Thao tác với dữ liệu sử dụng Pandas

Giảng viên: TS**. Nguyễn Văn Quyết**

Email: quyetict@gmail.com

Nguyễn Văn Quyết

- **Đào tạo:**
  - 2015-2019: Tiến sĩ KHMT, Đại học Quốc gia Chonnam, Hàn Quốc
  - 2011-2013: Thạc sĩ CNTT, Đại học Bách Khoa Hà Nội
- **Lĩnh vực nghiên cứu:**
  - Phân tích dữ liệu lớn (Big Data Analytics)
  - Xử lý đồ thị lớn (Big Graph Processing)
  - Tính toán song song (Parallel Computing)
  - Các hệ thống phân tán (Distributed Systems)
  - Kết nối vạn vật (Internet of Things)

Nội dung

- Giới thiệu về Pandas
- Cài đặt và sử dụng Pandas
- Các cấu trúc dữ liệu trong Pandas
- Xử lý dữ liệu khuyết thiếu
- Sắp xếp dữ liệu
- Thống kê dữ liệu

1. Giới thiệu về Pandas

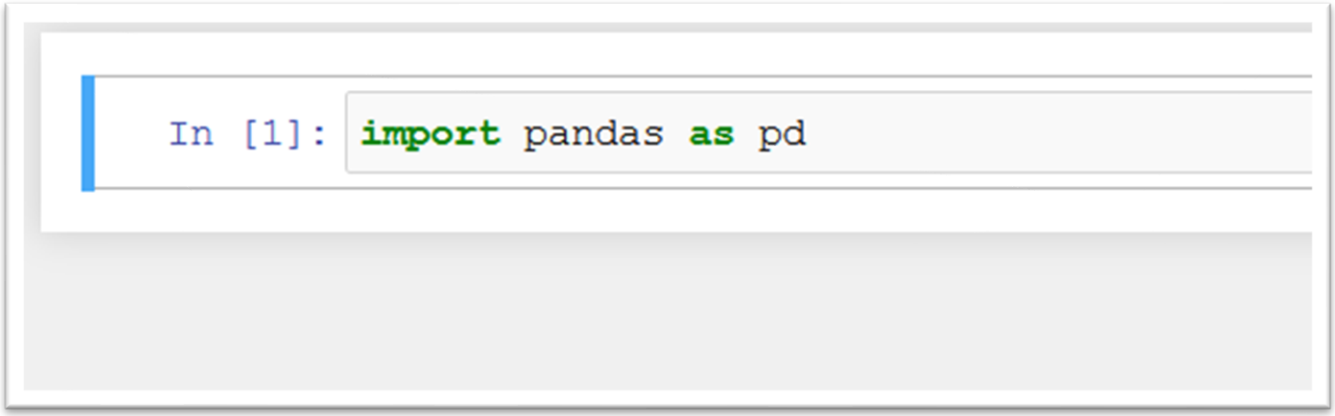
- Pandas là một thư viện chứa các công cụ lưu trữ với dữ liệu có cấu trúc bậc cao.
- Pandas được thiết kế nhằm mục đích đơn giản và làm cho việc phân tích dữ liệu trở nên nhanh chóng và dễ dàng.
- Pandas được xây dựng trên nền của NumPy làm cho nó dễ dàng được sử dụng trong các ứng dụng xoay quanh NumPy.

2. Cài đặt và sử dụng Pandas

- Sử dụng các câu lệnh sau để cài đặt Pandas:
  - conda install pandas #khi làm việc với Anaconda
  - pip install pandas

Import thư viện pandas

- Ta thực hiện import thư viện pandas bằng câu lệnh sau:
  - import pandas as pd



Series

- Series là mảng một chiều có thể chứa mọi kiểu dữ liệu (integer, string, floating point numbers, Python objects, etc).

Index	Data
1	'A'
2	'B'
3	'C'
4	'D'
5	'E'

Hàm tạo Series

- Hàm pandas.Series được dùng để tạo một đối tượng Series có cấu trúc như sau:
  - s = pd.Series(data, index=index)
  - Trong đó:
    - data là các đối tượng chứa dữ liệu (list, ndarray, số nguyên,…)
    - index là một list chứa các nhãn (label).

```
In [2]: #tạo một Series chứa chiều cao (cm)của 5 bạn trong lớp
        height=pd.Series([163,165,168,164,175],index=["Hùng","Nam","Long","Đặng","Tùng"])
        height

Out[2]: Hùng    163
        Nam     165
        Long    168
        Đặng    164
        Tùng    175
        dtype: int64
```

Dtype trong Series

- Series cũng chứa kiểu dữ liệu dtype giống như trong NumPy tuy nhiên do được mở rộng dựa trên NumPy nên dtype trong Series cũng có một số kiểu dữ liệu mở rộng.

```
In [3]: height.dtype
Out[3]: dtype('int64')
```

Truy xuất phần tử trong Series

- Phần tử trong Series được truy xuất giống như trong từ điển (dict) của Python bằng cách gọi kèm theo index.
  - s[index]

```
In [4]: #Lấy về chiều cao của Hùng
        height["Hùng"]

Out[4]: 163
```

