Lab #2: Pandas + Matlotlib

This lab is the first step to getting familiar with Python and a common Machine Learning library, named **Pandas** and **Matlotlib**.

==========

SECTION 1. PANDAS

==========

Pandas:

- A very powerful package of Python for manipulating tables.
- Built on top of Numpy, so is efficient.
- Save you a lot of effort from writing lower Python code for manipulating, extracting, and deriving table-related information.
- Easy visualization with Matplotlib.
- Main data structures – Series and DataFrame

Import:

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

Series: an indexed 1D array

```
data = pd.Series([0.25, 0.5, 0.75, 1.0])
data
```

```
0    0.25
1    0.50
2    0.75
3    1.00
dtype: float64
```

Explicit index

```
data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=['a','b','c','d'])
data
```

```
a    0.25
b    0.50
c    0.75
d    1.00
dtype: float64
```

Access data

```
data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=['a','b','c','d'])
data['b']
```

```
0.5
```

Can work as a dictionary

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
pop
```

```
Ha Noi          100
Ho Chi Minh     300
Binh Duong      250
Vung Tau        150
dtype: int64
```

Access and slice data

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
pop["Binh Duong"]
```

```
250
```

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
pop["Ha Noi":"Binh Duong"]
```

```
Ha Noi          100
Ho Chi Minh     300
Binh Duong      250
dtype: int64
```

DataFrame Object: Generalized two dimensional array with flexible row and column indices

Constructing a dataframe from a dictionary

```
dat = {'col1':[1,2, 3], 'col2':[4, 5, 6], 'col3':[7, 8, 9]}
df = pd.DataFrame(data=dat)
df
```

	col1	col2	col3
0	1	4	7
1	2	5	8
2	3	6	9

Constructing a dataframe from numpy ndarray

```
data = pd.DataFrame(np.random.randint(low=0, high=10, size=(5,5)),
                    columns=['a','b','c','d','e'])
data
```

	a	b	c	d	e
0	5	3	2	8	6
1	5	7	8	3	1
2	6	9	6	8	1
3	8	4	3	8	5
4	0	7	8	3	8

Constructing a dataframe from Pandas Series

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
pop
```

```
Ha Noi          100
Ho Chi Minh     300
Binh Duong      250
Vung Tau        150
dtype: int64
```

Constructing a dataframe from Pandas Series (cont.)

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df
```

	population	area
Ha Noi	100	3000
Ho Chi Minh	300	2500
Binh Duong	250	3500
Vung Tau	150	2800

Viewing Data

View the first or last N rows

//df.head(2), default 5

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.head(2)
```

	population	area
Ha Noi	100	3000
Ho Chi Minh	300	2500

//df.tail(2)

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.tail(2)
```

	population	area
Binh Duong	250	3500
Vung Tau	150	2800

Display the index, columns, and data

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.index
```

```
Index(['Ha Noi', 'Ho Chi Minh', 'Binh Duong', 'Vung Tau'], dtype='object')
```

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.columns
```

```
Index(['population', 'area'], dtype='object')
```

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.values
```

```
array([[ 100, 3000],
       [ 300, 2500],
       [ 250, 3500],
       [ 150, 2800]])
```

Quick statistics

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.describe()
```

	population	area
count	4.000000	4.00000
mean	200.000000	2950.00000
std	91.287093	420.31734
min	100.000000	2500.00000
25%	137.500000	2725.00000
50%	200.000000	2900.00000
75%	262.500000	3125.00000
max	300.000000	3500.00000

Sorting: sort by the index (i.e., reorder columns or rows), not by the data in the table

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.sort_index(axis=1, ascending=True)
```

	area	population
Ha Noi	3000	100
Ho Chi Minh	2500	300
Binh Duong	3500	250
Vung Tau	2800	150

Sorting: sort by the data values

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.sort_values(by="area")
```

	population	area
Ho Chi Minh	300	2500
Vung Tau	150	2800
Ha Noi	100	3000
Binh Duong	250	3500

Selecting Data

Selecting using a label

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.loc['Ho Chi Minh']
```

```
population     300
area          2500
Name: Ho Chi Minh, dtype: int64
```

Multi-axis, by label

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.loc[:,["area"]]
```

	area
Ha Noi	3000
Ho Chi Minh	2500
Binh Duong	3500
Vung Tau	2800

Select by position

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.iloc[2]
```

```
population     250
area          3500
Name: Binh Duong, dtype: int64
```

