Dataframe

- Tạo Dataframe từ một list

```
import pandas as pd
traicay=["Cóc","Ổi","Cam","Mít","Táo"]
df=pd.DataFrame(traicay)
df
```

	0
0	Cóc
1	Ổi
2	Cam
3	Mít
4	Táo

Dataframe

- Tạo Dataframe từ một list

```
traicay=[["Cóc",1000],["Ổi",900],
["Cam",1200],["Mít",950],["Táo",1300]]
df=pd.DataFrame(traicay)
df
```

	0	1
0	Cóc	1000
1	Ổi	900
2	Cam	1200
3	Mít	950
4	Táo	1300

Dataframe

- Tạo Dataframe từ một list các Từ điển

```
data=[{"a":1,"b":2},
{"b":3,"c":4}]
df=pd.DataFrame(data)
df
```

	a	b	c
0	1.0	2	NaN
1	NaN	3	4.0

Dataframe

- Tạo Dataframe từ một Từ điển các list

```
data={
    "a":[1,2,3,4,5],
    "b":[6,7,8,9,10],
    "c":[11,12,13,14,15]
}
df=pd.DataFrame(data)
df
```

	a	b	c
0	1	6	11
1	2	7	12
2	3	8	13
3	4	9	14
4	5	10	15

Dataframe

- Tạo Dataframe từ một Series

```
s=pd.Series([1,23,4,15,6])
df=pd.DataFrame(s)
df
```

	0
0	1
1	23
2	4
3	15
4	6

Dataframe

- Tạo Dataframe từ 2 Series

```
#Dân số:
data1={
    "Hà Nội":8054,
    "Quảng Ninh":1415,
    "Hồ Chí Minh":8993,
    "Đà Nẵng":1134,
    "Cần Thơ":1282
}

s1=pd.Series(data1)
s1
```

```
Hà Nội          8054
Quảng Ninh      1415
Hồ Chí Minh     8993
Đà Nẵng         1134
Cần Thơ         1282
dtype: int64
```

Dataframe

- Tạo Dataframe từ 2 Series

```
#Diện tích
data2={
    "Hà Nội":3359,
    "Quảng Ninh":6102,
    "Hồ Chí Minh":2061,
    "Đà Nẵng":1285,
    "Cần Thơ":1439
}

s2=pd.Series(data2)
s2
```

```
Hà Nội          3359
Quảng Ninh      6102
Hồ Chí Minh     2061
Đà Nẵng         1285
Cần Thơ         1439
dtype: int64
```

Dataframe

- Tạo Dataframe từ 2 Series

```
#Gộp 2 Series s1,s2 thành từ điển
data={
    "Dân số":s1,
    "Diện tích":s2
}
df=pd.DataFrame(data)
df
```

	Dân số	Diện tích
Hà Nội	8054	3359
Quảng Ninh	1415	6102
Hồ Chí Minh	8993	2061
Đà Nẵng	1134	1285
Cần Thơ	1282	1439

Dataframe

- Tạo Dataframe với chỉ mục tự cho

```
traicay=[["Cóc",1000],["Ổi",900],
["Cam",1200],["Mít",950],["Táo",1300]]
df=pd.DataFrame(traicay)
df
```

	0	1
0	Cóc	1000
1	Ổi	900
2	Cam	1200
3	Mít	950
4	Táo	1300

Dataframe

- Tạo Dataframe với chỉ mục tự cho

```
traicay=[["Cóc",1000],["Ổi",900],
["Cam",1200],["Mít",950],["Táo",1300]]
idx=[1,2,3,4,5]
cols=["Tên","Giá"]
df=pd.DataFrame(traicay, index=idx,
columns=cols)
df
```

	Tên	Giá
1	Cóc	1000
2	Ổi	900
3	Cam	1200
4	Mít	950
5	Táo	1300

Dataframe – Thao tác trên Dataframe

- Truy xuất dữ liệu:
  - Truy xuất các cột như Từ điển

	Tên	Giá
1	Cóc	1000
2	Ổi	900
3	Cam	1200
4	Mít	950
5	Táo	1300

```
df["Tên"]
```

```
1    Cóc
2     Ổi
3    Cam
4    Mít
5    Táo
Name: Tên, dtype: object
```