Truy xuất phần tử trong Series

- Thay vì index được định nghĩa, ta có thể truy xuất bằng số index như trong mảng.

```
In [11]: height[0]

Out[11]: 163
```

Cắt (slicing) in Series

- Tương tự như trong NumPy, việc cắt Series tương tự như với ndarray tuy nhiên việc này sẽ khiến cả index bị cắt theo.

```
In [8]: rand_s=pd.Series(np.random.rand(5),index=[0,1,2,3,4])
        rand_s

Out[8]: 0    0.376878
        1    0.297313
        2    0.073761
        3    0.475148
        4    0.873543
        dtype: float64

In [9]: sliced_s1=rand_s[2:4]
        sliced_s1

Out[9]: 2    0.073761
        3    0.475148
        dtype: float64
```

Thao tác vector hóa

- Tương tự như mảng ndarray, Series cũng có thể hoạt động như một vector.

```
In [4]: s=pd.Series([1,2,3,4,5])
        s

Out[4]: 0    1
        1    2
        2    3
        3    4
        4    5
        dtype: int64

In [5]: s+s

Out[5]: 0     2
        1     4
        2     6
        3     8
        4    10
        dtype: int64

In [6]: s*2

Out[6]: 0     2
        1     4
        2     6
        3     8
        4    10
        dtype: int64
```

Thuộc tính Name

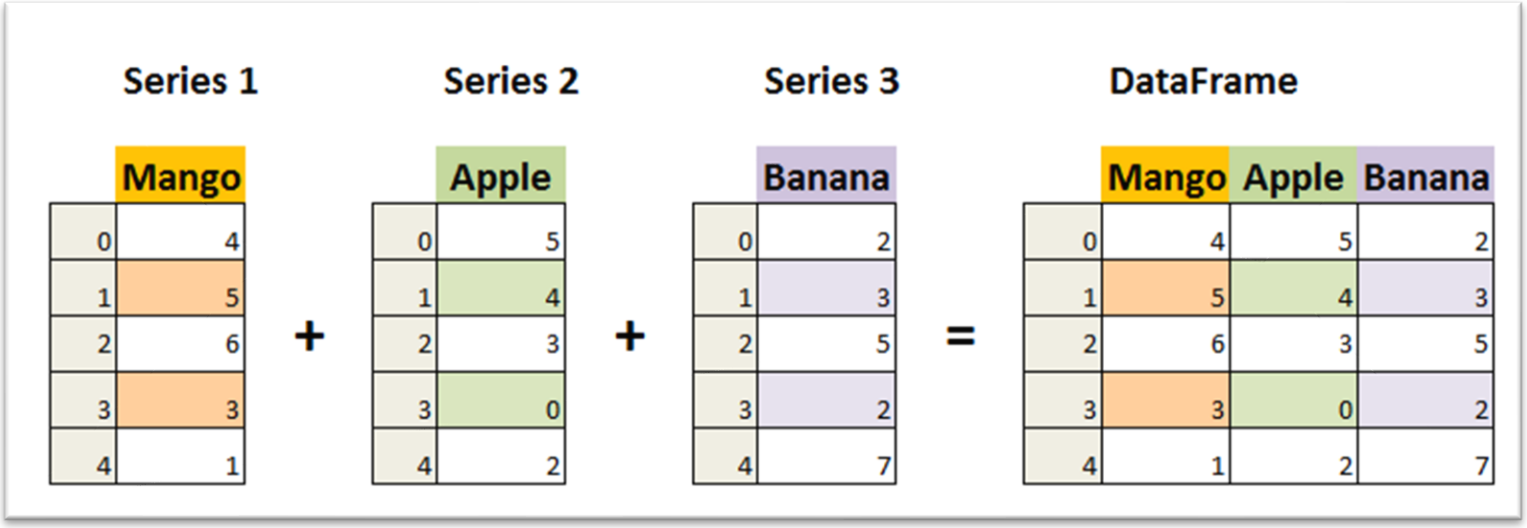
- Series có thể có thuộc tính name, thuộc tính name sẽ trở thành tên cột của Series trong DataFrame.

```
In [7]: s.name="something"
        s.name

Out[7]: 'something'
```

Dataframe là gì?

- Dataframe là cấu trúc dữ liệu dán nhãn 2 chiều với các cột có thể có kiểu dữ liệu khác nhau.
- Dataframe là được ghép lại từ nhiều Series với nhau.
- Mỗi cột của Dataframe sẽ có kiểu dữ liệu dtypes tương ứng với Series tạo nên nó.



Tạo dataframe từ dict của Series

- Tạo độ index (tên hàng) của DataFrame sẽ là giao của các index của các Series thành phần.

```
In [9]: dict={
            "Mango":pd.Series([4,5,6,3,1], index=[0,1,2,3,4]),
            "Apple":pd.Series([5,4,3,0,2],index=[0,1,2,3,4]),
            "Banana":pd.Series([2,3,5,2,7], index=[0,1,2,3,5])
        }
        df=pd.DataFrame(dict)
        df
```

Out[9]:

	Mango	Apple	Banana
0	4.0	5.0	2.0
1	5.0	4.0	3.0
2	6.0	3.0	5.0
3	3.0	0.0	2.0
4	1.0	2.0	NaN
5	NaN	NaN	7.0

