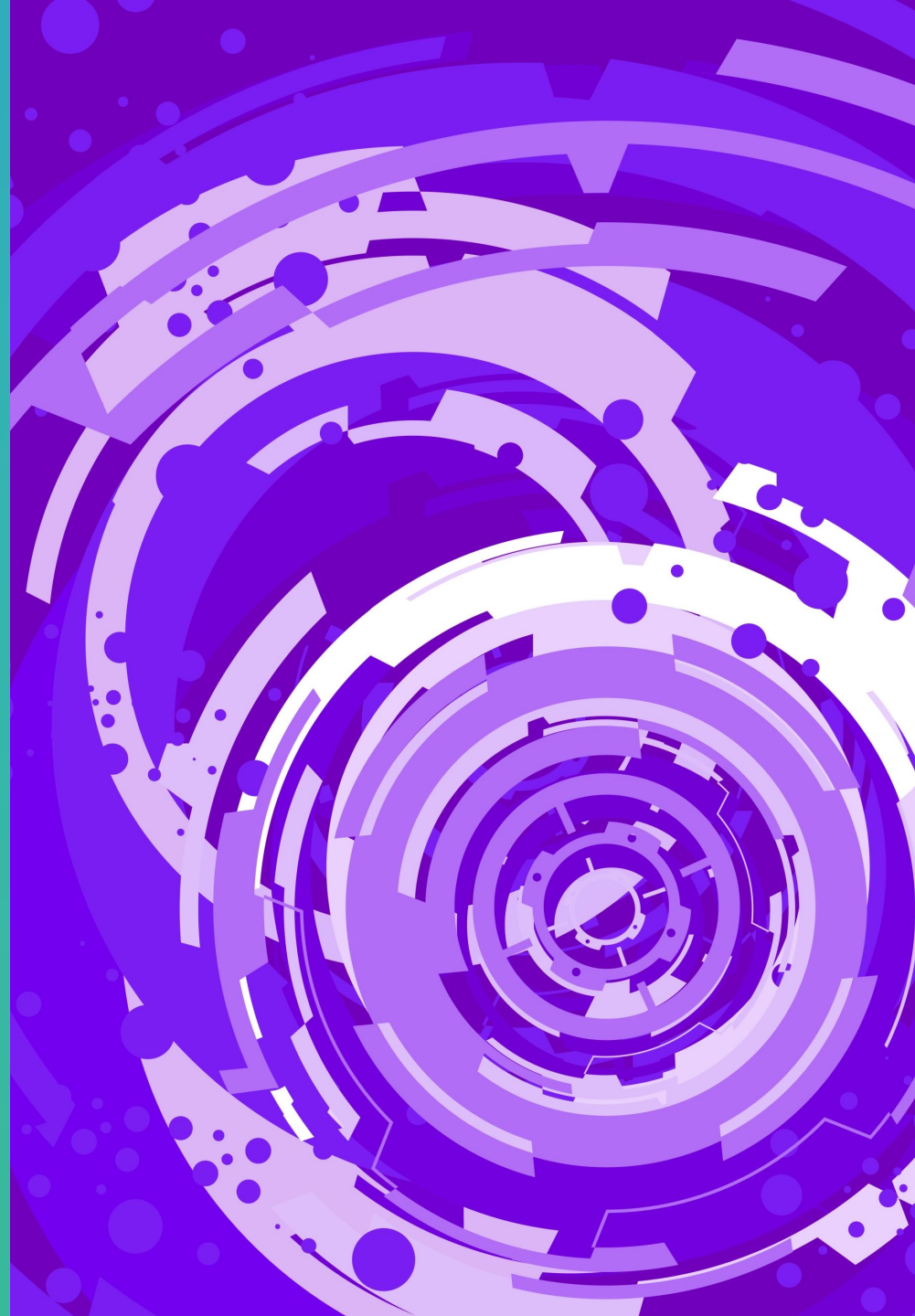
URBAN GEOGRAPHIC INFORMATION SYSTEM

Python Basic III – Numpy & Pandas

Chun-Hsiang Chan

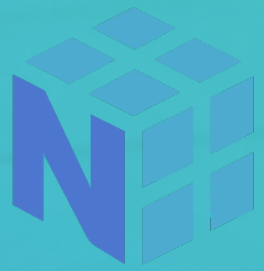
Department of Geography,

National Taiwan Normal University

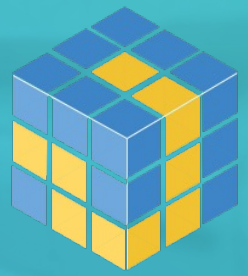


Outline

- Numpy
- Pandas



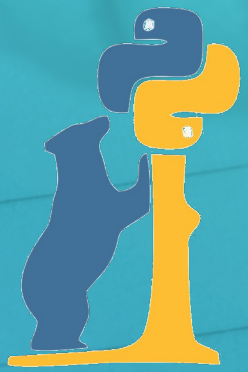
NumPy



NumPy

pandas

Pandas



NumPy

- NumPy is a powerful package for data processing because one of the most important function in NumPy – ndarray is developed along with the nature of CPU architecture, which leverages continuous memory for storing array (unlike list).
- Why NumPy is always faster than Python list? Because only a few is written in Python and most of parts (need powerful computation and fast calculation) are written in C or C++.
- Before we start, here, let's tell you how to call a Python library.

```
# import python packages
import numpy as np # simplify numpy into np
a = np.ones((3,2))
```

NumPy

- The most powerful part in NumPy is array and its calculation; therefore, we will introduce several functions for NumPy's array. First of all, we introduce the simpliest way to declare a NumPy array.

```
# declare a numpy array
arr = np.array([0, 1, 2, 3, 4])
print(arr)
print(arr[3])
print(type(arr))
mylist = [0, 1, 2, 3, 4]
arr_ = np.array(mylist) # Is there any difference between arr and arr_?
```

NumPy

