pandas Series

1 pandas Series

[1]:
```
# import những thư viện cần thiết
import numpy as np
import pandas as pd
```

1.1 Đối tượng `pandas.Series`

`pandas.Series` là mảng 1 chiều **có gắn nhãn**. Tức là, mỗi điểm dữ liệu sẽ có một tên riêng.

[2]:
```
# tạo Series từ list
test_data = [0, 0.25, 0.5, 0.75, 1.0] #dữ liệu
s = pd.Series(test_data)
print(s)
```

```
0    0.00
1    0.25
2    0.50
3    0.75
4    1.00
dtype: float64
```

Để thấy sự khác nhau giữa `pandas.Series` và `ndarray`, chúng ta tạo ra một `ndarray` tương ứng từ list `data`

[3]:
```
a = np.array(test_data)
print(a)
```

```
[0.   0.25 0.5  0.75 1.  ]
```

Như các bạn có thể thấy, trong `pandas.Series` ngoài các giá trị có trong list `data` thì nó có các thuộc tính khác:

- Thuộc tính giá trị [0, 1, 2, 3]: gọi là nhãn (label, hoặc index) của `pandas.Series` s.
- Thuộc tính là `dtype`: kiểu dữ liệu chung của các phần tử trong `pandas.Series` s.

Có thể lấy các thuộc tính này bằng các câu lệnh tương ứng sau:

[4]:
```
print("Dữ liệu: ", s.values)
print("Index: ", s.index)
print("Kiểu của dữ liệu: ", s.dtype)
```

```
Dữ liệu:  [0.   0.25 0.5  0.75 1.  ]
Index:  RangeIndex(start=0, stop=5, step=1)
Kiểu của dữ liệu:  float64
```

Nhưng nếu chỉ có như vậy thì cũng không có gì khác biệt so với `ndarray` 1 chiều?

Câu trả lời chính là bạn có thể thay đổi `label` của một Series theo cách như sau :

[5]:
```
test_data = [1, 2, 3, 4]
s_new = pd.Series(
    data = test_data,
    index = ['a', 'b', 'c', 'd'])
print(s_new)
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

Vậy là đã có index (nhãn) khác rồi.

[ ]:
```
# phương thức khởi tạo Series cơ bản
pd.Series(
    data,     # Có thể là array, list, dictionary hoặc giá trị vô hướng
    index,    # Index của Series
    dtype,    # Kiểu dữ liệu (tuỳ chọn)
    name      # Tên Series (tuỳ chọn)
)
```

Lưu ý:

- Số lượng phần tử của `data` phải bằng số lượng phần tử của `index`.
- Nếu không chỉ ra `index` thì sẽ tự động sinh ra `index` là số thứ tự.

[6]:
```
a1 = [i*i for i in range(5)]
# Tạo Series, nếu không có giá trị cho index thì nó sẽ tự động sinh ra
s1 = pd.Series(a1)
print(s1)
```

```
0     0
1     1
2     4
3     9
4    16
dtype: int64
```

[7]:
```
# Tạo Series từ dictionary
# Tạo dictionary
d = {
    "UK" : "Pound",
    "USA" : "US Dollar"
}
# Tạo Series, lúc này label sẽ là key của dictionary (nếu không chỉ ra index)
s2 = pd.Series(d)
print(s2)
```

```
UK          Pound
USA     US Dollar
dtype: object
```

[8]:
```
# Tạo Series từ 1 giá trị vô hướng
s3 = pd.Series('chihuahua', index = ['a', 'b', 'c'])
print(s3)
```

```
a    chihuahua
b    chihuahua
c    chihuahua
dtype: object
```

2 Các thao tác trên `pandas.Series`

2.1 Truy xuất phần tử từ `pandas.Series`

Chúng ta có thể truy xuất đến một (hoặc nhiều) phần tử của một `pandas.Series` bằng `index` của một (hoặc nhiều) phần tử đó. Ví dụ:

[9]:
```
# Tạo Series
s4 = pd.Series([1, 2, 3, 4, 5], index = ['a', 'b', 'c', 'd', 'e'])
print(s4)
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