Tạo DataFrame từ dict của Series

- Người dùng cũng có thể chỉ định các index cho DataFrame muốn tạo, những index không được liệt kê sẽ không xuất hiện trong DataFrame.

```
In [10]: dict={
             "Mango":pd.Series([4,5,6,3,1], index=[0,1,2,3,4]),
             "Apple":pd.Series([5,4,3,0,2],index=[0,1,2,3,4]),
             "Banana":pd.Series([2,3,5,2,7], index=[0,1,2,3,5])
         }
         df=pd.DataFrame(dict, index=[2,3,5])
         df
```

Out[10]:

	Mango	Apple	Banana
2	6.0	3.0	5
3	3.0	0.0	2
5	NaN	NaN	7

Tạo DataFrame từ dict của Series

- Người dùng cũng có thể chỉ định những cột nào được xuất hiện trong dataframe.

```
In [13]: dict={
             "Mango":pd.Series([4,5,6,3,1], index=[0,1,2,3,4]),
             "Apple":pd.Series([5,4,3,0,2],index=[0,1,2,3,4]),
             "Banana":pd.Series([2,3,5,2,7], index=[0,1,2,3,5])
         }
         df=pd.DataFrame(dict, index=[2,3,5], columns=["Mango","Apple"])
         df
```

Out[13]:

	Mango	Apple
2	6.0	3.0
3	3.0	0.0
5	NaN	NaN

Tạo dataframe từ dict của ndarray/lists

- Điều kiện: các mảng ndarray phải có độ dài giống nhau, nếu index được thêm thì độ dài của index cũng phải bằng độ dài các mảng. Nếu không them index, thì mặc định sẽ là range(n).

```
In [15]: d={"Mango":[2,3,4,1],"Apple":[3,4,5,6]}
         df=pd.DataFrame(d)
         df
```

Out[15]:

	Mango	Apple
0	2	3
1	3	4
2	4	5
3	1	6

```
In [17]: df=pd.DataFrame(d,index=["a","b","c","d"])
         df
```

Out[17]:

	Mango	Apple
a	2	3
b	3	4
c	4	5
d	1	6

Tạo dataframe từ tệp có cấu trúc dạng bảng

- Dataframe cũng có thể được tạo ra từ các tệp dữ liệu có cấu trúc dạng bảng.

```
In [2]: df=pd.read_csv("COVID-19 Dataset.csv")
        df
```

Out[2]:

	Continent	Country	Last_Updated_Date	Total_Cases	Total_Deaths	Total_Cases_Per_Million	Total_Deaths_Per_Million	Total_tests	Positive_rate	Total_V
0	Asia	Afghanistan	11-10-2021	155540.0	7228.0	3904.565	181.447	NaN	NaN	
1	Europe	Albania	11-10-2021	175163.0	2777.0	60970.074	966.608	NaN	NaN	
2	Africa	Algeria	11-10-2021	204695.0	5855.0	4587.864	131.229	NaN	NaN	
3	Europe	Andorra	11-10-2021	15307.0	130.0	197882.462	1680.585	215733.0	0.054	
4	Africa	Angola	11-10-2021	61580.0	1629.0	1814.720	48.006	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
211	Asia	Vietnam	11-10-2021	843281.0	20670.0	8590.110	210.556	24871501.0	0.055	
212	Oceania	Wallis and Futuna	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
213	Asia	Yemen	11-10-2021	9402.0	1782.0	308.357	58.444	NaN	NaN	

Truy suất trong DataFrame

Cách thức	Kí hiệu	Kết quả trả về
Chọn ô	df.at[row/columns]	Value
Chọn ô bằng index	df.iat[irow/icolumns]	Value
Chọn cột	df[col]	Series
Chọn hàng bằng nhãn (location)	df.loc[label]	Series
Chọn hàng bằng vị trí số (integer location)	df.iloc[loc]	Series
Cắt (slice) các hàng	df[5:10]	DataFrame
Chọn các hàng bằng Boolean vector	df[bool_vec]	DataFrame

Ví dụ: Chọn ô trong DataFrame

- Chọn ô trong DataFrame bằng cách gọi tọa độ hàng cột với thuộc tính at.

```
In [2]: df=pd.read_csv("COVID-19 Dataset.csv")
        df
```

Out[2]:

	Continent	Country	Last_Updated_Date	Total_Cases	Total_Deaths	Total_Cases_Per_Million	Total_Deaths_Per_Million	Total_tests	Positive_rate	Total_V
0	Asia	Afghanistan	11-10-2021	155540.0	7228.0	3904.565	181.447	NaN	NaN	
1	Europe	Albania	11-10-2021	175163.0	2777.0	60970.074	966.608	NaN	NaN	
2	Africa	Algeria	11-10-2021	204695.0	5855.0	4587.864	131.229	NaN	NaN	
3	Europe	Andorra	11-10-2021	15307.0	130.0	197882.462	1680.585	215733.0	0.054	
4	Africa	Angola	11-10-2021	61580.0	1629.0	1814.720	48.006	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
211	Asia	Vietnam	11-10-2021	843281.0	20670.0	8590.110	210.556	24871501.0	0.055	
212	Oceania	Wallis and Futuna	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
213	Asia	Yemen	11-10-2021	9402.0	1782.0	308.357	58.444	NaN	NaN	
214	Africa	Zambia	11-10-2021	209353.0	3654.0	11064.785	193.122	2509600.0	0.005	
215	Africa	Zimbabwe	11-10-2021	131875.0	4643.0	8737.974	307.643	1288436.0	0.038	