- Other approaches to declare a NumPy array.

```
# declare a numpy array
arr1 = np.array([4])
arr2 = np.array(4)
arr3 = np.array((4))
arr4 = np.array((4,4))
# what is the difference among arr1, arr2, arr3, and arr4?
# tell me your finding…
```

NumPy

- After we introduce the 0-dimensional and 1-dimensional array, it is time for 2-dimensional and 3-dimensional or even higher dimensional arrys.

```
# declare a numpy array
arr2D = np.array([[1, 2, 3],[4, 5, 6]])
arr3D = np.array([[[1, 2, 3],[4, 5, 6]], [[7, 8, 9],[10, 11, 12]]])
# indexing
arr2D[1, 2] = ?
arr3D[1, 0, 2] = ?
arr3D[-1, -2, -3] = ?
# what if we want to obtain 11 in the arr3D, then how can we index it?
```

NumPy

- Confirming the dimension of an array is crucial, especially in deep learning model because we leverage several arrays for calculation; however, their dimensions usually are different.

```
# confirm the array dimension
print(arr2D.ndim)
print(arr3D.ndim)
arr5 = np.array([1, 2, 3, 4], ndmin=5)
# print array
print(arr5)
print('number of dimensions :', arr5.ndim)
```

NumPy

- Advance indexing in NumPy.

```
# indexing
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
# try these codes
print(arr[:3])
print(arr[::2])
print(arr[0:8:2])
print(arr[::-1])
# 2-D array
arr_ = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr_[1:2, 0:2])
```

NumPy

- NumPy array supports several data types.

