

PYTHON FOR MACHINE LEARNING, DATA SCIENCE & DATA VISUALIZATION

Bài 5: Pandas – Xử lý dữ liệu



Phòng LT & Mạng

Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

I/O – Đọc/ghi dữ liệu

□ Đọc/ghi dữ liệu

- Định dạng tập tin: .csv, .xlsx, .hdf...
- Nơi lưu trữ:
 - Máy tính cục bộ: \data\<tập tin data>
 - Internet

<https://storage.googleapis.com/tf-datasets/titanic/train.csv>

I/O – Đọc/ghi dữ liệu

□ File csv

- Đọc dữ liệu: dùng

```
pd.read_csv(đường_dẫn_tên_tập_tin, sep =  
dấu_phân_cách, index_col =  
index_cột, nrows = số_lượng_dòng,  
header = None – nếu không có dòng tiêu  
đề, usecols = [index_cột,...],  
encoding='utf-8')
```

- Kết quả trả về: DataFrame hoặc TextParser

I/O – Đọc/ghi dữ liệu

■ Ví dụ

```
df2 = pd.read_csv('daily_weather.csv', index_col = 0, sep=',', nrows=10)
print(df2.shape, type(df2))
```

```
(10, 10) <class 'pandas.core.frame.DataFrame'>
```

```
df2.head()
```

	air_pressure_9am	air_temp_9am	avg_wind_direction_9am	avg_wind_speed_9am	max_wind_direction_9am	max_wind_speed_9am
number						
0	918.060000	74.822000	271.100000	2.080354	295.400000	2.863283
1	917.347688	71.403843	101.935179	2.443009	140.471549	3.533324
2	923.040000	60.638000	51.000000	17.067852	63.700000	22.100967
3	920.502751	70.138895	198.832133	4.337363	211.203341	5.190045
4	921.160000	44.294000	277.800000	1.856660	136.500000	2.863283

I/O – Đọc/ghi dữ liệu

- Ghi dữ liệu: dùng

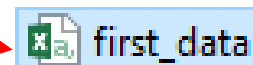
`df.to_csv(đường_dẫn_tên_tập_tin, sep=',',
, na_rep='', columns=None, header=True,
mode='w', encoding='utf-8', decimal='.')`

- Ví dụ

```
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f', 'h'], columns=['one', 'two', 'three'])
print(df)
```

	one	two	three
a	-1.282701	1.114661	1.025230
c	0.743163	-0.715507	-0.676640
e	1.238341	0.352514	0.345264
f	-0.209314	1.300077	-1.517246
h	1.225018	-0.322216	1.219772

```
df.to_csv('first_data.csv')
```



I/O – Đọc/ghi dữ liệu

❑ File excel

- Cài thư viện: `pip install xlrd`
- Đọc file chỉ có một sheet: dùng
`pd.read_excel(đường_dẫn_tên_tập_tin, encoding='utf-8', index_col = index_cột, nrows = số_lượng_dòng...)`
 - Kết quả trả về: DataFrame hoặc TextParser

I/O – Đọc/ghi dữ liệu

■ Ví dụ:

```
df3 = pd.read_excel('first_data_excel.xlsx', index_col = 0, encoding='utf-8')
print(df3.shape, type(df3))
```

```
(5, 3) <class 'pandas.core.frame.DataFrame'>
```

```
print(df3)
```

	one	two	three
a	-1.282701	1.114661	1.025230
c	0.743163	-0.715507	-0.676640
e	1.238341	0.352514	0.345264
f	-0.209314	1.300077	-1.517246
h	1.225018	-0.322216	1.219772

I/O – Đọc/ghi dữ liệu

□ File excel

- Đọc file có nhiều sheet: dùng `xlsx = pd.ExcelFile()` và `pd.read_excel(xlsx, "Tên sheet")`
 - Kết quả trả về: DataFrame hoặc TextParser

I/O – Đọc/ghi dữ liệu

■ Ví dụ

```
xlsx = pd.ExcelFile('movies.xlsx')
df4 = pd.read_excel(xlsx, 'movies')
print(df4.shape, type(df4))
```

```
(9125, 3) <class 'pandas.core.frame.DataFrame'>
```

```
print(df4.head(3))
```

	movieId	title \
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)

	genres
0	Adventure Animation Children Comedy Fantasy
1	Adventure Children Fantasy
2	Comedy Romance

I/O – Đọc/ghi dữ liệu

- Ghi file: dùng `df.to_excel(đường_dẫn_tên_tập_tin, "tên sheet")`
 - Ví dụ:

```
df5 = df4.head(10)
df5.to_excel("movies_sub.xlsx", "movies")
```



Sau khi đọc dữ liệu

❑ Kiểm tra kiểu dữ liệu các cột

Pandas Type	Description
object	numbers and strings
int64	Numeric characters
float64	Numeric characters with decimals
datetime64, timedelta[ns]	time data.

df.dtypes

❑ Quan sát dữ liệu: head, tail

Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

Làm sạch dữ liệu

❑ Dữ liệu ban đầu rất lộn xộn vì:

- Thiếu giá trị
- Có các giá trị ngoại lai (outlier)
- Có các giá trị không hợp lệ (ví dụ: giá trị âm cho tuổi, cân nặng)
- Chứa giá trị NaN (np.NaN), không có giá trị (None)

Làm sạch dữ liệu

❑ Xử lý các vấn đề về dữ liệu:

- Thay thế giá trị
- Điền bằng dữ liệu phía trên hoặc phía dưới cho những giá trị NaN
- Loại bỏ các field
- Nội suy giá trị

Làm sạch dữ liệu

□ Các phương thức

- Thay thế dữ liệu: Dùng
`df.replace(giá_trị_cũ, giá_trị_mới)`

■ Ví dụ:

```
# thay the gia tri 9999.0 bang 0
df3 = df3.replace(9999.0, 0)
```

```
print(df3)
```

	0	1
0	-0.349596	-2.01720
1	9999.000000	9999.000000
2	9999.000000	9999.000000
3	0.113800	0.61610
4	0.014700	-1.73166
5	9999.000000	9999.000000
6	1.233080	0.72010
7	9999.000000	9999.000000
8	9999.000000	9999.000000
9	9999.000000	9999.000000



```
print(df3)
```

	0	1
0	-0.349596	-2.01720
1	0.000000	0.000000
2	0.000000	0.000000
3	0.113800	0.61610
4	0.014700	-1.73166
5	0.000000	0.000000
6	1.233080	0.72010
7	0.000000	0.000000
8	0.000000	0.000000
9	0.000000	0.000000

Làm sạch dữ liệu

□ Các phương thức

- Điền bằng dữ liệu phía trên hoặc phía dưới cho những giá trị NaN: dùng

`df.fillna(method='ffill')` hoặc
`df.fillna(method='backfill')`

■ Ví dụ:

```
print(df4)
```

	0	1
0	-0.349596	-2.0172
1	NaN	NaN
2	1.578300	0.6374
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	-1.145700	0.0529

```
df5 = df4.fillna(method='ffill')
print(df5)
```

	0	1
0	-0.349596	-2.0172
1	-0.349596	-2.0172
2	1.578300	0.6374
3	1.578300	0.6374
4	1.578300	0.6374
5	1.578300	0.6374
6	-1.145700	0.0529

```
df6 = df4.fillna(method='backfill')
print(df6)
```

	0	1
0	-0.349596	-2.0172
1	1.578300	0.6374
2	1.578300	0.6374
3	-1.145700	0.0529
4	-1.145700	0.0529
5	-1.145700	0.0529
6	-1.145700	0.0529

Làm sạch dữ liệu

❑ Các phương thức

- Điền bằng dữ liệu khác:

```
myDataFrame.fillna(fill-value)
```

- `<df>.fillna(<giá trị>)`
- `<df>.fillna(value={'tên column': <giá trị>, ...})`

Làm sạch dữ liệu

- Xóa bỏ dòng dữ liệu có chứa NaN: dùng `df.dropna(axis=0 hoặc 1)`

■ Ví dụ:

```
print(df7)
```

	0	1	2
0	-0.349596	-2.01720	-0.33450
1	NaN	NaN	0.06800
2	1.578300	0.63740	0.07521
3	NaN	NaN	-0.25890
4	NaN	NaN	0.56550
5	NaN	NaN	-2.28520
6	-1.145700	0.05290	-1.85410
7	0.770500	0.08500	-0.68540
8	0.097600	0.13705	1.25890

```
# xóa dòng dữ liệu có NaN
df8 = df7.dropna(axis = 0)
print(df8)
```

	0	1	2
0	-0.349596	-2.01720	-0.33450
2	1.578300	0.63740	0.07521
6	-1.145700	0.05290	-1.85410
7	0.770500	0.08500	-0.68540
8	0.097600	0.13705	1.25890

```
# xóa cột dữ liệu có NaN
df9 = df7.dropna(axis = 1)
print(df9)
```