216 rows × 18 columns

```
In [29]: df.at[0,"Country"]
Out[29]: 'Afghanistan'
```

Truy xuất trong DataFrame

- Ví dụ: Chọn hàng “b” trong DataFrame.

```
In [17]: df=pd.DataFrame(d,index=["a","b","c","d"])
         df
```

Out[17]:

	Mango	Apple
a	2	3
b	3	4
c	4	5
d	1	6

```
In [18]: df.loc["b"]
```

```
Out[18]: Mango    3
         Apple    4
         Name: b, dtype: int64
```

Các phép toán với DataFrame

- DataFrame có thể dùng để tính toán như các ma trận tương tự như mảng ndarray.

```
In [17]: df=pd.DataFrame(d,index=["a","b","c","d"])
         df
```

Out[17]:

	Mango	Apple
a	2	3
b	3	4
c	4	5
d	1	6

```
In [19]: df*2
```

Out[19]:

	Mango	Apple
a	4	6
b	6	8
c	8	10
d	2	12

4. Xử lý dữ liệu khuyết thiếu

- Trong cuộc sống, trong một tập dữ liệu có thể có những dữ liệu bị mất hoặc chưa được cập nhật như những học sinh chưa có điểm,…
- Pandas và NumPy sử dụng 2 giá trị đặc biệt để đại diện giá trị khuyết thiếu:
  - NaN (Not a Number): Là giá trị dấu phẩy động đặc biệt theo chuẩn IEEE đại diện cho dữ liệu không phải là số.
  - None: là đối tượng Python được dùng để đại diện cho đối tượng bị mất.

None

- None là một đối tượng Python singleton (giá trị chỉ có một thể hiện duy nhất trên toàn bộ chương trình) đại diện cho dữ liệu khuyết thiếu trong mã Python.
- Vì None là một đối tượng Python nên None chỉ được sử dụng trong NumPy, Pandas khi mảng ndarray, Series có dtype=object.

```
In [3]: s=pd.Series([0,None,2,3], dtype=object)
        s

Out[3]: 0       0
        1    None
        2       2
        3       3
        dtype: object
```

Phép tính với giá trị None

- Tất cả các phép toán đối với None sẽ sinh ra lỗi.

```
In [3]: s=pd.Series([0,None,2,3], dtype=object)
        s

Out[3]: 0       0
        1    None
        2       2
        3       3
        dtype: object

In [4]: sum(s)

        ---------------------------------------------------------------------------
        TypeError                                 Traceback (most recent call last)
        ~\AppData\Local\Temp/ipykernel_25764/402837985.py in <module>
        ----> 1 sum(s)

        TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

NaN (Not a Number)

- NaN: là một giá trị dấu phẩy động đặc biệt theo chuẩn IEEE đại diện cho số bị khuyết thiếu do vậy nó sẽ chỉ được sử dụng trong mảng ndarray/Series có kiểu dữ liệu là dấu phẩy động (float) hoặc object.

```
In [3]: s=pd.Series([0,None,2,3], dtype=object)
        s

Out[3]: 0       0
        1    None
        2       2
        3       3
        dtype: object
```

Phép tính với giá trị NaN

- Tất cả các phép tính với NaN đều trả về NaN.

```
In [3]: s=pd.Series([0,None,2,3], dtype=object)
        s

Out[3]: 0       0
        1    None
        2       2
        3       3
        dtype: object

In [10]: sum(s)

Out[10]: nan

In [11]: 1+np.nan #Cộng 1 số với NaN

Out[11]: nan
```

NaN và None trong Pandas

- Nếu xuất hiện cả NaN và None trong cùng một ndarray/ Series, Pandas và NumPy sẽ tự động ép kiểu giữa hai phần tử sao cho phù hợp.

```
In [16]: s=pd.Series([0,None,np.nan,3])
         s

Out[16]: 0    0.0
         1    NaN
         2    NaN
         3    3.0
         dtype: float64
```

Dữ liệu khuyết thiếu với các kiểu không có giá trị đặc biệt

- Đối với các kiểu dữ liệu không có giá trị đặc biệt cho phần tử khuyết thiếu như int, string, bool,… , Pandas và NumPy sẽ tự động ép kiểu dữ liệu cho ndarray/ Series sao cho phù hợp.