[10]:
```
# Lấy phần tử đầu tiên
print(s4['a'])
```

```
1
```

[11]:
```
# Lấy các phần tử có index là 'a' và 'e'
print(s4[['a', 'e']])
```

```
a    1
e    5
dtype: int64
```

Ngoài ra, chúng ta còn có hai phương thức là `.head(n)` và `.tail(n)` để lấy ra `n` phần tử ở đầu và cuối của `pandas.Series`

[12]:
```
# Lấy ra 2 phần tử đầu
print('Hai phần tử đầu:')
print(s4.head(2))
# Lấy ra 2 phần tử cuối
print('Hai phần tử ở cuối:')
print(s4.tail(2))
```

```
Hai phần tử đầu:
a    1
b    2
dtype: int64
Hai phần tử ở cuối:
d    4
e    5
dtype: int64
```

Câu hỏi : Cho s là một `pandas.Series`, `s.head()` và `s.tail()` sẽ trả về cái gì?

2.2 Cập nhật phần tử `pandas.Series`

Để thay đổi giá trị của một (hoặc nhiều) phần tử, cần gọi phần tử bạn muốn rồi gán giá trị mới cho nó.

[13]:
```
# Tạo Series
s5 = pd.Series([1, 2, 3], index = ['a', 'b', 'c'])
print('Trước : ')
print(s5)
# Gán giá trị mới cho phần tử có label là 'b'
s5.loc['b'] = 100
print('Sau : ')
print(s5)
```

```
Trước :
a    1
b    2
c    3
dtype: int64
Sau :
a      1
b    100
c      3
dtype: int64
```

2.3 Thêm bớt phần tử vào `pandas.Series`

Để thêm **một** phần tử mới vào `pandas.Series`, thực hiện tương tự việc thay đổi giá trị, điểm khác là `index` chưa tồn tại trong `Series`.

[14]:
```
# Tạo Series
s6 = pd.Series([1, 2, 3, 4], index = ['a', 'b', 'c', 'd'])
print('Trước : ')
print(s6)
# Thêm một phần tử mới có giá trị là 100 và label là 'g'
s6.loc['g'] = 100
print('Sau : ')
print(s6)
```

```
Trước :
a    1
b    2
c    3
d    4
dtype: int64
Sau :
a      1
b      2
c      3
d      4
g    100
dtype: int64
```

Để xoá **một** phần tử có sẵn trong `pandas.Series`, dung phương thức `.drop()` và chỉ ra các `index` cần xóa:

[15]:
```
# Chúng ta sẽ dùng lại Series s6 ở trên
print('Trước khi xoá : ')
print(s6)
# Xoá phần tử có label là 'a'
s6 = s6.drop(['a'])
print("Sau khi xoá phần tử có label 'a' : ")
print(s6)
```

```
Trước khi xoá :
a      1
b      2
c      3
d      4
g    100
dtype: int64
Sau khi xoá phần tử có label 'a' :
b      2
c      3
d      4
```

```
g    100
dtype: int64
```

2.4 Truy xuất một số thông tin về một `pandas.Series` bất kỳ

Ngoài các thuộc tính `.values`, `.index` và `.dtype` như đề cập ở trên, một `pandas.Series` còn có nhiều thuộc tính khác, như:

[16]:
```
# Tạo Series
s6 = pd.Series(np.random.randint(5))
# Lấy kích thước của Series
print('Kích thước : ', s6.size)
# Kiểm tra Series có phải rỗng
print('Rỗng : ', s6.empty)
```

```
Kích thước :  1
Rỗng :  False
```

2.5 Một số phương thức thống kê

Chúng ta có các phương thức thống kê như sau:

- `.count()`: trả về số lượng các phần tử khác NaN (Not a Number, một giá trị đặc biệt của python).
- `.sum()`: trả về tổng các phần tử.
- `.mean()`: trả về trung bình các phần tử.
- `.median()`: trả về trung vị.
- `.mode()`: trả về mốt (phần tử xuất hiện nhiều lần nhất).
- `.std()`: trả về độ lệch chuẩn.
- `.min()`: trả về giá trị nhỏ nhất.
- `.max()`: trả về giá trị lớn nhất.
- `.cumsum()`: trả về tổng tích luỹ.
- `.describe()`: trả về thống kê mô tả.
- ...