy.org/doc/numpy/reference/routines.html</a>

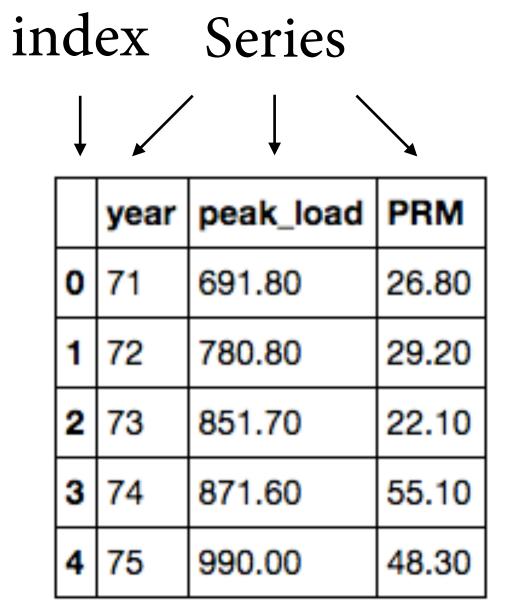
Notes

▶ as:使用別名



#### Series

- · Series(序列):有index的一維的ndarray
- DataFrame的columns即是Series
  - pd.Series(list)
  - pd.Series(ndarray)



DataFrame

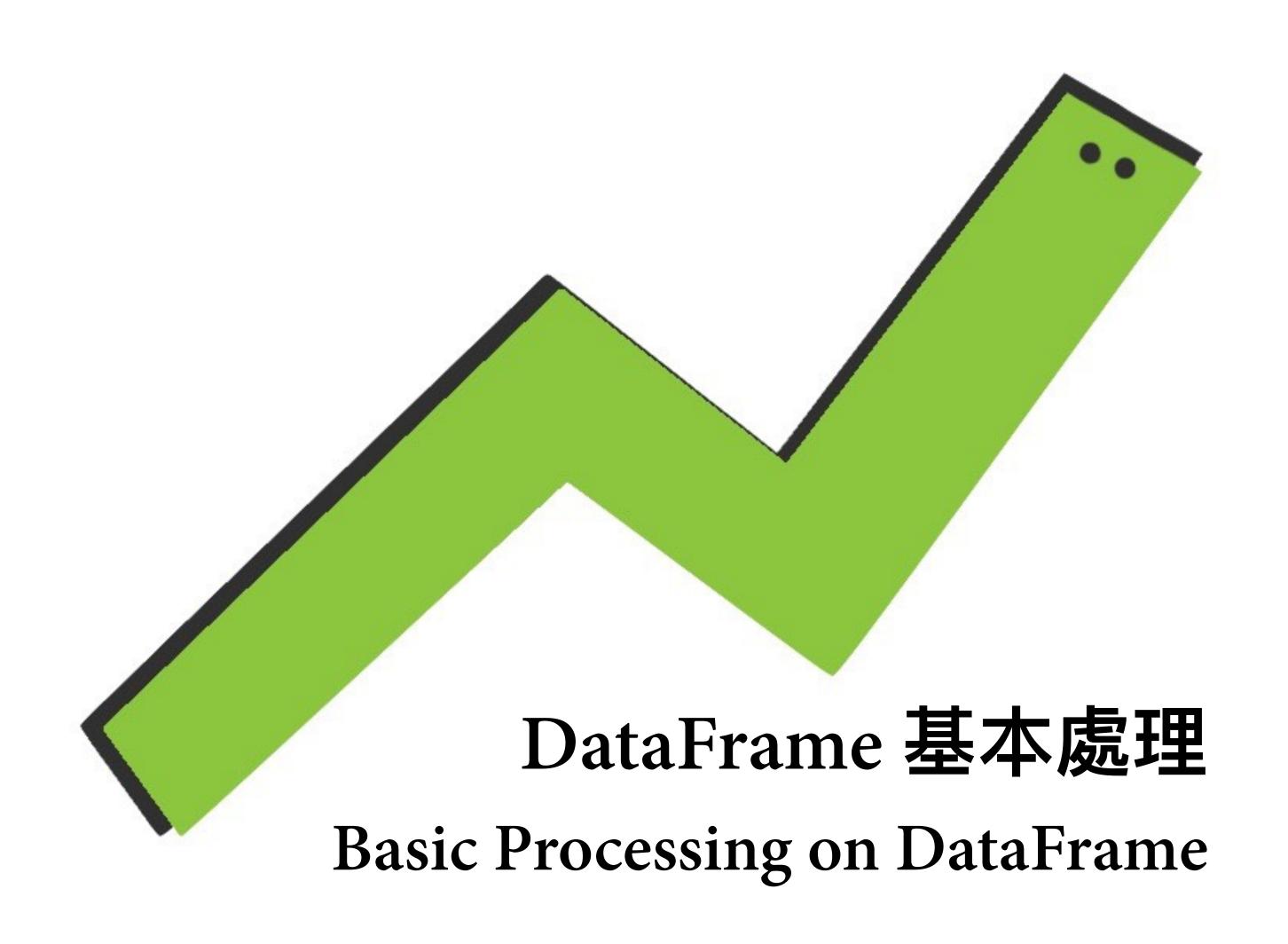
Document: <a href="http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html">http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html</a>