- `i` integer
- `b` Boolean
- `u` unsigned integer
- `f` float
- `c` complex float
- `m` timedelta
- `M` datetime
- `O` object
- `S` string
- `U` unicode string

```
# example 1
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print(arr.dtype)
# example 2
arrs = np.array(['apple','ball','cyan'])
print(arrs.dtype)
# example 3
arr1 = np.array([1, 2, 3, 4, 5, 6], dtype='f')
print(arr1.dtype)
# try this
arr1.astype('S')
```

NumPy

- When you need to duplicate an array, you need to remember that using copy function instead of directly assigning.
- Another issue is how get the dimension information in an array. Here, we introduce a new function – **shape**.

```
# copy an array
arr = np.array([1, 2, 3, 4])
b = arr.copy()
arr[2] = 100
# observe the array
print(arr)
print(b)
```

```
# print the shape of an array
print(arr.shape())
# print the specfic dimension
arr2 = np.array([[1,2],[3,4],[5,6]])
print(arr2.shape)
print(arr2.shape[1])
```

NumPy

- Sometimes, we want to modify the shape of an array.

```
# reshape
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
# 1-D to 2-D
arr_ = arr.reshape(2, 6)
print(arr_)
# 1-D to 3-D
arr_1 = arr.reshape(2, 3, 2)
print(arr_1)
# try this
arr_2 = arr_1.reshape(-1)
print(arr_2)
```

NumPy

- Here, we introduce how to join one array to another array.

```
# concat two arrays
arr1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
arr2 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
arr_0 = np.concatenate((arr1, arr2))
print(arr_0)
# concat two arrays
arr3 = np.array([[1, 2, 3, 4, 5, 6], [ 7, 8, 9, 10, 11, 12]])
arr4 = np.array([[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12]])
arr_1 = np.concatenate((arr3, arr4), axis=0)
arr_2 = np.concatenate((arr3, arr4), axis=1)
print(arr_1)
print(arr_2)
```

NumPy

- You may join one array to another, so you may split one array into several sub-arrays.

```
# split into three arrays
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
arr1 = np.array_split(arr, 3)
print(arr1)
arr2 = np.array([[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12]])
arr3 = np.array_split(arr2, 3, axis=1)
arr4 = np.array_split(arr2, 2, axis=0)
print(arr3)
print(arr4)
```

NumPy

- Sometimes, you want to the special value in your array.

```
# search in the array
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
x = np.where(arr == 3)
print(x) # I perfer to use x[0], and do you why?

# advance search
y = np.where(arr%2 == 0)
print(y)
```

NumPy

- Sorting is one of the most important function in the numerical analysis.

```
# sort 1-D array
arr1 = np.array(['x','t','a','h','z'])
arr2 = np.array([True, False, False, True])
print(np.sort(arr1))
print(np.sort(arr2))
# sort 2-D array
arr3 = np.array([[33, 5, 4],[42, 105, 78]])
print(np.sort(arr3))
```

NumPy

- Filter is very important in data cleaning and data preprocessing.

```
# design a filter in the array
arr = np.array([11, 24, 35, 24, 55, 62, 71, 84, 29, 10, 61, 42])
print(arr>50)
print(arr[arr>50])
# what is the difference between arr>50 and arr[arr>50]?
# tell me your idea…
```

NumPy

- Random variable generation

```
from numpy import random
x1 = random.randint(100)
print(x1)
x2 = random.randint(100, size=(2, 4))
print(x2)
y = random.rand(5, 2)
print(y)
z = random.choice([1,2,3,4], p=[0.5, 0.1, 0.2, 0.2], size=(2,5))
print(z)
# observe the differences among these codes
```

NumPy

- Imagine that you are playing cards, so you need to …

```
arr = np.array([1, 2, 3, 4, 5])
# using shuffle
random.shuffle(arr)
print(arr)
# using permutation
print(random.permutation(arr))
# observe the differences among these codes
```

NumPy

- Simple descriptive statistical analysis for an array.

```
arr = np.array([11, 24, 35, 24, 55, 62, 71, 84, 29, -10, -61, 42])
# get simple descriptive statistics
print(np.mean(arr)) # mean
print(np.var(arr)) # variance
print(np.std(arr, ddof=1)) # standard deviation
print(np.median(arr)) # median
print(np.absolute(arr)) # absolute
# create a range
print(np.arange(10))
print(np.arange(0, 100, 10)) # from 0 to 100 with 10-step hopping
```