	2
0	-0.33450
1	0.06800
2	0.07521
3	-0.25890
4	0.56550
5	-2.28520
6	-1.85410
7	-0.68540
8	1.25890

Làm sạch dữ liệu

- Xóa bỏ dòng dữ liệu trùng
 - Ví dụ:

```

k1  k2
0  one  1
1  two  1
2  one  2
3  two  3
4  one  3
5  two  4
6  two  4

```

`data.duplicated()` →

```

0  False
1  False
2  False
3  False
4  False
5  False
6   True
dtype: bool

```

`data.drop_duplicates()` →

```

k1  k2
0  one  1
1  two  1
2  one  2
3  two  3
4  one  3
5  two  4

```

Làm sạch dữ liệu

- Thực hiện nội suy tuyến tính: dùng `df.interpolate()`

- Ví dụ

```
print(df7)
```



```
df10 = df7.interpolate()
print(df10)
```

	0	1	2
0	-0.349596	-2.01720	-0.33450
1	NaN	NaN	0.06800
2	1.578300	0.63740	0.07521
3	NaN	NaN	-0.25890
4	NaN	NaN	0.56550
5	NaN	NaN	-2.28520
6	-1.145700	0.05290	-1.85410
7	0.770500	0.08500	-0.68540
8	0.097600	0.13705	1.25890

	0	1	2
0	-0.349596	-2.017200	-0.33450
1	0.614352	-0.689900	0.06800
2	1.578300	0.637400	0.07521
3	0.897300	0.491275	-0.25890
4	0.216300	0.345150	0.56550
5	-0.464700	0.199025	-2.28520
6	-1.145700	0.052900	-1.85410
7	0.770500	0.085000	-0.68540
8	0.097600	0.137050	1.25890

Làm sạch dữ liệu

• Dùng hàm map

■ Ví dụ

	food	ounces
0	bacon	4.0
1	pulled pork	3.0
2	bacon	12.0
3	Pastrami	6.0
4	corned beef	7.5
5	Bacon	8.0
6	pastrami	3.0
7	honey ham	5.0
8	nova lox	6.0



```
meat_to_animal = {
    'bacon': 'pig',
    'pulled pork': 'pig',
    'pastrami': 'cow',
    'corned beef': 'cow',
    'honey ham': 'pig',
    'nova lox': 'salmon'
}
```



```
lowercased = data['food'].str.lower()
data['animal'] = lowercased.map(meat_to_animal)
```

	food	ounces	animal
0	bacon	4.0	pig
1	pulled pork	3.0	pig
2	bacon	12.0	pig
3	Pastrami	6.0	cow
4	corned beef	7.5	cow
5	Bacon	8.0	pig
6	pastrami	3.0	cow
7	honey ham	5.0	pig
8	nova lox	6.0	salmon

Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. **Trực quan hóa dữ liệu**
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết



Trực quan hóa dữ liệu

❑ Có thể trực quan hóa dữ liệu bằng biểu đồ:

Plotting

`DataFrame.plot` is both a callable method and a namespace attribute for specific plotting methods of the form `DataFrame.plot.<kind>`.

<code>DataFrame.plot([X, y, kind, ax,])</code>	DataFrame plotting accessor and method
<code>DataFrame.plot.area([X, y])</code>	Area plot
<code>DataFrame.plot.bar([X, y])</code>	Vertical bar plot.
<code>DataFrame.plot.barh([X, y])</code>	Make a horizontal bar plot.
<code>DataFrame.plot.box([by])</code>	Make a box plot of the DataFrame columns.
<code>DataFrame.plot.density([bw_method, ind])</code>	Generate Kernel Density Estimate plot using Gaussian kernels.
<code>DataFrame.plot.hexbin(X, y[, C, ...])</code>	Generate a hexagonal binning plot.
<code>DataFrame.plot.hist([by, bins])</code>	Draw one histogram of the DataFrame's columns.
<code>DataFrame.plot.kde([bw_method, ind])</code>	Generate Kernel Density Estimate plot using Gaussian kernels.
<code>DataFrame.plot.line([X, y])</code>	Plot DataFrame columns as lines.
<code>DataFrame.plot.pie([y])</code>	Generate a pie plot.
<code>DataFrame.plot.scatter(X, y[, s, c])</code>	Create a scatter plot with varying marker point size and color.
<code>DataFrame.boxplot([column, by, ax, ...])</code>	Make a box plot from DataFrame columns.
<code>DataFrame.hist([column, by, grid, ...])</code>	Make a histogram of the DataFrame's.

Trực quan hóa dữ liệu

□ Bar plot

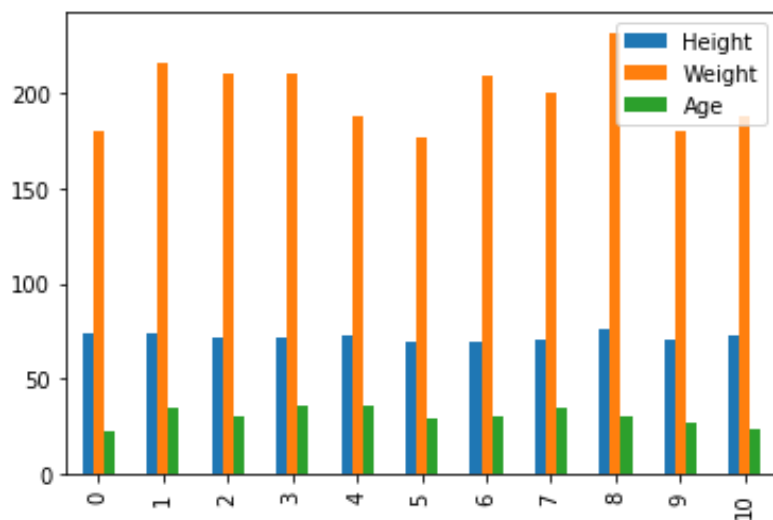
- Là dạng biểu đồ biểu diễn dữ liệu dưới dạng các khối chữ nhật, mô tả sự so sánh giữa các category khác nhau, một trục là category, một trục là giá trị của category đó.
- Tạo bar plot theo trục tung của Data frame columns: dùng `df.plot.bar()`

Trực quan hóa dữ liệu

• Ví dụ

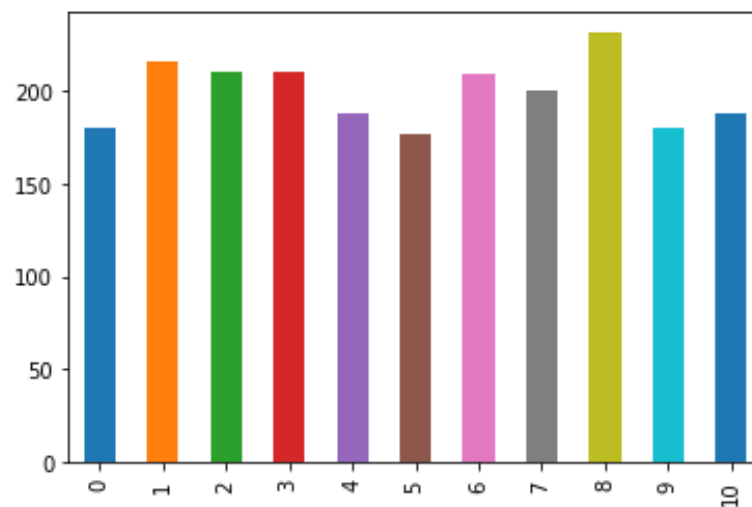
```
df.plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0xcdce7d0>