Dataframe – Thao tác trên Dataframe

- Truy xuất dữ liệu:
  - Truy xuất hàng: dùng values

	Tên	Giá
1	Cóc	1000
2	Ổi	900
3	Cam	1200
4	Mít	950
5	Táo	1300

```
df.values
```

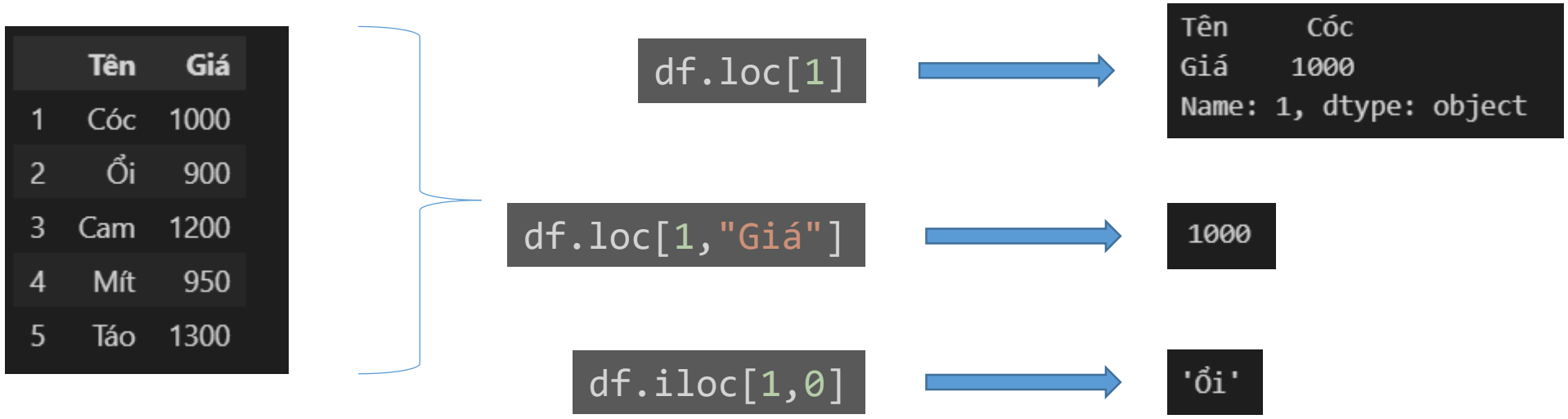
```
array([['Cóc', 1000],
       ['Ổi', 900],
       ['Cam', 1200],
       ['Mít', 950],
       ['Táo', 1300]], dtype=object)
```

```
df.values[0]
```

```
array(['Cóc', 1000], dtype=object)
```

Dataframe – Thao tác trên Dataframe

- Truy xuất dữ liệu:
  - Dùng loc, iloc



Dataframe – Thao tác trên Dataframe

- Truy xuất dữ liệu:
  - Đảo bảng

	Tên	Giá
1	Cóc	1000
2	Ổi	900
3	Cam	1200
4	Mít	950
5	Táo	1300

```
df.T
```

	1	2	3	4	5
Tên	Cóc	Ổi	Cam	Mít	Táo
Giá	1000	900	1200	950	1300

Dataframe – Thao tác trên Dataframe

- Truy xuất dữ liệu:
  - Tạo cột mới

	Tên	Giá
1	Cóc	1000
2	Ổi	900
3	Cam	1200
4	Mít	950
5	Táo	1300

```
df["Thuế"]=df["Giá"]*0.05
df
```

	Tên	Giá	Thuế
1	Cóc	1000	50.0
2	Ổi	900	45.0
3	Cam	1200	60.0
4	Mít	950	47.5
5	Táo	1300	65.0

Phương thức apply()

- Dùng để áp dụng một hàm trên Series, Dataframe

	Tên	Giá	Thuế
1	Cóc	1000	50.0
2	Ổi	900	45.0
3	Cam	1200	60.0
4	Mít	950	47.5
5	Táo	1300	65.0

```
def them(x):
    x+=x*0.1
    return x
df["Giá"].apply(them)
```

```
1    1100.0
2     990.0
3    1320.0
4    1045.0
5    1430.0
Name: Giá, dtype: float64
```

Các phép toán

- **Operators : +, -, *, /, //, %, ****
- **Object Methods : add(), sub(), mul(), div(), mod(), pow()**
- **Nguyên tắc thực hiện phép toán:**
  - **Các giá trị cùng chỉ số sẽ thực hiện với nhau. Nếu không sẽ trả về NaN**
  - **Nếu là phép toán với 1 số, thì thực hiện phép toán đó trên tất cả các giá trị**

Các phép toán

- Ví dụ:

```
s1=pd.Series([1,2,3],index=["a","b","c"])
s1
```

```
a    1
b    2
c    3
dtype: int64
```

```
s2=pd.Series([4,5,6],index=["b","c","d"])
s2
```

```
b    4
c    5
d    6
dtype: int64
```

```
s1+s2
```

```
s1.add(s2)
```

```
a    NaN
b    6.0
c    8.0
d    NaN
dtype: float64
```

Các phép toán

- Ví dụ:

```
s1=pd.Series([1,2,3],index=["a","b","c"])
s1
```

```
a    1
b    2
c    3
dtype: int64
```

```
s2=pd.Series([4,5,6],index=["b","c","d"])
s2
```

```
b    4
c    5
d    6
dtype: int64
```

```
s1.add(s2,fill_value=0)
```

```
a    1.0
b    6.0
c    8.0
d    6.0
dtype: float64
```