Boolean indexing

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df[df>200]
```

	population	area
Ha Noi	NaN	3000
Ho Chi Minh	300.0	2500
Binh Duong	250.0	3500
Vung Tau	NaN	2800

Setting Data

Setting values by label:

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.at['Lam Dong','area']=5000
df.at['Lam Dong','population']=500
df
```

	population	area
Ha Noi	100.0	3000.0
Ho Chi Minh	300.0	2500.0
Binh Duong	250.0	3500.0
Vung Tau	150.0	2800.0
Lam Dong	500.0	5000.0

Setting values by position:

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.iat[1,0]=500
df.iat[1,1]=5000
df
```

	population	area
Ha Noi	100	3000
Ho Chi Minh	500	5000
Binh Duong	250	3500
Vung Tau	150	2800

Setting by assigning a numpy array

```
pop_dict = {"Ha Noi": 100, "Ho Chi Minh": 300, "Binh Duong": 250, "Vung Tau": 150}
pop = pd.Series(pop_dict)
area_dict = {"Ha Noi": 3000, "Ho Chi Minh": 2500, "Binh Duong": 3500, "Vung Tau": 2800}
area = pd.Series(area_dict)
df = pd.DataFrame({'population':pop, 'area':area})
df.loc[:, 'Temp']=np.array([5]*len(df))
df
```

	population	area	Temp
Ha Noi	100	3000	5
Ho Chi Minh	300	2500	5
Binh Duong	250	3500	5
Vung Tau	150	2800	5

Operations

Across axis 0 (rows), i.e., column mean: **df.mean()**
Across axis 1 (column), i.e., row mean: **df.mean(1)**
Apply: allow the users to pass a function and apply it on every single value of the Pandas series.

```
s.apply(func, convert_dtype=True, args=())
```

where,

- **func:** .apply takes a function and applies it to all values of pandas series.
- **convert_dtype:**Convert dtype as per the function's operation.

– **args=():** Additional arguments to pass to function instead of series.

```
import pandas as pd

def calc_sum(x):
  return x.sum()

data = {
  "x": [50, 40, 30],
  "y": [300, 1112, 42]
}

df = pd.DataFrame(data)

x = df.apply(calc_sum)

print(x)
```

```
x     120
y    1454
dtype: int64
```

Histogram:

```
import pandas as pd
import numpy as np

s = pd.Series(np.random.randint(0,7,size=10))
print(s)
print(s.value_counts())
```

```
0    0
1    4
2    5
3    3
4    3
5    3
6    3
7    2
8    2
9    1
dtype: int64
3    4
2    2
0    1
4    1
5    1
1    1
dtype: int64
```

Merge Tables

```
left = pd.DataFrame({'key':['foo', 'bar'], 'lval':[1,2]})
right = pd.DataFrame({'key':['foo', 'bar'], 'rval':[4,5]})
pd.merge(left, right, on='key')
```

	key	lval	rval
0	foo	1	4
1	bar	2	5

```
left = pd.DataFrame({'key':['foo', 'bar'], 'lval':[1,2]})
right = pd.DataFrame({'key':['foo', 'bar'], 'rval':[4,5]})
re = pd.merge(left, right, on='key')
sub = re.iloc[1]
re.append(sub)
```

	key	lval	rval
0	foo	1	4
1	bar	2	5
1	bar	2	5

Grouping

Split Data into Groups

Pandas object can be split into any of their objects. There are multiple ways to split an object like −

- obj.groupby('key')
- obj.groupby(['key1','key2'])
- obj.groupby(key,axis=1)

```
# import the pandas library
import pandas as pd

ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
   'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
   'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
   'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
   'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
df = pd.DataFrame(ipl_data)

print (df.groupby('Team').groups)
```

```
{'Devils': [2, 3], 'Kings': [4, 6, 7], 'Riders': [0, 1, 8, 11], 'Royals': [9, 10], 'kings': [5]}
```

```
# import the pandas library
import pandas as pd

ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
   'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
   'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
   'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
   'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
df = pd.DataFrame(ipl_data)

print(df.groupby(['Team','Year']).groups)
```

File I/O

Reading CSV file: **pd.read_csv("filename.csv")**
Reading CSV file: **pd.read_excel("filename.csv", "Sheet1", index_col=None, na_values=["NA"])**