```
df['Weight'].plot.bar()
```

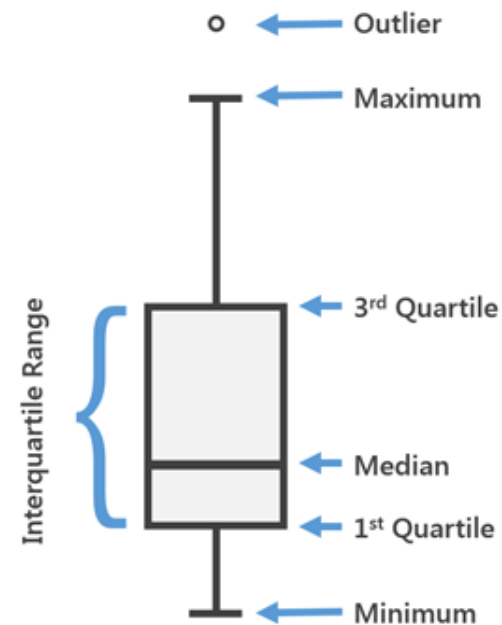
<matplotlib.axes._subplots.AxesSubplot at 0x16462d0>



Trực quan hóa dữ liệu

□ Boxplot

- Là dạng biểu đồ box trên đó mô tả các giá trị của dữ liệu như sau:



- Tạo boxplot của Data frame columns: dùng `df.plot.box()`

https://en.wikipedia.org/wiki/Box_plot

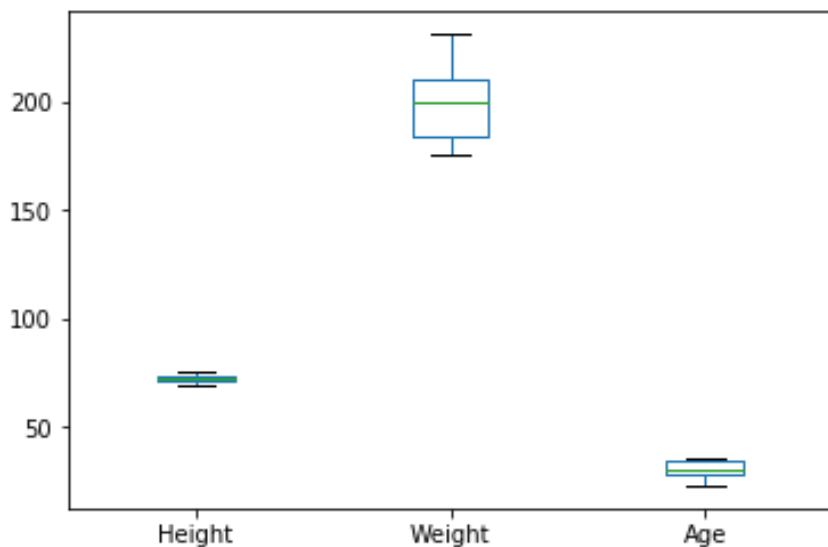
Trực quan hóa dữ liệu

Boxplot

• Ví dụ

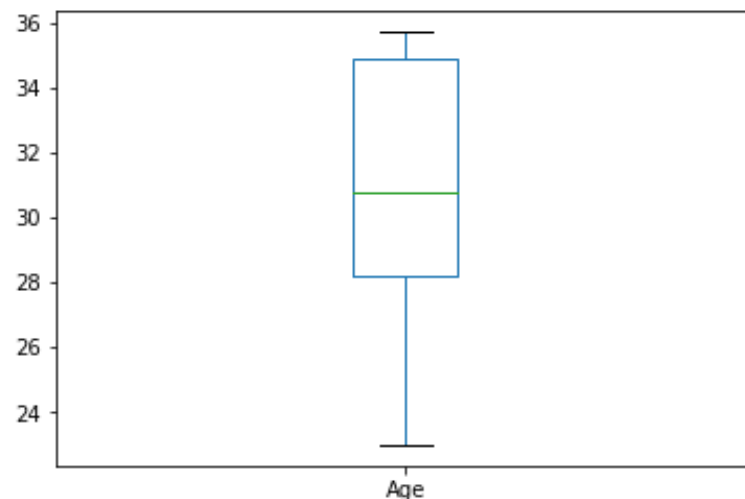
```
df.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xcde43b0>
```



```
df["Age"].plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1567e50>
```



Trực quan hóa dữ liệu

□ Histogram

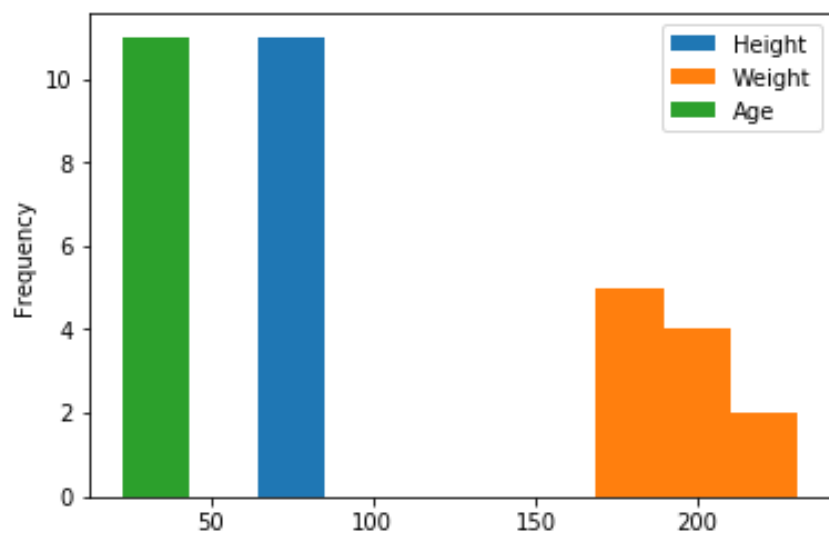
- Là dạng biểu đồ biểu diễn dạng phân phối tần suất của một tập dữ liệu liên tục.
- Tạo histogram của Data frame columns:
dùng `df.plot.hist()`

Trực quan hóa dữ liệu

• Ví dụ

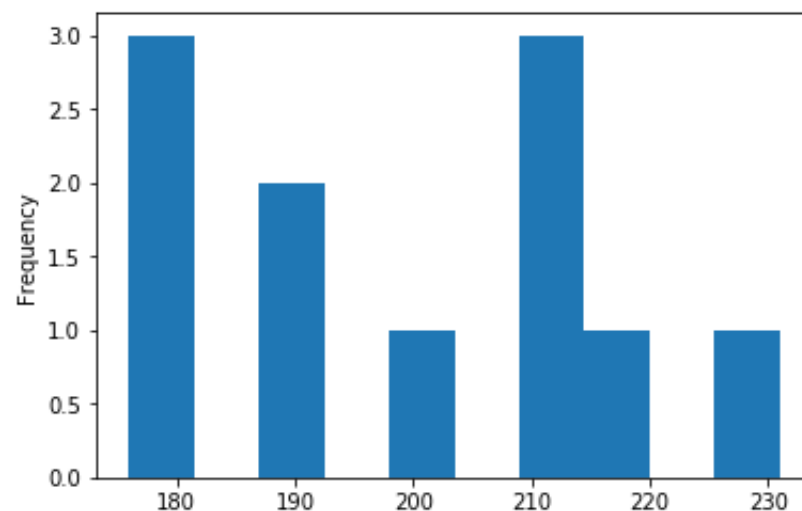
```
df.plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x5aeaef0>



```
df['Weight'].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0xcb6bc70>



Trực quan hóa dữ liệu

□ Plot

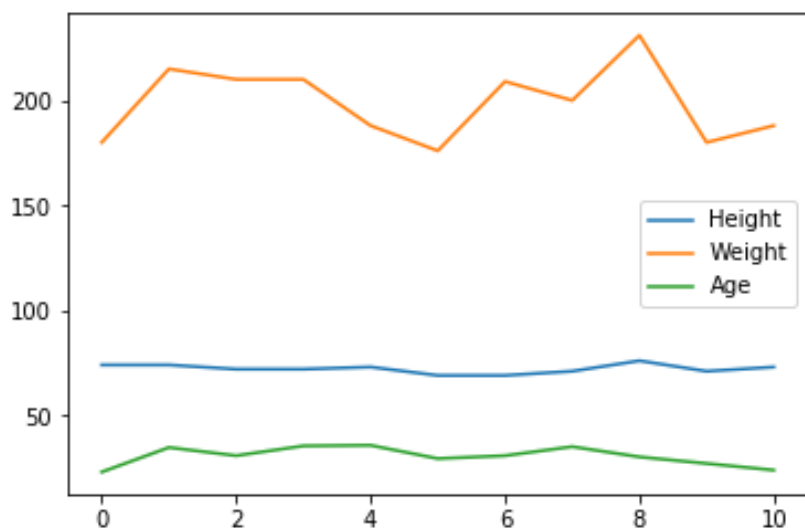
- Là dạng biểu đồ thể hiện các giá trị dưới các điểm nối với nhau bằng các line
- Tạo plot của Data frame columns: dùng `df.plot()`

Trực quan hóa dữ liệu

• Ví dụ

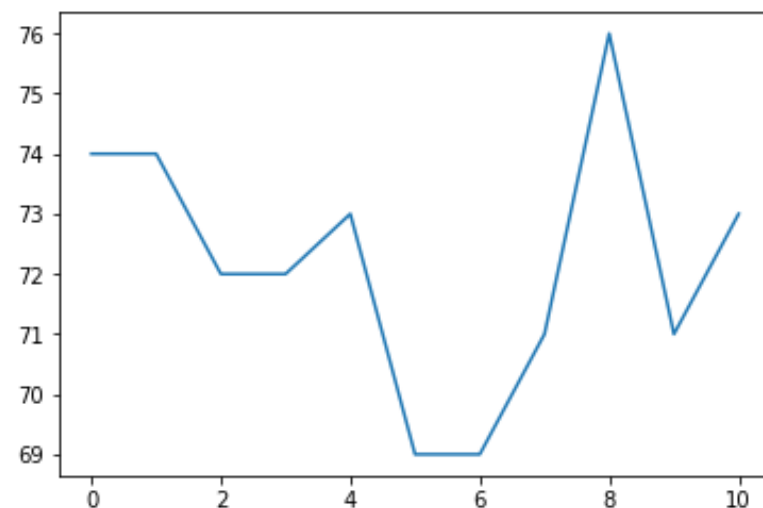
```
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xe179970>
```