NumPy

- In the statistics, we have to use random variable from specific distribution, such as, normal, binomial, and uniform distribution.

```
# normal distribution
nor1 = random.normal(size=(100, 3))
print(nor1)
# given an average and standard deviation
nor2 = random.normal(loc=5, scale=1, size=(100, 3))
print(nor2)
# but how to show whether they are normal distribution?
import matplotlib.pyplot as plt
plt.hist(nor1) # you may try nor2
plt.show()
```

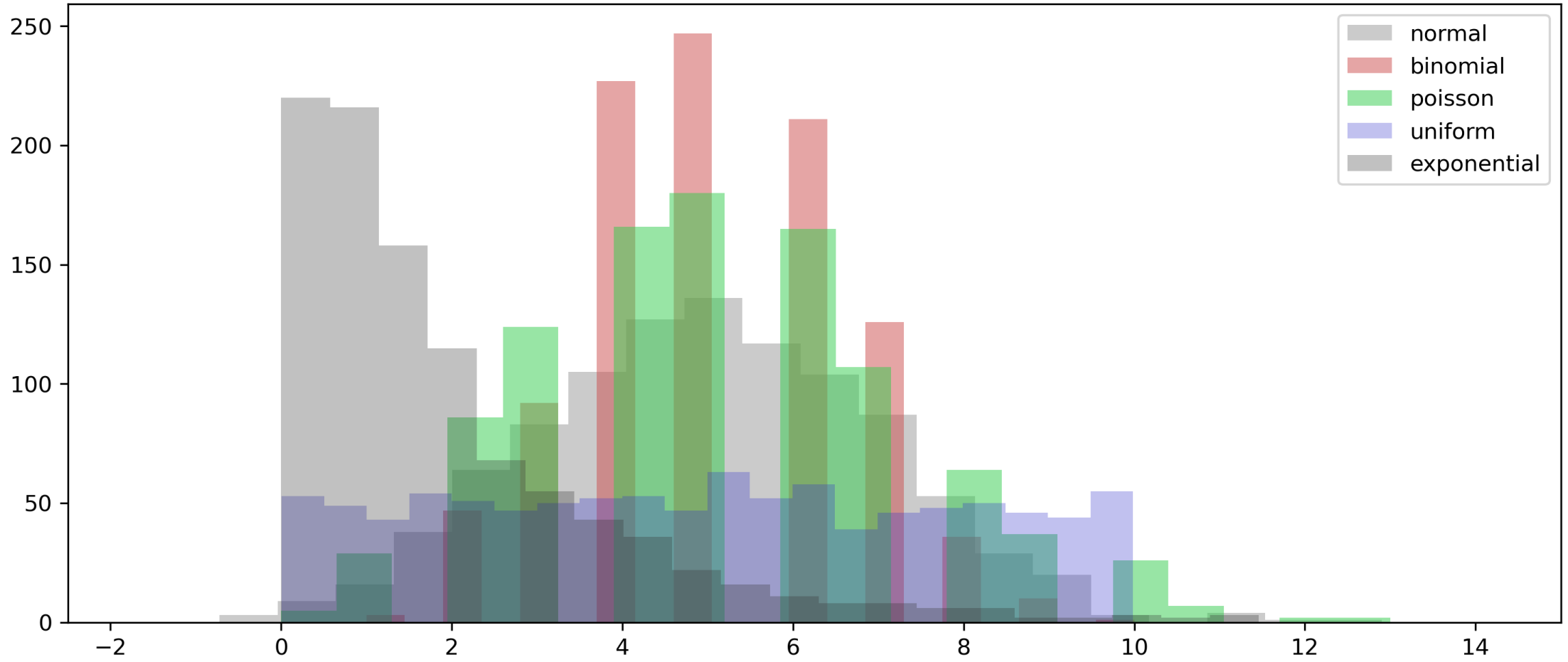
NumPy

- After demonstrating normal distribution, you may try other distribution as you wish.

```
plt.hist(random.normal(loc=5, scale=2, size=1000), bins=20, color=(0.6, 0.6, 0.6,
        0.5), label='normal')
plt.hist(random.binomial(n=10, p=0.5, size=1000), bins=20, color=(0.8, 0.3, 0.3,
        0.5), label='binomial')
plt.hist(random.poisson(lam=5, size=1000), bins=20, color=(0.2, 0.8, 0.3, 0.5),
        label='poisson')
plt.hist(random.uniform(size=1000)*10, bins=20, color=(0.2, 0.2, 0.8, 0.3),
        label='uniform')
plt.hist(random.exponential(scale=2, size=1000), bins=20, color=(0.2, 0.2, 0.2,
        0.3), label='exponential')
plt.legend()
plt.show()
```

NumPy

When **n** is very large, …



Pandas

- Pandas is another powerful Python library for data cleaning or data preprocessing, even for data analysis.
- Basically, you can imagine pandas as excel or table-like data (panel data).

```
# import package
import pandas as pd
# declare dict
ds = {'name':['mike','amy','may'], 'score':[70, 80, 23]}
df = pd.DataFrame.from_dict(ds) # use dict
df.head()
```

Pandas

- In addition to dictionary, we may also use list to construct a Pandas dataframe.

```
# declare list
ls = [['mike','amy','may','joy','mai','zoy','hoy'], [70, 80, 23 , 50, 72 ,
        82, 73]]
df2 = pd.DataFrame.from_records(np.column_stack(ls),
        columns=['name','score']) # use list
# how many rows show in the following code?
df2.head()
# try this
df2.head(7)
```

Pandas

- Get to know the data type and indexing of pandas dataframe.

```
# data type
print(type(df2))
print(type(df2['name']))
# indexing
print(df2['name'][2])
# use iloc
df2.iloc[:2, :4]
