[2-2. Series, DataFrame 다루기]

1. Series 주요 속성

- index : Series의 인덱스 데이터 반환
- values : Series의 데이터 반환

In [1]:
```
import pandas as pd
from pandas import Series, DataFrame
```

In [10]:
```
a = Series([1, 2, 3, 4, 2, 4], index=['a', 'b', 'c', 'd', 'e', 'f']
)
```

In [11]:
```
a.index #Series의 index 속성
```

Out[11]:
```
Index(['a', 'b', 'c', 'd', 'e', 'f'], dtype='object')
```

In [12]:
```
a.index[0]
```

Out[12]:
```
'a'
```

In [13]:
```
a.values #Series의 values 속성
```

Out[13]:
```
array([1, 2, 3, 4, 2, 4])
```

In [14]:
```
a.values[0]
```

Out[14]:
```
1
```

In [ ]:
```

```

2. Series 주요 메서드

- append : 2개 이상의 시리즈 연결
- describe : 요약통계 계산
- min/max : 최소값/최대값 반환
- mean : 평균 반환
- sort_values : 값을 정렬
- isin : 입력된 값이 시리즈에 있는지 확인
- unique : 데이터의 유니크한 값들을 배열로 반환
- to_frame : 시리즈를 데이터프레임으로 변환

In [21]:
```
a
```

Out[21]:
```
a    1
b    2
c    3
d    4
e    2
f    4
dtype: int64
```

In [22]:
```
b = Series([10, 20, 30, 10, 20], index = [1, 2, 3, 4, 5])
b
```

Out[22]:
```
1    10
2    20
3    30
4    10
5    20
dtype: int64
```

append

In [25]:
```
k = a.append(b)
k
```

Out[25]:
```
a     1
b     2
c     3
d     4
e     2
f     4
1    10
2    20
3    30
4    10
5    20
dtype: int64
```

describe

In [26]:
```
a.describe()
```

Out[26]:
```
count    6.000000
mean     2.666667
std      1.211060
min      1.000000
25%      2.000000
50%      2.500000
75%      3.750000
max      4.000000
dtype: float64
```

In [27]:
```
b.describe()
```

Out[27]:
```
count     5.0000
mean     18.0000
std       8.3666
min      10.0000
25%      10.0000
50%      20.0000
75%      20.0000
max      30.0000
dtype: float64
```

min / max / mean

In [28]:
```
#min
a.min() #시리즈 메서드
```

Out[28]: 1

In [29]:
```
min(a) #파이썬 내장함수
```

Out[29]: 1

In [30]:
```
#max
b.max() #시리즈 메서드
```

Out[30]: 30

In [31]:
```
max(b) #파이썬 내장함수
```

Out[31]: 30

In [32]:
```
#mean
a.mean() #시리즈 메서드
```

Out[32]: 2.6666666666666665

In [33]:
```
mean(a) #파이썬 내장함수 없음
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
/var/folders/01/c04y5bhn61l8kzg4jnpy_8dw0000gp/T/ipykernel_53507/2
528573478.py in <module>
----> 1 mean(a) #파이썬 내장함수

NameError: name 'mean' is not defined
```

In [34]:
```
sum(a) / len(a) #파이썬 내장함수
```

Out[34]: 2.6666666666666665

sort_values

In [38]:
```
a.sort_values()
```

Out[38]:
```
a    1
b    2
e    2
c    3
d    4
f    4
dtype: int64
```

In [39]:
```
a.sort_values(ascending=False)
```

Out[39]:
```
d    4
f    4
c    3
b    2
e    2
a    1
dtype: int64
```

isin

In [41]:
```
a.isin([3]) #입력값은 리스트로 넣어줘야 함
```

Out[41]:
```
a    False
b    False
c     True
d    False
e    False
f    False
dtype: bool
```