```
df['Height'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16e6d70>
```



Trực quan hóa dữ liệu

□ Pie

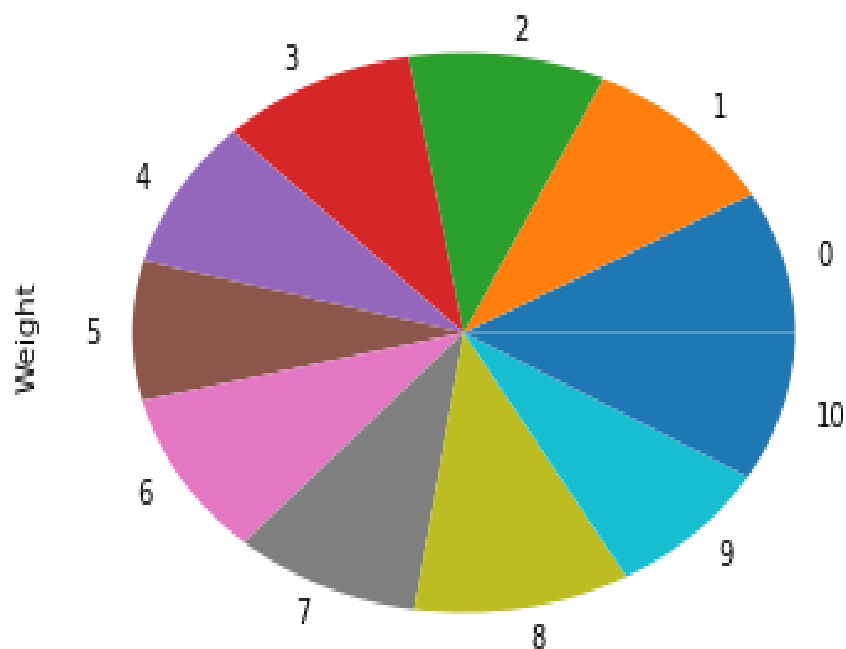
- Là dạng biểu đồ tròn mà mỗi giá trị là một miếng của biểu đồ tròn đó
- Tạo pie của Data frame column: dùng `df.plot.pie()`

Trực quan hóa dữ liệu

• Ví dụ

```
df['Weight'].plot.pie()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xcaa4430>
```



Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

Phương thức thao tác dữ liệu

□ Lấy dữ liệu cột: dùng tên cột

```
print(df)
```

	Name	Team	Position	Height	Weight	Age
0	Adam_Donachie	BAL	Catcher	74	180	22.99
1	Paul_Bako	BAL	Catcher	74	215	34.69
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78
3	Kevin_Millar	BAL	First_Baseman	72	210	35.43
4	Chris_Gomez	BAL	First_Baseman	73	188	35.71
5	Brian_Roberts	BAL	Second_Baseman	69	176	29.39
6	Miguel_Tejada	BAL	Shortstop	69	209	30.77
7	Melvin_Mora	BAL	Third_Baseman	71	200	35.07
8	Aubrey_Huff	BAL	Third_Baseman	76	231	30.19
9	Adam_Stern	BAL	Outfielder	71	180	27.05
10	Jeff_Fiorentino	BAL	Outfielder	73	188	23.88
11	Freddie_Bynum	BAL	Outfielder	73	180	26.96
12	Nick_Markakis	BAL	Outfielder	74	185	23.29
13	Brandon_Fahey	BAL	Outfielder	74	160	26.11
14	Corey_Patterson	BAL	Outfielder	69	180	27.55
15	Jay_Payton	BAL	Outfielder	70	185	34.27

```
tuoi = df['Age']
print(tuoi)
```

0	22.99
1	34.69
2	30.78
3	35.43
4	35.71
5	29.39
6	30.77
7	35.07
8	30.19
9	27.05
10	23.88
11	26.96
12	23.29
13	26.11
14	27.55
15	34.27

Name: Age, dtype: float64

Phương thức thao tác dữ liệu

❑ Lọc theo điều kiện

```
# tuổi lớn hơn 30
tuoi_hon_30 = df[df['Age'] > 30]
print(tuoi_hon_30)
```

	Name	Team	Position	Height	Weight	Age
1	Paul_Bako	BAL	Catcher	74	215	34.69
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78
3	Kevin_Millar	BAL	First_Baseman	72	210	35.43
4	Chris_Gomez	BAL	First_Baseman	73	188	35.71
6	Miguel_Tejada	BAL	Shortstop	69	209	30.77
7	Melvin_Mora	BAL	Third_Baseman	71	200	35.07
8	Aubrey_Huff	BAL	Third_Baseman	76	231	30.19
15	Jay_Payton	BAL	Outfielder	70	185	34.27

Phương thức thao tác dữ liệu

❑ Thêm cột mới

```
df['Height_m'] = df['Height'] * 0.0254
print(df.head(3))
```

	Name	Team	Position	Height	Weight	Age	Height_m
0	Adam_Donachie	BAL	Catcher	74	180	22.99	1.8796
1	Paul_Bako	BAL	Catcher	74	215	34.69	1.8796
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78	1.8288

❑ Thêm dòng mới

```
df.loc[df.shape[0]] = ['Jonny', 'BAL', 'Catcher', 74, 180, 22.99, 1.8796]
print(df.tail(3))
```

	Name	Team	Position	Height	Weight	Age	Height_m
14	Corey_Patterson	BAL	Outfielder	69	180	27.55	1.7526
15	Jay_Payton	BAL	Outfielder	70	185	34.27	1.7780
16	Jonny	BAL	Catcher	74	180	22.99	1.8796

Phương thức thao tác dữ liệu

❑ Xóa cột

```
del df['Height_m']
print(df.head(3))
```

	Name	Team	Position	Height	Weight	Age
0	Adam_Donachie	BAL	Catcher	74	180	22.99
1	Paul_Bako	BAL	Catcher	74	215	34.69
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78

❑ Xóa dòng

```
df = df.drop(df.index[[16]])
print(df.tail(3))
```

	Name	Team	Position	Height	Weight	Age
13	Brandon_Fahey	BAL	Outfielder	74	160	26.11
14	Corey_Patterson	BAL	Outfielder	69	180	27.55
15	Jay_Payton	BAL	Outfielder	70	185	34.27

Phương thức thao tác dữ liệu

□ Nhóm dữ liệu và tổng hợp

```
print(df)
```

	Name	Team	Position	Height	Weight	Age
0	Adam_Donachie	BAL	Catcher	74	180	22.99
1	Paul_Bako	BAL	Catcher	74	215	34.69
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78
3	Kevin_Millar	BAL	First_Base	72	210	35.43
4	Chris_Gomez	BAL	First_Base	73	188	35.71
5	Brian_Roberts	BAL	Second_Base	69	176	29.39
6	Miguel_Tejada	BAL	Shortstop	69	209	30.77
7	Melvin_Mora	BAL	Third_Base	71	200	35.07
8	Aubrey_Huff	BAL	Third_Base	76	231	30.19
9	Adam_Stern	BAL	Outfielder	71	180	27.05
10	Jeff_Florentino	BAL	Outfielder	73	188	23.88
11	Freddie_Bynum	BAL	Outfielder	73	180	26.96
12	Nick_Markakis	BAL	Outfielder	74	185	23.29
13	Brandon_Fahey	BAL	Outfielder	74	160	26.11
14	Corey_Patterson	BAL	Outfielder	69	180	27.55
15	Jay_Payton	BAL	Outfielder	70	185	34.27

```
df.groupby('Position').mean()
```

	Height	Weight	Age
Catcher	73.333333	201.666667	29.486667
First_Base	72.500000	199.000000	35.570000
Outfielder	72.000000	179.714286	27.015714
Second_Base	69.000000	176.000000	29.390000
Shortstop	69.000000	209.000000	30.770000
Third_Base	73.500000	215.500000	32.630000

Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

Gộp dữ liệu

□ Dùng `pd.concat([df1, df2,...])`

```
print(df_left)
```

	_key1	_key2	city	user_name
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3

→ # gộp 2 dữ liệu cùng cột
`df_1 = pd.concat([df_left, df_left])`
`print(df_1)`

	_key1	_key2	city	user_name
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3

Gộp dữ liệu

❑ Dùng `pd.concat([df1, df2, ...])`

```
print(df_left)
```

	_key1	_key2	city	user_name
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3

```
print(df_right)
```

	_key1	_key2	hide_date	profession
0	K0	z0	h_0	p_0
1	K1	z1	h_1	p_1
2	K2	z2	h_2	p_2
3	K3	z3	h_3	p_3