# use loc
df2.loc[1:3,'score']
# ask data row if it has may and mike
df2['name'].isin(['may','mike'])
```

```
# get data based on a condition
df2[df2['name'].isin(['may','mike'])]
# get the data type of elements
print(type(df2['score'][0]))
# change data type
df2['score'].astype(int)
# get column values
df2['score'].values
```

Pandas

```
# use two or more conditions
df2.loc[(df2['score']>60) & (df2['score']<80)]
# use two or more conditions and reset index
df2.loc[(df2['score']>60) & (df2['score']<80)].reset_index()
# use two or more conditions and reset index
df2.loc[(df2['score']>60) & (df2['score']<80)].reset_index(drop=True)
# what is the difference between abovementioned codes?

# use two or more conditions and reset index
df2.loc[(df2['score']>60) | (df2['score']<80)].reset_index(drop=True)
```

Pandas

- Sorting in Pandas dataframe

```
# sorting by column value
# get sorted value
df2['score'].sort_values()
# get sorted index
df2['score'] = df2['score'].sort_values().index
# get sorted index for re-order dataframe
df2.iloc[df2['score'].sort_values().index]
# how to re-index the sorted dataframe?
```

Pandas

- What is the difference between merge and concat?

```
df1 = pd.DataFrame({
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    },index=[0, 1, 2, 3],)

df2 = pd.DataFrame({
        "B": ["B2", "B3", "B6", "B7"],
        "D": ["D2", "D3", "D6", "D7"],
        "F": ["F2", "F3", "F6", "F7"],
    },index=[0, 1, 2, 3],)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	B	D	F
0	B2	D2	F2
1	B3	D3	F3
2	B6	D6	F6
3	B7	D7	F7

Pandas

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	B	D	F
0	B2	D2	F2
1	B3	D3	F3
2	B6	D6	F6
3	B7	D7	F7

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

	A	B	C	D	F
0	A0	B0	C0	D0	NaN
1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	NaN
0	NaN	B2	NaN	D2	F2
1	NaN	B3	NaN	D3	F3
2	NaN	B6	NaN	D6	F6
3	NaN	B7	NaN	D7	F7

pd.concat([df1, df2], axis=1)

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	B2	D2	F2
1	A1	B1	C1	D1	B3	D3	F3
2	A2	B2	C2	D2	B6	D6	F6
3	A3	B3	C3	D3	B7	D7	F7

Pandas

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	B	D	F
0	B2	D2	F2
1	B3	D3	F3
2	B6	D6	F6
3	B7	D7	F7