```
In [20]: s=pd.Series([None,"1","2","3"], dtype=str)
         s

Out[20]: 0    None
         1       1
         2       2
         3       3
         dtype: object
```

Các kiểu dữ liệu Pandas tự động chuyển đổi khi chứa dữ liệu khuyết thiếu

Kiểu lớp	Chuyển đổi khi có khuyết thiếu	Giá trị đại diện
floating	Không đổi	np.nan
object	Không đổi	None hoặc np.nan
integer	Ép kiểu sang float64	np.nan
boolean	Ép kiểu sang object	None hoặc np.nan

Tương tác với các giá trị khuyết thiếu

- Pandas cung cấp những hàm sau nhằm mục đích tương tác với dữ liệu khuyết thiếu.
  - isnull(): Nhận vào là một đối tượng giống như mảng và trả về đối tượng có cấu trúc dữ liệu tương tự với các giá trị Boolean xác định có phải là dữ liệu khuyết thiếu hay không.
  - notnull(): Giống như isnull() nhưng xác định giá trị không phải khuyết thiếu hay không.
  - dropna(): Trả về phiên bản được loại bỏ dữ liệu khuyết thiếu.
  - fillna(): Trả về giá trị bản sao của dữ liệu với giá trị khuyết thiếu đã được thay thế hoặc bổ sung.

Isnull()

- Ví dụ: Với Series isnull() trả về một series là các giá trị Boolean đại diện cho các giá trị tương ứng trong Series ban đầu có phải là giá trị khuyết thiếu hay không.

```
In [20]: s=pd.Series([None,"1","2","3"])
         s

Out[20]: 0    None
         1       1
         2       2
         3       3
         dtype: object

In [21]: s.isnull()

Out[21]: 0     True
         1    False
         2    False
         3    False
         dtype: bool
```

Notnull()

- Ví dụ: Với Series isnull() trả về một series là các giá trị Boolean đại diện cho các giá trị tương ứng trong Series ban đầu không phải là giá trị khuyết thiếu.

```
In [20]: s=pd.Series([None,"1","2","3"])
         s

Out[20]: 0    None
         1       1
         2       2
         3       3
         dtype: object

In [25]: s.notnull()

Out[25]: 0    False
         1     True
         2     True
         3     True
         dtype: bool
```

Isnull() và notnull() trong việc truy suất trong Pandas

- Ta có thể dùng isnull() và notnull() như các vector Boolean để thu được Series hoặc DataFrame

```
In [20]: s=pd.Series([None,"1","2","3"])
         s

Out[20]: 0    None
         1       1
         2       2
         3       3
         dtype: object

In [26]: s[s.notnull()]

Out[26]: 1    1
         2    2
         3    3
         dtype: object

In [27]: s[s.isnull()]

Out[27]: 0    None
         dtype: object
```

dropna()

- dropna(): trả về Series hoặc DataFrame với các giá trị khuyết thiếu bị loại bỏ. Đối với DataFrame, dropna() sẽ mặc định loại bỏ toàn bộ hàng hoặc cột có giá trị khuyết thiếu.

```
df=pd.DataFrame(dict)
df
```

Out[28]:

	Mango	Apple	Banana
0	4.0	5.0	2.0
1	5.0	4.0	3.0
2	6.0	3.0	5.0
3	3.0	0.0	2.0
4	1.0	2.0	NaN
5	NaN	NaN	7.0

In [31]:
```
df.dropna()
```

Out[31]:

	Mango	Apple	Banana
0	4.0	5.0	2.0
1	5.0	4.0	3.0
2	6.0	3.0	5.0
3	3.0	0.0	2.0

Dropna()

- Có thể điều chỉnh các tham số axis và how của dropna() để điều chỉnh hành vi của dropna()
  - axis: ***{0 or 'index', 1 or 'columns'}*** mặc định là 0 quyết định dropna() sẽ loại bỏ hàng hay cột khi có giá trị khuyết thiếu.
  - how: ***{'any', 'all'}*** mặc định là 'any' quyết định dropna() sẽ loại bỏ khi có ít nhất 1 hay tất cả các giá trị hàng hoặc cột là giá trị khuyết thiếu.

Out[37]:

	Mango	Apple	Banana	Wax Apple
0	4.0	5.0	2.0	4
1	5.0	4.0	3.0	2
2	6.0	3.0	5.0	3
3	3.0	0.0	2.0	4
4	1.0	2.0	NaN	5
5	NaN	NaN	1.0	1

In [38]:
```
df.dropna(axis="columns")
```

Out[38]:

	Wax Apple
0	4
1	2
2	3
3	4
4	5
5	1

Fillna()

- Hàm fillna() trả về một mảng mới dùng để lấp đầy những giá trị khuyết thiếu bằng những giá trị do người dùng quy định.

Out[37]:

	Mango	Apple	Banana	Wax Apple
0	4.0	5.0	2.0	4
1	5.0	4.0	3.0	2
2	6.0	3.0	5.0	3
3	3.0	0.0	2.0	4
4	1.0	2.0	NaN	5
5	NaN	NaN	1.0	1

In [40]:
```
df.fillna(value=0) #Thay các giá trị khuyết thiếu bằng 0
```

Out[40]:

	Mango	Apple	Banana	Wax Apple
0	4.0	5.0	2.0	4
1	5.0	4.0	3.0	2
2	6.0	3.0	5.0	3
3	3.0	0.0	2.0	4
4	1.0	2.0	0.0	5
5	0.0	0.0	1.0	1

Tham số inplace

- Với dropna() và fillna() mặc định sẽ trả về một Series/ DataFrame mới với các giá trị được lọc. Để thay đổi trực tiếp trên Series/ DataFrame ta đặt inplace=True, các hàm sẽ trả về None và việc lọc sẽ diễn ra trực tiếp trên hàm ban đầu.

Out[37]:

	Mango	Apple	Banana	Wax Apple
0	4.0	5.0	2.0	4
1	5.0	4.0	3.0	2
2	6.0	3.0	5.0	3
3	3.0	0.0	2.0	4
4	1.0	2.0	NaN	5
5	NaN	NaN	1.0	1

In [42]:
```
df.fillna(value=0, inplace=True) #Thay các giá trị khuyết thiếu bằng 0
df                               #trực tiếp trên df
```

Out[42]:

	Mango	Apple	Banana	Wax Apple
0	4.0	5.0	2.0	4
1	5.0	4.0	3.0	2
2	6.0	3.0	5.0	3
3	3.0	0.0	2.0	4
4	1.0	2.0	0.0	5
5	0.0	0.0	1.0	1

5. Sắp xếp dữ liệu trong DataFrame

- Pandas cung cấp công cụ thực hiện sắp xếp dữ liệu dạng bảng qua DataFrame với hàm sort_values().

pandas.DataFrame.sort_values