```
# gộp 2 dữ liệu khác cột
df_2 = pd.concat([df_left, df_right])
print(df_2)
```

	_key1	_key2	city	hide_date	profession	user_name
0	K0	z0	city_0	NaN	NaN	user_0
1	K1	z1	city_1	NaN	NaN	user_1
2	K2	z2	city_2	NaN	NaN	user_2
3	K3	z3	city_3	NaN	NaN	user_3
0	K0	z0	NaN	h_0	p_0	NaN
1	K1	z1	NaN	h_1	p_1	NaN
2	K2	z2	NaN	h_2	p_2	NaN
3	K3	z3	NaN	h_3	p_3	NaN

Gộp dữ liệu

❑ Kết dữ liệu: dùng `pd.concat([df1, df2, ...], axis = 1, join='inner')`

```
print(df_left)
```

	_key1	_key2	city	user_name
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3

```
print(df_right)
```

	_key1	_key2	hide_date	profession
0	K0	z0	h_0	p_0
1	K1	z1	h_1	p_1
2	K2	z2	h_2	p_2
3	K3	z3	h_3	p_3



```
df3 = pd.concat([df_left, df_right], axis=1, join='inner')
print(df3)
```

	_key1	_key2	city	user_name	_key1	_key2	hide_date	profession
0	K0	z0	city_0	user_0	K0	z0	h_0	p_0
1	K1	z1	city_1	user_1	K1	z1	h_1	p_1
2	K2	z2	city_2	user_2	K2	z2	h_2	p_2
3	K3	z3	city_3	user_3	K3	z3	h_3	p_3

Gộp dữ liệu

❑ Nối dữ liệu: dùng `df1.append(df2)`

```
print(df_left)
```

	_key1	_key2	city	user_name
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3

```
print(df_right)
```

	_key1	_key2	hide_date	profession
0	K0	z0	h_0	p_0
1	K1	z1	h_1	p_1
2	K2	z2	h_2	p_2
3	K3	z3	h_3	p_3



```
df = df_left.append(df_right)
print(df)
```

	_key1	_key2	city	hide_date	profession	user_name
0	K0	z0	city_0	NaN	NaN	user_0
1	K1	z1	city_1	NaN	NaN	user_1
2	K2	z2	city_2	NaN	NaN	user_2
3	K3	z3	city_3	NaN	NaN	user_3
0	K0	z0	NaN	h_0	p_0	NaN
1	K1	z1	NaN	h_1	p_1	NaN
2	K2	z2	NaN	h_2	p_2	NaN
3	K3	z3	NaN	h_3	p_3	NaN

Gộp dữ liệu

❑ Kết dữ liệu: dùng `pd.merge(df1, df2, how='inner')`


```
print(df_left)
```

	_key1	_key2	city	user_name
0	K0	z0	city_0	user_0
1	K1	z1	city_1	user_1
2	K2	z2	city_2	user_2
3	K3	z3	city_3	user_3

```
print(df_right)
```

	_key1	_key2	hide_date	profession
0	K0	z0	h_0	p_0
1	K1	z1	h_1	p_1
2	K2	z2	h_2	p_2
3	K3	z3	h_3	p_3

```
df_merge = pd.merge(df_left, df_right, how='inner')
print(df_merge)
```



	_key1	_key2	city	user_name	hide_date	profession
0	K0	z0	city_0	user_0	h_0	p_0
1	K1	z1	city_1	user_1	h_1	p_1
2	K2	z2	city_2	user_2	h_2	p_2
3	K3	z3	city_3	user_3	h_3	p_3

Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

Phương thức thao tác trên chuỗi

❑ Tách dữ liệu chuỗi thành list: dùng
`df["Tên_cột"].str.split("ký_tự_tách")`

● Ví dụ

```
df_merge = pd.merge(df_left, df_right, how='inner')
print(df_merge)
```

	_key1	_key2	city	user_name	hide_date	profession
0	K0	z0	city_0	user_0	h_0	p_0
1	K1	z1	city_1	user_1	h_1	p_1
2	K2	z2	city_2	user_2	h_2	p_2
3	K3	z3	city_3	user_3	h_3	p_3

```
city = df_merge['city'].str.split('_')
print(city)
```

```
0    [city, 0]
1    [city, 1]
2    [city, 2]
3    [city, 3]
Name: city, dtype: object
```

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

```
'2'
```



Phương thức thao tác trên chuỗi

❑ Tìm chuỗi có nằm trong chuỗi hay không: dùng

`df["Tên_cột"].str.contains("chuỗi")`

● Ví dụ

```
city_find = df_merge['city'].str.contains("2")  
print(city_find)
```

```
0    False  
1    False  
2     True  
3    False  
Name: city, dtype: bool
```

Phương thức thao tác trên chuỗi

❑ Thay chuỗi bằng chuỗi: dùng

`df["Tên_cột"].str.replace("chuỗi_cũ",
"chuỗi_mới")`

● Ví dụ

```
city_new = df_merge['city'].str.replace('_', ' No ')  
print(city_new)
```

```
0    city No 0  
1    city No 1  
2    city No 2  
3    city No 3  
Name: city, dtype: object
```

Phương thức thao tác trên chuỗi

❑ Tìm chuỗi đầu tiên thỏa regular expression (RE): dùng

`df["tên_cột"].str.extract('RE')`

● Ví dụ:

```
city_extract = df_left['city'].str.extract('([a-z]{0,})', expand=True)
print(type(city_extract))
print(city_extract)
```

```
<class 'pandas.core.frame.DataFrame'>
0
0 city
1 city
2 city
3 city
```

```
city_num = df_left['city'].str.extract('(\d)', expand=False)
print(type(city_num))
print(city_num)
```

```
<class 'pandas.core.series.Series'>
0    0
1    1
2    2
3    3
```

Name: city, dtype: object

Regular Expression

- RegEx là chuỗi ký tự đặc biệt để so khớp hoặc so sánh chuỗi thỏa điều kiện nào đó.
- Ví dụ:
 - `^a...s$`
 - `[0-9]{2,4}`
- Để sử dụng thư viện RegEx : **`import re`**

Regular Expression

- Một số ký hiệu:

- Hoặc : |
- Nhóm : ()
- Số lượng ký tự : $?^{*+}\{m,n\}$
- Ký tự đánh dấu : ^ \$
- Ký tự meta : . [] [-][^]
- Ký tự : \d\D\w\W...

- Ví dụ:

- “cat|mat” ~ “cat” or “mat”
- “gr(e|a)y” ~ “grey” or “gray”

Regular Expression

- Số lượng ký tự: $?^{*}+\{m,n\}$
- Ví dụ:
 - “colou?r” ~ “colour” or “color”
 - “94*9” ~ “99” or “9449” or “944449”
 - “36+40” ~ “3640” or “366640”
 - “go{2,3}gle” ~ “google” or “gooogle”
 - “9{3}” ~ “999”
 - “s{2,}” ~ “ss” or “sss” or “sssss”

Regular Expression

- Ký tự đánh dấu : ^ \$
- Ví dụ:
 - “^object” ~ “object” or “object-oriented” ...
 - “^2020” ~ “2020” or “2020/01/05” ...
 - “er\$” ~ “driver” or “programer” ...
 - “2019\$” ~ “2019” or “05/01/2019” ...

Regular Expression

- Ký tự meta : . [] [-][^]
- Ví dụ:
 - “87.1” ~ “8721” or “8731” or “8751”
 - “[xyz]” ~ “x” or “y” or “z”
 - “[a-zA-Z]” -> tất cả ký tự (chữ hoa, chữ thường)
 - “[^0-9]” -> Không lấy các ký số từ 0-9

Regular Expression

- Ký tự : `\d\D\w\W...`
- Ví dụ:
 - `\d` : ký số [0-9]
 - `\D` : không phải ký số
 - `\s` : ký tự đơn là tab(`\t`), newline (`\n`), khoảng trắng (`\v`)
 - `\w` : ký tự [a-zA-Z0-9_]
 - `\w+` : 1 hoặc nhiều ký tự [a-zA-Z0-9_]

Regular Expression

```
# Giới tính -
df['female'] = df['data'].str.extract('(\d)', expand=True)
df
```

	data	female
0	Arizona 1 2014-12-23 3242.0	1
1	Iowa 1 2010-02-23 3453.7	1
2	Oregon 0 2014-06-20 2123.0	0
3	Maryland 0 2014-03-14 1123.6	0
4	Florida 1 2013-01-15 2134.0	1
5	Georgia 0 2012-07-14 2345.6	0