```
df1.merge(left_on='B', right=df2, right_on='B', how='left')
```

left

	A	B	C	D_x	D_y	F
0	A0	B0	C0	D0	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN
2	A2	B2	C2	D2	D2	F2
3	A3	B3	C3	D3	D3	F3

inner

	A	B	C	D_x	D_y	F
0	A2	B2	C2	D2	D2	F2
1	A3	B3	C3	D3	D3	F3

right

	A	B	C	D_x	D_y	F
0	A2	B2	C2	D2	D2	F2
1	A3	B3	C3	D3	D3	F3
2	NaN	B6	NaN	NaN	D6	F6
3	NaN	B7	NaN	NaN	D7	F7

outer

	A	B	C	D_x	D_y	F
0	A0	B0	C0	D0	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN
2	A2	B2	C2	D2	D2	F2
3	A3	B3	C3	D3	D3	F3
4	NaN	B6	NaN	NaN	D6	F6
5	NaN	B7	NaN	NaN	D7	F7

Pandas

- groupby and drop duplicate

```
ls = [['mike','amy','may','joy','mai','zoy','hoy','mai','zoy','hoy'], [70, 80, 23 , 50, 72 , 82, 73 , 80, 100, 45]]
df2 = pd.DataFrame.from_records(np.column_stack(ls),
          columns=['name','score'])
# drop duplicates
df2.drop_duplicates(['name'])
# groupby with count, mean, median, std, sum, …
df2.groupby(['name']).count().reset_index()
# try and observe
```

Lab Practice 1

- Read File (csv, xlsx, xls, and json…)

```
# read file
df = pd.read_csv('Airline Dataset Updated - v2.csv')
df.head()
```

	Passenger ID	First Name	Last Name	Gender	Age	Nationality	Airport Name	Airport Country Code	Country Name	Airport Continent	Continents	Departure Date	Arrival Airport	Pilot Name	Flight Status
0	ABVWIg	Edithe	Leggis	Female	62	Japan	Coldfoot Airport	US	United States	NAM	North America	6/28/2022	CXF	Fransisco Hazeldine	On Time
1	jkXXAX	Elwood	Catt	Male	62	Nicaragua	Kugluktuk Airport	CA	Canada	NAM	North America	12/26/2022	YCO	Marla Parsonage	On Time
2	CdUz2g	Darby	Felgate	Male	67	Russia	Grenoble-Isère Airport	FR	France	EU	Europe	1/18/2022	GNB	Rhonda Amber	On Time
3	BRS38V	Dominica	Pyle	Female	71	China	Ottawa / Gatineau Airport	CA	Canada	NAM	North America	9/16/2022	YND	Kacie Commucci	Delayed
4	9kvTLo	Bay	Pencost	Male	21	China	Gillespie Field	US	United States	NAM	North America	2/25/2022	SEE	Ebonee Tree	On Time

Lab Practice 1

- Number of pax took an On Time flight
- How many arrival airports in the dataset?
- How many men and women in the dataset?
- How many pax in the dataset?
- How many countries in the dataset?
- Which country has the highest number of pax in this dataset?
- Which airport has the highest number of pax in this dataset?
- How many male Russian pax arrive in France?
- How many female Japan pax arrive in Canada on time?
- Follow the previous question, what is the average age of them?

Question Time

- **Assignment:**
- **Download today's lab practice and upload to moodle.**
- **Thx**

The End

Thank you for your attention!

Email: chchan@ntnu.edu.tw
Web: toodou.github.io

SCAN ME

SCAN ME