Một số Phương thức

- s.ndim: trả về số chiều của s
- s.values: trả về list các phần tử của s
- s.head(n): trả về n phần tử đầu của s
- s.tail(n): trả về n phần tử cuối của s
- s.empty: trả về True nếu s rỗng
- s.axes: trả về list các chỉ mục

Xử lý dữ liệu bị thiếu

- Vấn đề xảy ra khi thu thập dữ liệu trong thực tế:
  - Dữ liệu thu được ít khi nào sạch (clean) và đồng nhất.
  - Đa số trường hợp sẽ có một lượng data bị mất hoặc thiếu.
  - Pandas gọi những data bị thiếu này là null, NaN hoặc giá trị NA

Xử lý dữ liệu bị thiếu

- Có hai quy ước chính cho việc thể hiện dữ liệu trống
  - Dùng mask : ví dụ mảng boolean 2 chiều -> false thể hiện giá trị bị trống.
  - Dùng một giá trị gác (sentinel value) : ví dụ như đối với trường hợp số nguyên bị thiếu thì dùng -9999

Xử lý dữ liệu bị thiếu

- Hai cách này có những hạn chế nhất định.
  - Dùng mask : Tăng thời gian tính toán + Tốn bộ nhớ
  - Dùng một giá trị gác (sentinel value) : Giới hạn số lượng giá trị có thể sử dụng trong bảng
    + Các CPU thường không được tối ưu cho logic của các giá trị gác

Xử lý dữ liệu bị thiếu

- Hai cách này có những hạn chế nhất định.
  - Dùng mask : Tăng thời gian tính toán + Tốn bộ nhớ
  - Dùng một giá trị gác (sentinel value) : Giới hạn số lượng giá trị có thể sử dụng trong bảng
    + Các CPU thường không được tối ưu cho logic của các giá trị gác

Trên thực tế không có một cách lựa chọn nào hoàn toàn tối ưu (one-size-fit-all answer), tuỳ vào ngôn ngữ lập trình mà lựa chọn quy ước nào hợp lý

Xử lý dữ liệu bị thiếu

- Pandas lựa chọn giá trị gác (sentinel value) để biểu diễn giá trị trống
  - None [Type : object]
  - NaN (Not a number) [Type : float64]
- Tất cả các phép tính toán chứa NaN đều sẽ trả về NaN
- Pandas có thể luân chuyển giữa NaN và None khi cảm thấy phù hợp

Xử lý dữ liệu bị thiếu

- Phát hiện dữ liệu rỗng

```
s=pd.Series([1,2,None,4,np.NaN,6])
s
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    6.0
dtype: float64
```

Xử lý dữ liệu bị thiếu

- Phát hiện dữ liệu rỗng

```
s=pd.Series([1,2,None,4,np.NaN,6])
s
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    6.0
dtype: float64
```

```
s.isnull()
```

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

Xử lý dữ liệu bị thiếu

- Phát hiện dữ liệu rỗng

```
s=pd.Series([1,2,None,4,np.NaN,6])
s
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    6.0
dtype: float64
```

```
s[s.isnull()]
```

```
2   NaN
4   NaN
dtype: float64
```

Xử lý dữ liệu bị thiếu

- Xóa dữ liệu rỗng

```
s=pd.Series([1,2,None,4,np.NaN,6])
s
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    6.0
dtype: float64
```

```
s.dropna()
```

```
0    1.0
1    2.0
3    4.0
5    6.0
dtype: float64
```

Xử lý dữ liệu bị thiếu

- Xóa dữ liệu rỗng

```
s=pd.Series([1,2,None,4,np.NaN,6])
s
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    6.0
dtype: float64
```

```
s.dropna()
```

```
0    1.0
1    2.0
3    4.0
5    6.0
dtype: float64
```

Lưu ý: Đối với DataFrame, dropna() sẽ xoá cả hàng (cột) nếu hàng (cột) đó chứa dữ liệu trống

Xử lý dữ liệu bị thiếu

- Thay thế dữ liệu rỗng

```
s=pd.Series([1,2,None,4,np.NaN,6])
s
```

```
0    1.0
1    2.0
2    NaN
3    4.0
4    NaN
5    6.0
dtype: float64
```

```
s.fillna(-1)
```

```
0    1.0
1    2.0
2   -1.0
3    4.0
4   -1.0
5    6.0
dtype: float64
```

Đọc file csv

- Sử dụng Phương thức read_csv("tên_file.csv")

```
import pandas as pd
data=pd.read_csv("casi.csv")
data
```

	idCS	HoTenCS
0	1	Cẩm Ly
1	2	Cẩm Vân
2	3	Khánh Ly
3	4	Quang Dũng
4	5	Hồng Nhung
5	6	Lệ Thu
6	7	Trịnh Công Sơn
7	8	Vĩnh Trinh
8	9	Hương Lan
9	10	Bích Phượng
10	11	Thu Hiền
11	12	Phi Nhung

Lưu file csv

- Sử dụng Phương thức to_csv(“tên_file.csv”)

```
data1.to_csv("data1.csv",index=False)
```

Demo