```
# Nơi đăng ký
df['state'] = df['data'].str.extract('([A-Z]\w{0,})', expand=True)
df
```

	data	female	date	score	state
0	Arizona 1 2014-12-23 3242.0	1	2014-12-23	3242.0	Arizona
1	Iowa 1 2010-02-23 3453.7	1	2010-02-23	3453.7	Iowa
2	Oregon 0 2014-06-20 2123.0	0	2014-06-20	2123.0	Oregon
3	Maryland 0 2014-03-14 1123.6	0	2014-03-14	1123.6	Maryland
4	Florida 1 2013-01-15 2134.0	1	2013-01-15	2134.0	Florida
5	Georgia 0 2012-07-14 2345.6	0	2012-07-14	2345.6	Georgia

	data
0	Arizona 1 2014-12-23 3242.0
1	Iowa 1 2010-02-23 3453.7
2	Oregon 0 2014-06-20 2123.0
3	Maryland 0 2014-03-14 1123.6
4	Florida 1 2013-01-15 2134.0
5	Georgia 0 2012-07-14 2345.6

Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

Phương thức thao tác trên Timestamps

❑ Unix time / POSIX time / epoch time

- Số giây trôi qua từ:
 - 00:00:00
 - Giờ quốc tế - Coordinated Universal Time (UTC)
 - Thursday, 1 January 1970
- Prominent trong UNIX system
- Phân tích Timestamp: đọc thêm phần POSIX time và hiểu exact time stamp

Phương thức thao tác trên Timestamps

□ Kiểu dữ liệu Timestamps

- Kiểu dữ liệu chung: datetime64[ns]
- Chuyển dữ liệu int64 timestamp sang dữ liệu DateTime của Python

Phương thức thao tác trên Timestamps

❑ Chuyển Timestamps sang định dạng datetime của Python: dùng `pd.to_datetime(tên_cột_timestamp, unit='s')`

● Ví dụ

```
tags_sub = tags.head(5)
print(tags_sub)
```

userId	int64
movieId	int64
tag	object
timestamp	int64

	userId	movieId		tag	timestamp
0	15	339	sandra	'boring' bullock	1138537770
1	15	1955		dentist	1193435061
2	15	7478		Cambodia	1170560997
3	15	32892		Russian	1170626366
4	15	34162		forgettable	1141391765

```
# đổi thời gian
tags_sub = tags_sub.copy()
tags_sub['parsed_time'] = pd.to_datetime(tags_sub['timestamp'], unit='s')
print(tags_sub)
```

	userId	movieId		tag	timestamp	parsed_time
0	15	339	sandra	'boring' bullock	1138537770	2006-01-29 12:29:30
1	15	1955		dentist	1193435061	2007-10-26 21:44:21
2	15	7478		Cambodia	1170560997	2007-02-04 03:49:57
3	15	32892		Russian	1170626366	2007-02-04 21:59:26
4	15	34162		forgettable	1141391765	2006-03-03 13:16:05

Phương thức thao tác trên Timestamps

❑ Chọn dòng dựa trên Timestamp

```
year_t = tags_sub['parsed_time'] > '2007-01-01'
print(year_t)
selected_rows = tags_sub[year_t]
print(selected_rows)
```