In [42]:
```
a.isin([2, 4])
```

Out[42]:
```
a    False
b     True
c    False
d     True
e     True
f     True
dtype: bool
```

unique

In [84]:
```
a.unique()
```

Out[84]:
```
array([1, 2, 3, 4])
```

In [85]:
```
b.unique()
```

Out[85]:
```
array([10, 20, 30])
```

to_frame

In [52]:
```
a.to_frame()
```

Out[52]:

	0
a	1
b	2
c	3
d	4
e	2
f	4

In [53]:
```
a.to_frame().transpose() #데이터프레임의 transpose 메서드 이용 행렬 변환
```

Out[53]:

	a	b	c	d	e	f
0	1	2	3	4	2	4

In [54]:
```
a.to_frame().T #데이터프레임의 T 속성 이용 행렬 변환
```

Out[54]:

	a	b	c	d	e	f
0	1	2	3	4	2	4

In [ ]:
```

```

3. DataFrame 주요 속성

- index : 인덱스(행 레이블) 반환
- columns : 컬럼(열 레이블) 반환
- values : 데이터를 반환

In [86]:
```
data = [[10, 100, 1000],
        [20, 200, 2000],
        [30, 300, 3000],
        [40, 400, 4000]]
df = DataFrame(data, columns=['data0', 'data1', 'data2'],
               index=['one', 'two', 'three', 'four'])
```

In [87]:
```
df.index
```

Out[87]:
```
Index(['one', 'two', 'three', 'four'], dtype='object')
```

In [88]:
```
df.columns
```

Out[88]:
```
Index(['data0', 'data1', 'data2'], dtype='object')
```

In [89]:
```
df.values
```

Out[89]:
```
array([[  10,  100, 1000],
       [  20,  200, 2000],
       [  30,  300, 3000],
       [  40,  400, 4000]])
```

In [ ]:
```

```

4. DataFrame 주요 메서드

- count : 각 칼럼 또는 로우의 데이터 개수를 반환
- describe : 각 칼럼에 대한 요약통계 계산
- min / max : 각 칼럼 또는 로우에 대한 최소/최대 값 계산
- sum : 각 칼럼 또는 로우에 대하여 합 계산
- mean : 각 칼럼 또는 로우에 대한 평균 계산
- cumsum : 각 칼럼 또는 로우에 대하여 누적 합 계산

In [90]:
```
df
```

Out[90]:

	data0	data1	data2
one	10	100	1000
two	20	200	2000
three	30	300	3000
four	40	400	4000

count

In [91]:
```
df.count()  #칼럼의 데이터 개수 반환
```

Out[91]:
```
data0    4
data1    4
data2    4
dtype: int64
```

In [92]:
```
df.count(axis=1)  #로우의 데이터 개수 반환
```

Out[92]:
```
one      3
two      3
three    3
four     3
dtype: int64
```

describe

In [93]:
```
df.describe()  #각 칼럼에 대한 요약통계 계산
```

Out[93]:

	data0	data1	data2
count	4.000000	4.000000	4.000000
mean	25.000000	250.000000	2500.000000
std	12.909944	129.099445	1290.994449
min	10.000000	100.000000	1000.000000
25%	17.500000	175.000000	1750.000000
50%	25.000000	250.000000	2500.000000
75%	32.500000	325.000000	3250.000000
max	40.000000	400.000000	4000.000000

min / max

In [94]:
```
df.min()  #칼럼에 대한 최소값 계산
```

Out[94]:
```
data0      10
data1     100
data2    1000
dtype: int64
```

In [95]:
```
df.min(axis=1)  #로우에 대한 최소값 계산
```

Out[95]:
```
one      10
two      20
three    30
four     40
dtype: int64
```

In [96]:
```
df.max()  #칼럼에 대한 최대값 계산
```

Out[96]:
```
data0      40
data1     400
data2    4000
dtype: int64
```

In [97]:
```
df.max(axis=1)  #로우에 대한 최대값 계산
```

Out[97]:
```
one      1000
two      2000
three    3000
four     4000
dtype: int64
```