#### DataFrame

- · Pandas建構於Numpy上,是Python做資料分析最常用的套件
- Pandas提供DataFrame資料結構(spreadsheet-like)
- 提供功能:
  - ▶ Numpy既有的矩陣運算功能
  - ▶ SQL-like function(查表、排序、聚合、JOIN、樞紐...等)
  - 預處理(移除重複值、移除遺失值、資料取代、對應轉換...等)
  - ▶ 視覺化(直方圖、長條圖、折線圖、散佈圖、箱形圖…等)



	year	peak_load	PRM
0	71	691.80	26.80
1	72	780.80	29.20
2	73	851.70	22.10
3	74	871.60	55.10
4	75	990.00	48.30



#### 創建

- pd.DataFrame({'column\_name:[value list]',...}, index=[list])
  - e.g. df = pd.DataFrame({'cid':['co1','co2','co3','co4','co5'],'time\_spent':[23,41,342,97,456]})
- $pd.DataFrame([\{`column\_name: value',...\},\{...\},...],index=[list])$ 
  - e.g. df = pd.DataFrame([{'cid':'co2','num\_products':2},{'cid':'co3','num\_products':3}, {'cid':'co4','num\_products':6}]) # JSON list

#### 耳值

- 單欄: (Series or DataFrame)
  - e.g. df[['time\_spent']] (DataFrame Type)
  - e.g. df['time\_spent'] (Series Type)
  - e.g. df.time\_spent (Series Type)
- 多欄: (DataFrame)
  - eg. df[['cid','time\_spent']] (DataFrame Type)
- 列: (DataFrame)
  - 根據index: e.g. df[o:2], df['b':'d']
  - ▶ 根據指定條件: e.g. df[df['time\_spent']>100]

#### Notes

▶也可以使用.loc取值,請參考

P23~P24說明

#### 新增

- 欄
  - e.g. df['num\_products'] = [2,3,1,6,7]
- 列:append(DataFrame)
  - e.g. df = df.append(pd.DataFrame([{'cid':'co6','time\_spent':231,'num\_products': 3}])).reset\_index(drop=True)
  - 若要新增的欄位不存在,會自動新增

### 修改

- 欄位名稱: rename(columns={old\_column:new\_column,...})
  - e.g. df = df.rename(columns = {'cid':'pid','num\_products':'num\_purchase'})
- · 列: (DataFrame要創建成相同的index才能成功修改列內容)
  - e.g. df[0:1] = pd.DataFrame([{'pid':'co7','time\_spent':31,'num\_purchase':1}]) #default index = o
  - e.g. df[2:3] = pd.DataFrame([{'pid':'co3', 'num\_sell':5, 'test':23}],index=[2])

#### Notes

▶也可以使用.loc修改資料,請參 考P23~P24說明

### 删除

#### • 欄

- del : e.g. del df['pid']
- ▶ drop: e.g. df = df.drop('time\_spent',axis = 1) (axis = 1 表示要刪除的是欄不是列)

#### 列

- drop : df = df.drop([1,2])
- reset\_index : df = df.reset\_index(drop=True)
- ▶ drop=True會刪除舊的index重新排序index

#### drop=False(default)

	index	cid	time_spent
0	0	c01	23
1	3	c04	97
2	4	c05	456

▶ drop=False(default)則保留舊的index,再新增一欄重新排序的新index

## 補充:loc用法

• df.loc[row\_indexer,column\_indexer] 可取指定row, column位置的資料

```
In [1]: import pandas as pd
In [2]: df = pd.DataFrame({'A':[1,2,3,4],'B':[0,2,4,6],'C':[1,3,5,7]})
Out[2]:
        df.loc[1,'B'] = 10 #修改指定位置資料
        df
Out[3]:
```

# 補充:loc用法(cont.)

#### • 修改資料

```
In [4]: df.loc[:,'A'] = [2,4,6,8] #修改指定欄資料
Out[4]:
        1 4 10 3
In [5]: df.loc[2,:] = [1,3,5] #修改指定列資料
Out[5]:
            BC
       1 4 10 3
```