```
DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort',
na_position='last', ignore_index=False, key=None)
```
[source]

Sort by the values along either axis.

Parameters: **by** : ***str or list of str***

Name or list of names to sort by.

- if *axis* is 0 or *'index'* then *by* may contain index levels and/or column labels.
- if *axis* is 1 or *'columns'* then *by* may contain column levels and/or index labels.

axis : ***{0 or 'index', 1 or 'columns'}, default 0***

Axis to be sorted.

ascending : ***bool or list of bool, default True***

Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.

inplace : ***bool, default False***

If True, perform operation in-place.

kind : ***{'quicksort', 'mergesort', 'heapsort', 'stable'}, default 'quicksort'***

Choice of sorting algorithm. See also `numpy.sort()` for more information. *mergesort* and *stable* are the only stable algorithms. For DataFrames, this option is only applied when sorting on a single column or label.

Ví dụ: Sắp xếp dữ liệu với tệp COVID-19 Dataset.csv

- Thực hiện thao tác sắp xếp với tệp dữ liệu về Covid-19 của mỗi nước.
  - Bước 1: Tải tệp COVID-19 Dataset.csv tại đây
  - Bước 2: Tạo DataFrame từ tệp như sau:

```
In [2]: df=pd.read_csv("COVID-19 Dataset.csv")
        df
```

Out[2]:

	Continent	Country	Last_Updated_Date	Total_Cases	Total_Deaths	Total_Cases_Per_Million	Total_Deaths_Per_Million	Total_tests	Positive_rate	Total_V
0	Asia	Afghanistan	11-10-2021	155540.0	7228.0	3904.565	181.447	NaN	NaN	
1	Europe	Albania	11-10-2021	175163.0	2777.0	60970.074	966.608	NaN	NaN	
2	Africa	Algeria	11-10-2021	204695.0	5855.0	4587.864	131.229	NaN	NaN	
3	Europe	Andorra	11-10-2021	15307.0	130.0	197882.462	1680.585	215733.0	0.054	
4	Africa	Angola	11-10-2021	61580.0	1629.0	1814.720	48.006	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
211	Asia	Vietnam	11-10-2021	843281.0	20670.0	8590.110	210.556	24871501.0	0.055	
212	Oceania	Wallis and Futuna	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
213	Asia	Yemen	11-10-2021	9402.0	1782.0	308.357	58.444	NaN	NaN	
214	Africa	Zambia	11-10-2021	209353.0	3654.0	11064.785	193.122	2509600.0	0.005	
215	Africa	Zimbabwe	11-10-2021	131875.0	4643.0	8737.974	307.643	1288436.0	0.038	

216 rows × 18 columns

Ví dụ: Sắp xếp dữ liệu với tệp COVID-19 Dataset.csv

- Sắp xếp dữ liệu với bảng theo cột bằng cách đưa một list các cột cần sắp xếp vào tham số by.
  - VD: Sắp xếp số bảng theo tổng số ca nhiễm Covid-19 theo thứ tự từ thấp đến cao.

```
In [9]: df.sort_values(by=["Total_Cases"])
```

Out[9]:

	Continent	Country	Last_Updated_Date	Total_Cases	Total_Deaths	Total_Cases_Per_Million	Total_Deaths_Per_Million	Total_tests	Positive_rate	Total_Va
127	Oceania	Micronesia (country)	11-10-2021	1.0	NaN	8.602	NaN	NaN	NaN	
103	Oceania	Kiribati	11-10-2021	2.0	NaN	16.476	NaN	NaN	NaN	
166	Oceania	Samoa	11-10-2021	3.0	NaN	14.989	NaN	NaN	NaN	
208	Oceania	Vanuatu	11-10-2021	4.0	1.0	12.720	3.18	NaN	NaN	
123	Oceania	Marshall Islands	11-10-2021	4.0	NaN	67.094	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
139	Oceania	New Caledonia	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
175	North America	Sint Maarten (Dutch part)	08-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
196	Oceania	Tonga	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
200	North America	Turks and Caicos Islands	08-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
212	Oceania	Wallis and Futuna	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	

216 rows × 18 columns

Ví dụ: Sắp xếp dữ liệu với tệp COVID-19 Dataset.csv

- Có thể sắp xếp theo nhiều cột.