```
0    False
1     True
2     True
3     True
4    False
```

Name: parsed_time, dtype: bool

	userId	movieId	tag	timestamp		parsed_time
1	15	1955	dentist	1193435061	2007-10-26	21:44:21
2	15	7478	Cambodia	1170560997	2007-02-04	03:49:57
3	15	32892	Russian	1170626366	2007-02-04	21:59:26

Phương thức thao tác trên Timestamps

- ❑ Sắp xếp dữ liệu theo trật tự thời gian:
dùng `df.sort_values(by='tên_cột', ascending = True)`

● Ví dụ:

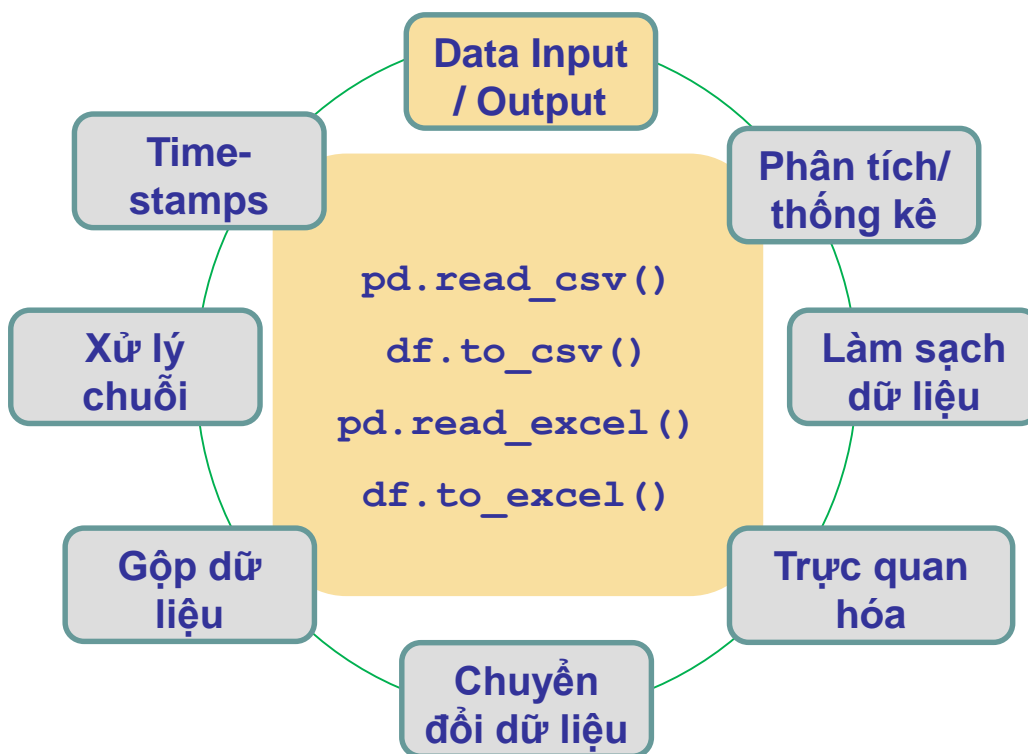
```
tags_sub = tags_sub.sort_values(by='parsed_time', ascending=True)
print(tags_sub)
```

	userId	movieId	tag	timestamp	parsed_time
0	15	339	sandra 'boring' bullock	1138537770	2006-01-29 12:29:30
4	15	34162	forgettable	1141391765	2006-03-03 13:16:05
2	15	7478	Cambodia	1170560997	2007-02-04 03:49:57
3	15	32892	Russian	1170626366	2007-02-04 21:59:26
1	15	1955	dentist	1193435061	2007-10-26 21:44:21

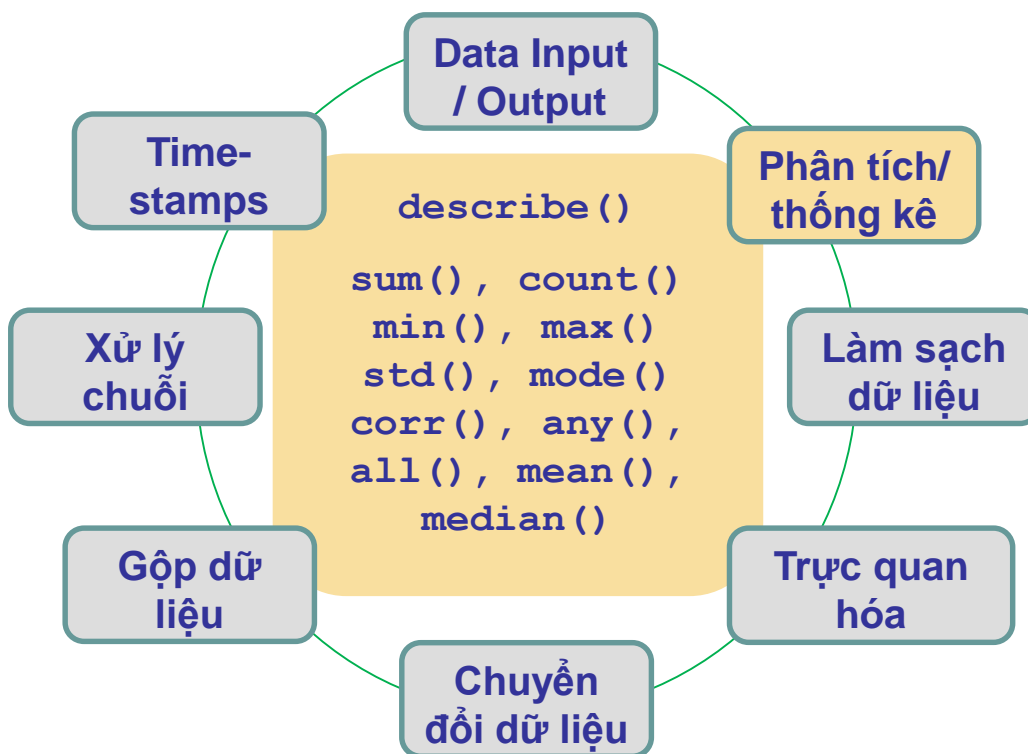
Nội dung

1. I/O – Đọc/ghi dữ liệu
2. Làm sạch dữ liệu
3. Trực quan hóa dữ liệu
4. Phương thức thao tác dữ liệu
5. Gộp dữ liệu
6. Phương thức thao tác trên chuỗi
7. Phương thức thao tác trên Timestamps
8. Tổng kết

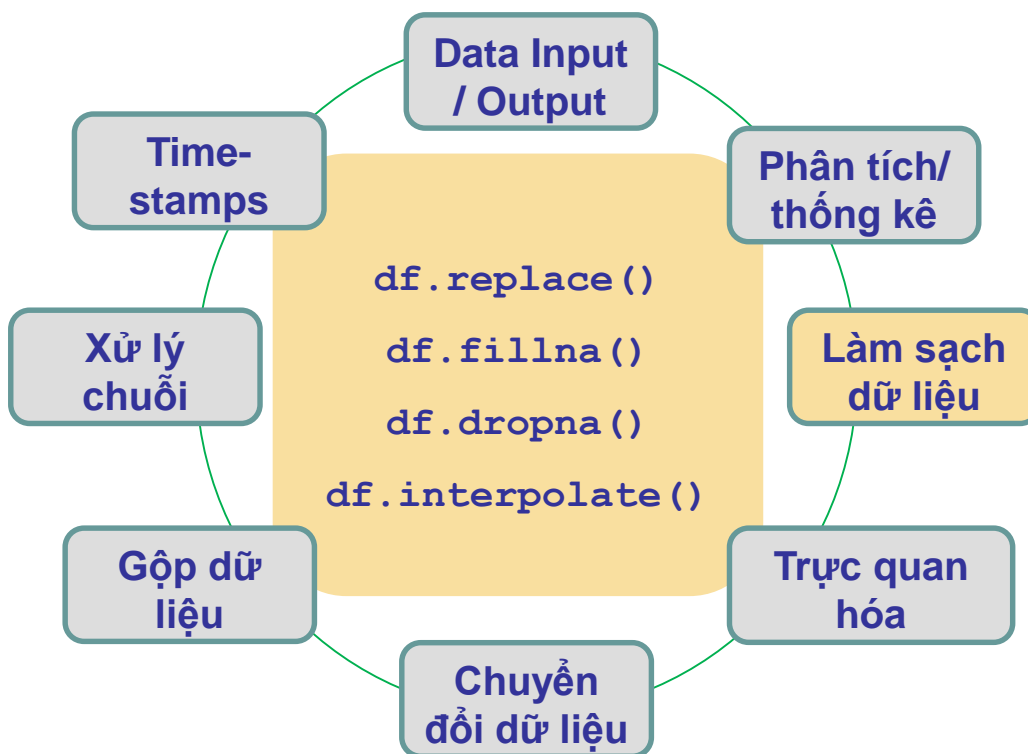
Tổng kết



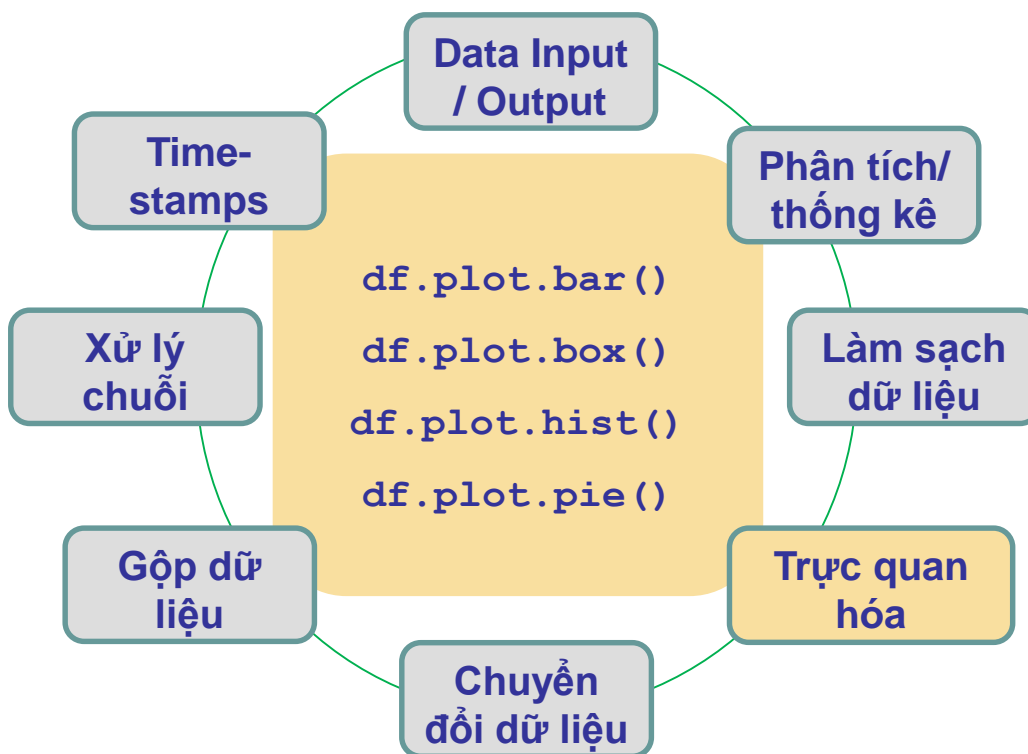
Tổng kết



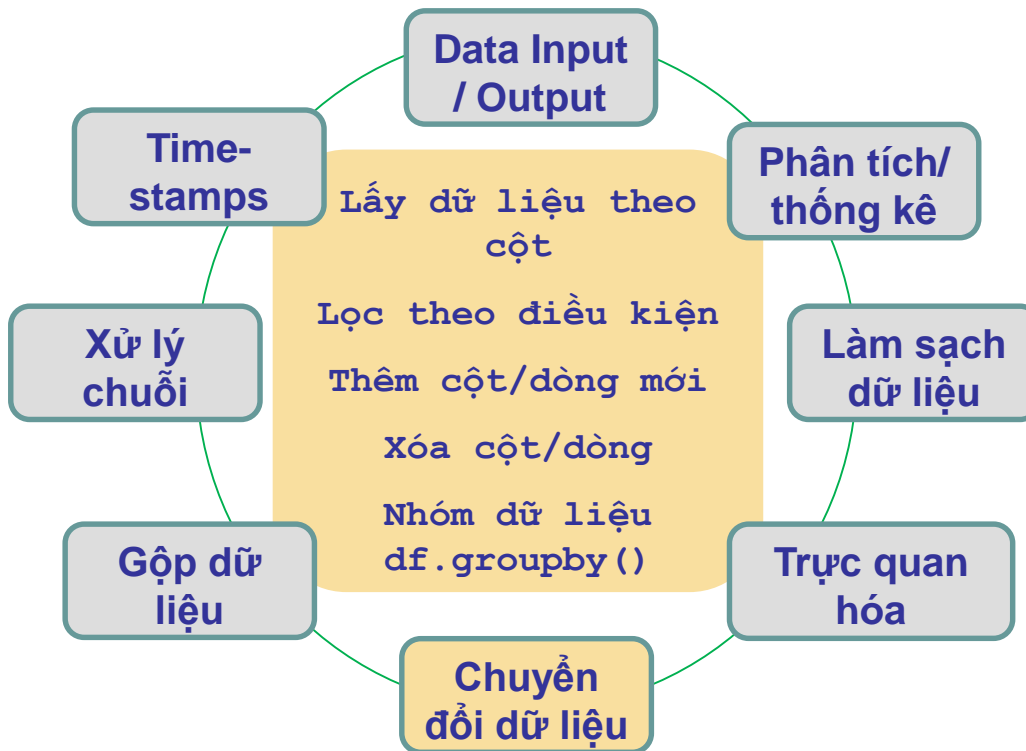
Tổng kết



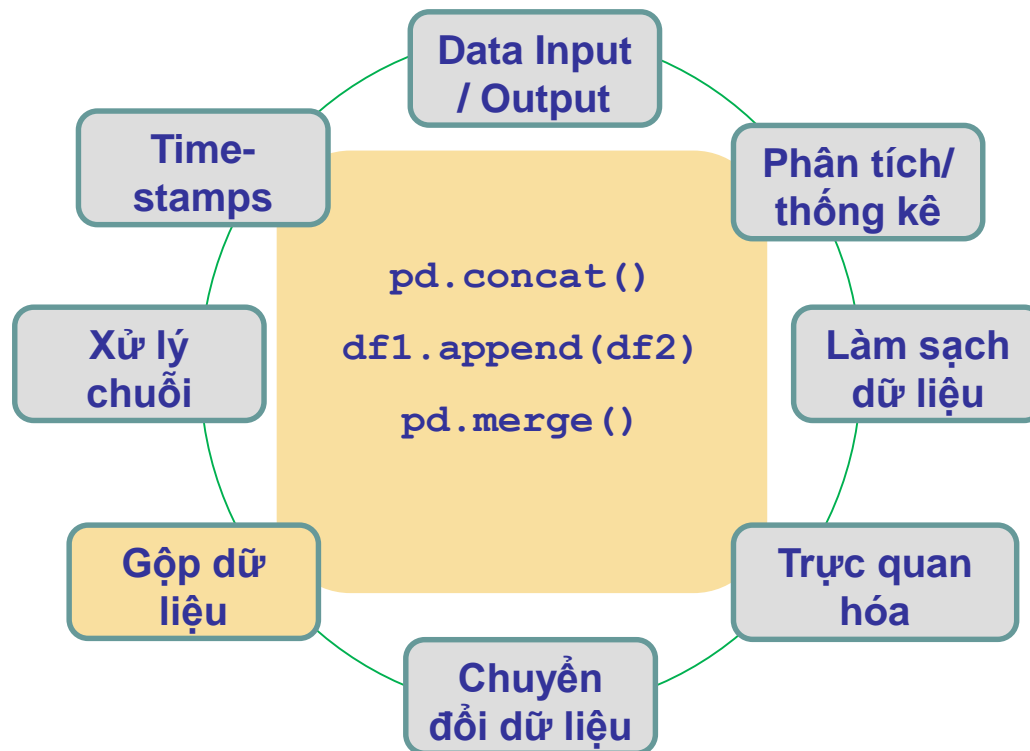
Tổng kết



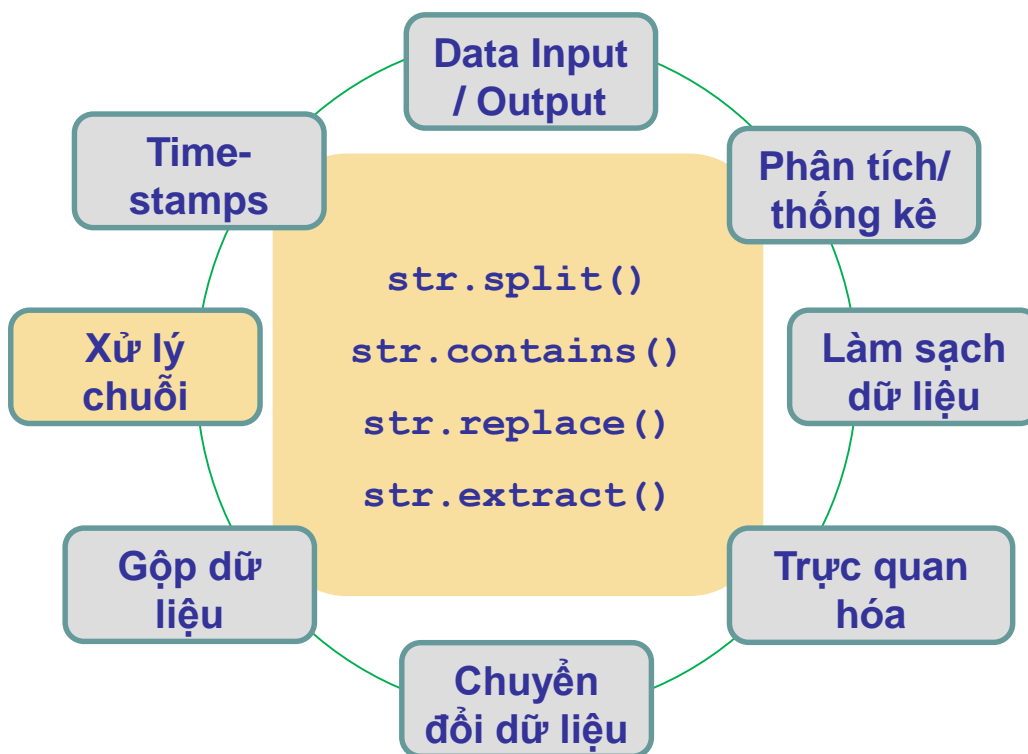
Tổng kết



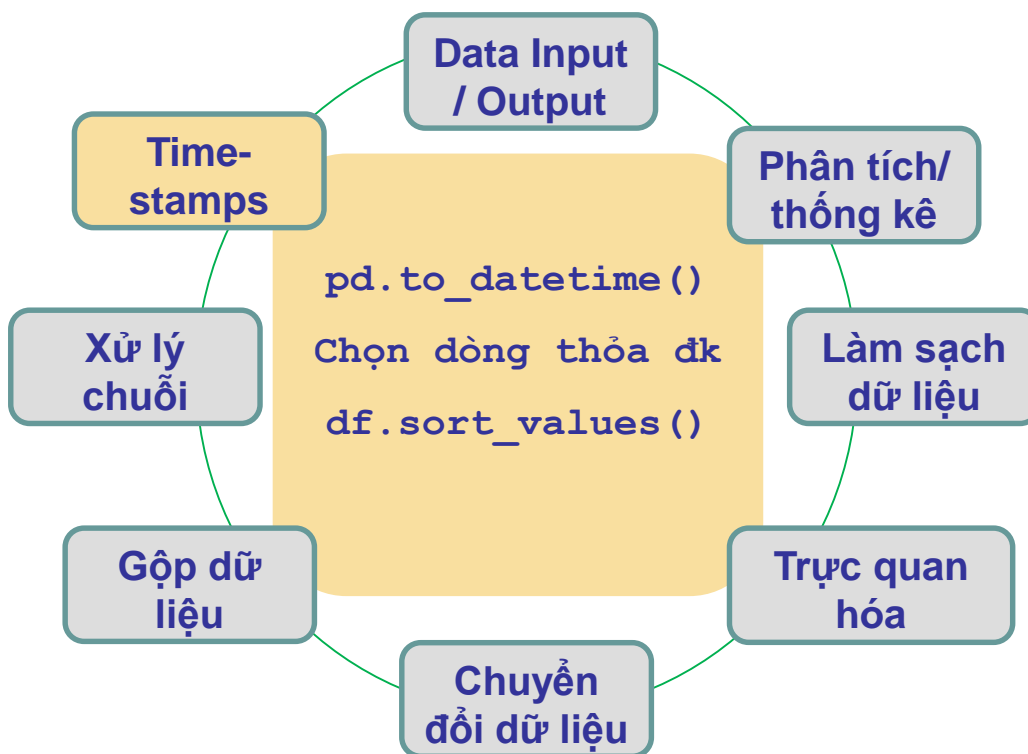
Tổng kết



Tổng kết



Tổng kết



Tổng kết

Scientifics Computing Libraries in Python

1. Scientifics Computing Libraries



Pandas

(Data structures & tools)



NumPy

(Arrays & matrices)



SciPy

(Integrals, solving differential equations, optimization)



Phân tích dữ liệu – Data Analysis

Tom wants to sell his car

Tom