sum

In [98]:
```
df.sum()  #칼럼에 대한 합 계산
```

Out[98]:
```
data0      100
data1     1000
data2    10000
dtype: int64
```

In [99]:
```
df.sum(axis=1)  #로우에 대한 합 계산
```

Out[99]:
```
one      1110
two      2220
three    3330
four     4440
dtype: int64
```

mean

In [100]:
```
df.mean()  #칼럼에 대한 평균 계산
```

Out[100]:
```
data0      25.0
data1     250.0
data2    2500.0
dtype: float64
```

In [101]:
```
df.mean(axis=1)  #로우에 대한 평균 계산
```

Out[101]:
```
one       370.0
two       740.0
three    1110.0
four     1480.0
dtype: float64
```

cumsum

In [102]:
```
df.cumsum()  #칼럼에 대한 누적 합 계산
```

Out[102]:

	data0	data1	data2
one	10	100	1000
two	30	300	3000
three	60	600	6000
four	100	1000	10000

In [103]:
```
df.cumsum(axis=1)  #로우에 대한 누적 합 계산
```

Out[103]:

	data0	data1	data2
one	10	110	1110
two	20	220	2220
three	30	330	3330
four	40	440	4440

In [ ]:
```

```

5. 데이터 정렬 및 계산

In [104]:
```
data = {'c': [1, 3, 2, 4],
        'a': [20, 10, 40, 30],
        'b': [400, 200, 300, 100]}
```

In [105]:
```
a = DataFrame(data, index=[1, 4, 2, 3])
a
```

Out[105]:

	c	a	b
1	1	20	400
4	3	10	200
2	2	40	300
3	4	30	100

(1) index 또는 columns을 기준으로 sorting 하기

In [106]:
```
#index를 기준으로 sorting
a.sort_index()
```

Out[106]:

	c	a	b
1	1	20	400
2	2	40	300
3	4	30	100
4	3	10	200

In [107]:
```
#index 기준 역순으로 sorting
a.sort_index(ascending=False)
```

Out[107]:

	c	a	b
4	3	10	200
3	4	30	100
2	2	40	300
1	1	20	400

In [108]:
```
#columns을 기준으로 sorting
a.sort_index(axis=1)
```

Out[108]:

	a	b	c
1	20	400	1
4	10	200	3
2	40	300	2
3	30	100	4

In [109]:
```
#columns 기준 역순으로 sorting
a.sort_index(axis=1, ascending=False)
```

Out[109]:

	c	b	a
1	1	400	20
4	3	200	10
2	2	300	40
3	4	100	30

In [110]:
```
#index, columns 두가지 기준으로 sorting
a.sort_index().sort_index(axis=1)
```

Out[110]:

	a	b	c
1	20	400	1
2	40	300	2
3	30	100	4
4	10	200	3

(2) 데이터 값을 기준으로 sorting 하기

In [111]:
```
data = {'a': [1, 2, 1, 2],
        'b': [40, 30, 20, 10],
        'c': [400, 200, 300, 100]}
```

In [112]:
```
a = DataFrame(data, index=[1, 2, 3, 4])
a
```

Out[112]:

	a	b	c
1	1	40	400
2	2	30	200
3	1	20	300
4	2	10	100

In [113]:
```
#하나의 column 값을 기준으로 sorting
a.sort_values(by='a')
```

Out[113]:

	a	b	c
1	1	40	400
3	1	20	300
2	2	30	200
4	2	10	100

In [114]:
```
#두개의 columns 값을 기준으로 sorting
a.sort_values(by=['a', 'b']) #<= a기준으로 sorting한 후 b를 기준으로 sort
ing
```

Out[114]:

	a	b	c
3	1	20	300
1	1	40	400
4	2	10	100
2	2	30	200

In [ ]:
```

```