```
In [11]: df.sort_values(by=["Total_Cases","Total_Deaths"])
```

Out[11]:

	Continent	Country	Last_Updated_Date	Total_Cases	Total_Deaths	Total_Cases_Per_Million	Total_Deaths_Per_Million	Total_tests	Positive_rate	Total_
127	Oceania	Micronesia (country)	11-10-2021	1.0	NaN	8.602	NaN	NaN	NaN	
103	Oceania	Kiribati	11-10-2021	2.0	NaN	16.476	NaN	NaN	NaN	
166	Oceania	Samoa	11-10-2021	3.0	NaN	14.989	NaN	NaN	NaN	
208	Oceania	Vanuatu	11-10-2021	4.0	1.0	12.720	3.18	NaN	NaN	
123	Oceania	Marshall Islands	11-10-2021	4.0	NaN	67.094	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
139	Oceania	New Caledonia	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
175	North America	Sint Maarten (Dutch part)	08-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
196	Oceania	Tonga	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
200	North America	Turks and Caicos Islands	08-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
		Wallis and								

Ví dụ: Sắp xếp dữ liệu với tệp COVID-19 Dataset.csv

- Việc sắp xếp mặc định sẽ là tăng dần, có thể thay đổi bằng việc điều chỉnh tham số ascending.
  - VD: Sắp xếp bảng theo số ca nhiễm với chiều giảm dần.

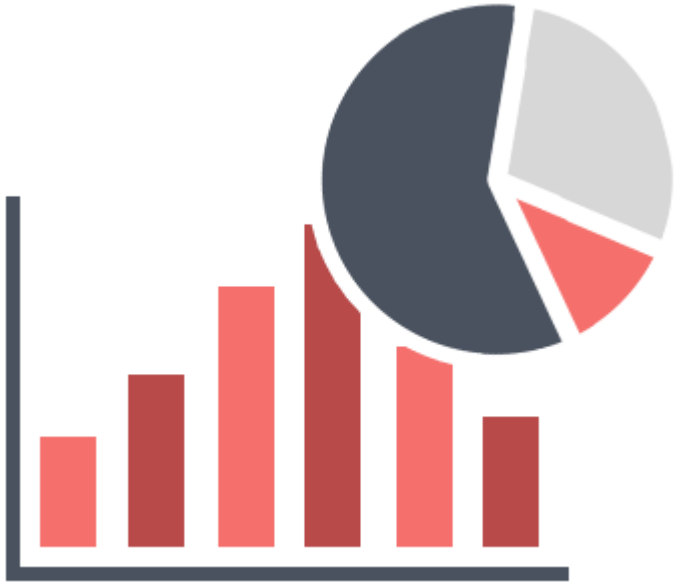
```
In [12]: df.sort_values(by=["Total_Cases"],ascending=False)
```

Out[12]:

	Continent	Country	Last_Updated_Date	Total_Cases	Total_Deaths	Total_Cases_Per_Million	Total_Deaths_Per_Million	Total_tests	Positive_rate	Total_Va
144	NaN	North America	11-10-2021	53474030.0	1084066.0	89634.106	1817.130	NaN	NaN	6
205	North America	United States	11-10-2021	44455949.0	714055.0	133535.404	2144.856	592381867.0	0.077	4
89	Asia	India	11-10-2021	33985920.0	450963.0	24390.483	323.640	583631490.0	0.014	9
26	South America	Brazil	11-10-2021	21582738.0	601213.0	100857.007	2809.493	NaN	NaN	2
204	Europe	United Kingdom	11-10-2021	8232327.0	138167.0	120696.017	2025.698	283376305.0	0.040	
...	...	...	...	...	...	...	...	...	...	...
139	Oceania	New Caledonia	05-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	
175	North America	Sint Maarten (Dutch part)	08-10-2021	NaN	NaN	NaN	NaN	NaN	NaN	

6. Thống kê dữ liệu

- Thống kê dữ liệu là một nhu cầu thường xuyên nhất là đối với dữ liệu dạng bảng.
- Pandas cung cấp nhưng công cụ giúp cho việc thống kê trong DataFrame trở nên khả thi.



Thống kê trên một hoặc nhiều cột

- Ta có thể còn thể thực hiện thống kê trên trên một hoặc nhiều cột bằng cách gọi các hàm thống kê trên các cột.
  - VD: Tính tổng số ca nhiễm Covid-19 trên toàn cầu.

```
In [22]: df[["Total_Cases"]].sum()

Out[22]: Total_Cases    291955371.0
         dtype: float64
```

Thống kê trên một hoặc nhiều cột

- VD2: Thống kê tổng số ca nhiễm Covid-19 và tổng số người chết vì Covid-19 trên toàn cầu.

```
In [23]: df[["Total_Cases","Total_Deaths"]].sum()

Out[23]: Total_Cases     291955371.0
         Total_Deaths      5943861.0
         dtype: float64
```

Các hàm thống kê trong DataFrame

Sr.No.	Function	Description
1	count()	Number of non-null observations
2	sum()	Sum of values
3	mean()	Mean of Values
4	median()	Median of Values
5	mode()	Mode of values
6	std()	Standard Deviation of the Values
7	min()	Minimum Value
8	max()	Maximum Value
9	abs()	Absolute Value
10	prod()	Product of Values
11	cumsum()	Cumulative Sum
12	cumprod()	Cumulative Product

Group By

- Hàm groupby() gom nhóm để thống kê bằng chiến thuật chia – áp dụng – kết hợp (split – apply - combine):
  - Split: Chia dữ liệu thành các nhóm dựa trên các điều kiện nhất định.
  - Apply: Sử dụng các hàm trên các nhóm một cách độc lập.
  - Combining: Trả về kết quả dưới dạng một cấu trúc dữ liệu nào đó.
- Hàm groupby() hoạt động giống với groupby() trong ngôn ngữ SQL.