How much
money should he
sell his car for?

The price he sets should not be too high,
but not too low either.

Phân tích dữ liệu – Data Analysis

Estimate used car prices

How can we help Tom determine the best price for his car?

- Is there data on the prices of other cars and their characteristics?
- What features of cars affect their prices?
 - Color? Brand? Horsepower? Something else?
- Asking the right questions in terms of data

Tom



No.	Attribute name	attribute range	No.	Attribute name	attribute range
1	symboling	-3, -2, -1, 0, 1, 2, 3.	14	curb-weight	continuous from 1488 to 4066.
2	normalized-losses	continuous from 65 to 256.	15	engine-type	dohc, dohcvt, l, ohc, ohcf, ohcv, rotor.
3	make	audi, bmw, etc.	16	num-of-cylinders	eight, five, four, six, three, twelve, two.
4	fuel-type	diesel, gas.	17	engine-size	continuous from 61 to 326.
5	aspiration	std, turbo.	18	fuel-system	1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
6	num-of-doors	four, two.	19	bore	continuous from 2.54 to 3.94.
7	body-style	hardtop, wagon, etc.	20	stroke	continuous from 2.07 to 4.17.
8	drive-wheels	4wd, fwd, rwd.	21	compression-ratio	continuous from 7 to 23.
9	engine-location	front, rear.	22	horsepower	continuous from 48 to 288.
10	wheel-base	continuous from 86.6 to 120.9.	23	peak-rpm	continuous from 4150 to 6600.
11	length	continuous from 141.1 to 208.1.	24	city-mpg	continuous from 13 to 49.
12	width	continuous from 60.3 to 72.3.	25	highway-mpg	continuous from 16 to 54.
13	height	continuous from 47.8 to 59.8.	26	price	continuous from 5118 to 45400.

Phân tích dữ liệu – Data Analysis

Basic insights from the data

- Understand your data before you begin any analysis
- Should check:
 - Data Types
 - Data Distribution
- Locate potential issues with the data