```
SELECT Column1, Column2, mean(Column3), sum(Column4)
FROM SomeTable
GROUP BY Column1, Column2
```

Ví dụ: Thống kê số ca nhiễm theo từng châu lục

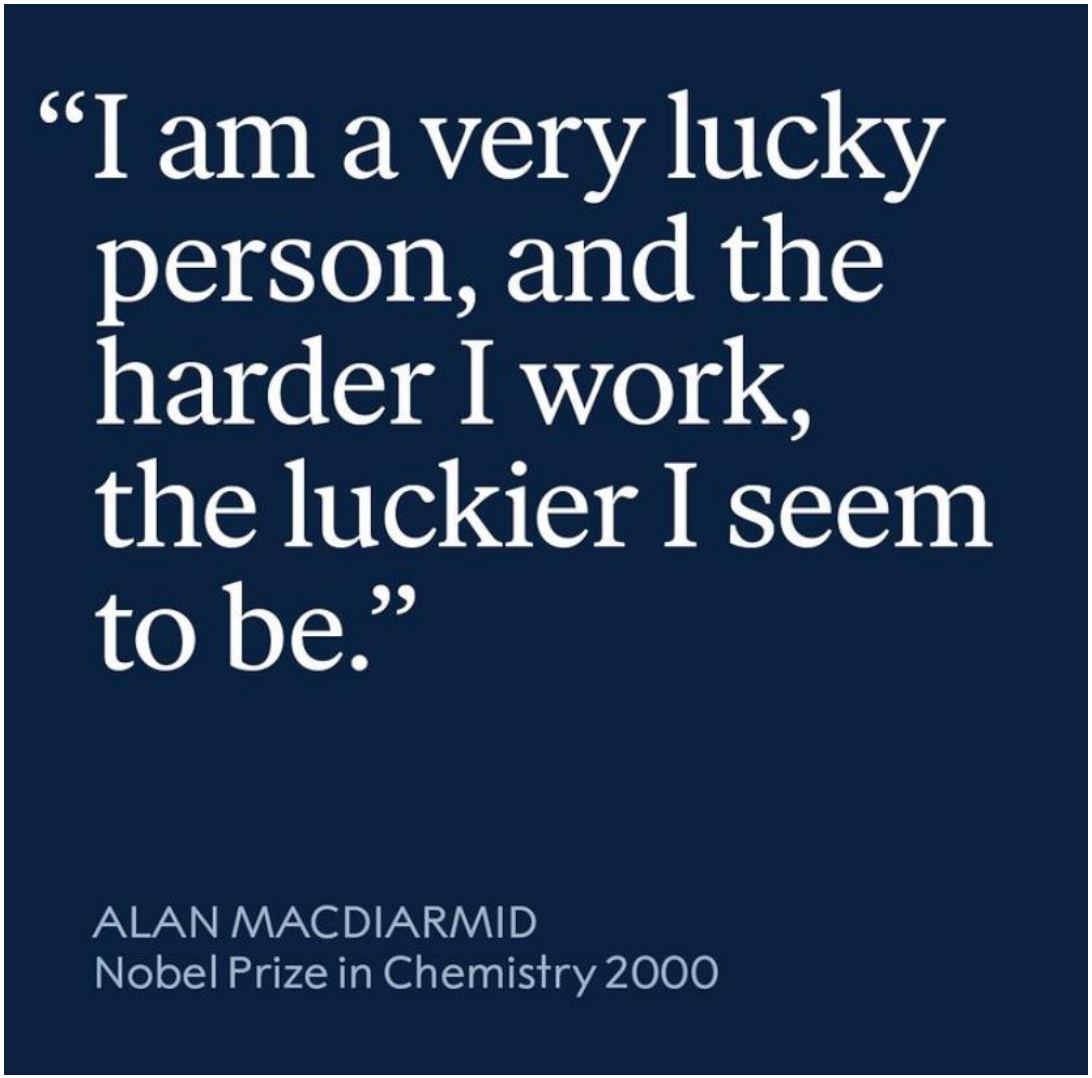
- Thống kê số ca nhiễm và số ca tử vong vì Covid-19 theo các châu lục.

```
In [24]: df[["Continent","Total_Cases","Total_Deaths"]].groupby("Continent").sum()
```

Out[24]:

Continent	Total_Cases	Total_Deaths
Africa	8395217.0	214140.0
Asia	77231697.0	1144914.0
Europe	60959340.0	1250900.0
North America	53474030.0	1084066.0
Oceania	210720.0	2388.0
South America	37999617.0	1160999.0

Quote


“I am a very lucky person, and the harder I work, the luckier I seem to be.”
ALAN MACDIARMID
Nobel Prize in Chemistry 2000