20강

pandas2

20-1 산술 연산과 정렬

- 산술연산
  - ✓ pandas에서 중요한 기능은 색인 다른 객체 간의 산술 연산이다. 객체를 연산 할 때 크기가 맞지 않는다면 한 쪽의 크기를 맞추어 연산이 진행된다.
  - ✓ 일반적인 연산 방법은 모자란 부분을 nan으로 채워 연산을 진행한다.
  - ✓ 산술 함수를 이용하면 fill_value를 이용하여 모자란 부분을 특정 값으로 채울 수 있다.

```
import pandas as pd
import numpy as np

df1 = pd.DataFrame(np.arange(12).reshape((3,4)),
                   columns=list('abcd'))
df2 = pd.DataFrame(np.arange(20).reshape((4,5)),
                   columns=list('abcde'))

print(df1)
print(df2)

print(df1 + df2)
print(df1.add(df2, fill_value=0))
```

[결과]

```
 a b  c  d
0 0 1  2  3
1 4 5  6  7
2 8 9 10 11

    a   b   c   d   e
0   0   1   2   3   4
1   5   6   7   8   9
2  10  11  12  13  14
3  15  16  17  18  19

      a     b     c     d   e
0   0.0   2.0   4.0   6.0 NaN
1   9.0  11.0  13.0  15.0 NaN
2  18.0  20.0  22.0  24.0 NaN
3   NaN   NaN   NaN   NaN NaN

      a     b     c     d     e
0   0.0   2.0   4.0   6.0   4.0
1   9.0  11.0  13.0  15.0   9.0
2  18.0  20.0  22.0  24.0  14.0
3  15.0  16.0  17.0  18.0  19.0
```

20-1 산술 연산과 정렬

- 정렬
  - ✓ pandas에서는 정렬하기 위한 두 함수를 제공한다.
  - ✓ sort_index 함수는 인덱스를 기준으로 정렬하며 이때 축도 같이 이동을 한다.
  - ✓ Dataframe을 정렬할 때는 axis를 이용하여 인덱스 또는 컬럼 기준으로 정렬을 할 수 있다.

```
import numpy as np
import pandas as pd

obj = pd.Series(np.arange(4), index=list('dabc'))
print(obj)
print(obj.sort_index())

frame = pd.DataFrame(np.arange(8).reshape(2,4),
                     index=['three','one'],
                     columns=list('dabc'))
print(frame)
print(frame.sort_index(axis=1))
print(frame.sort_index(axis=1, ascending=False))
print(frame.sort_index(axis=0))
```

```
[결과]
d    0
a    1
b    2
c    3
dtype: int32

a    1
b    2
c    3
d    0
dtype: int32

       d  a  b  c
three  0  1  2  3
one    4  5  6  7

       a  b  c  d
three  1  2  3  0
one    5  6  7  4

       d  c  b  a
three  0  3  2  1
one    4  7  6  5

       d  a  b  c
one    4  5  6  7
three  0  1  2  3
```

20-1 산술 연산과 정렬

- 정렬
  - ✓ Sort_value 함수는 value값을 기준으로 정렬하며 이때 축도 같이 이동을 한다.
  - ✓ Dataframe을 정렬할 때는 'by='를 이용하여, 기준 값을 지정하여 정렬을 시켜준다.

```
import numpy as np
import pandas as pd

obj2 = pd.Series([4,7,-5,3])
print(obj2.sort_values())

data = {'b':[4,7,-3,2], 'a':[0,1,0,1]}
frame2 = pd.DataFrame(data)

print(frame2)
print(frame2.sort_values(by='b'))
print(frame2.sort_values(by=['a','b']))
```

```
[결과]
2   -5
3    3
0    4
1    7
dtype: int64

   b  a
0  4  0
1  7  1
2 -3  0
3  2  1

   b  a
2 -3  0
3  2  1
0  4  0
1  7  1

   b  a
2 -3  0
0  4  0
3  2  1
1  7  1
```

20-2 기술 통계와 누락 처리

- 기술 통계
  - ✓ pandas 객체는 수학 메서드와 통계 메서드를 제공한다.
  - ✓ numpy 배열에서 제공하는 함수와 비슷하지만 pandas의 함수에는 누락된 데이터를 제외하는 기능이 추가되어 있다.

메서드	설명
count	NA 값을 제외한 값의 수를 반환한다.
describe	Series나 Dataframe의 각 칼럼에 대한 요약 통계를 계산한다.
min, max	최솟값, 최댓값을 계산한다.
argmin, argmax	각각 최솟값과 최댓값을 갖고 있는 색인의 위치를 반환한다.
quantile	0부터 1까지의 분위수를 계산한다.
sum	합을 계산한다.
mean	평균을 계산한다.
var	표본 분산의 값을 구한다.
std	표본 정규 분산의 값을 구한다.
cumsum, cmprod	누적합, 누적곱을 구한다.

20-2 기술 통계와 누락 처리

- 누락 처리
  - ✓ pandas의 설계 목표 중 하나는 누락 데이터를 가능한 쉽게 처리할 수 있도록 하는 것이다.
  - ✓ pandas는 누락된 데이터는 모두 NaN(Not a Number)으로 취급한다.
  - ✓ NA 처리 메서드는 아래 표와 같다.

인자	설명
dropna	누락된 데이터가 있는 축(로우, 칼럼)을 제외 시킨다.
fillna	누락된 데이터를 대신할 값을 채우거나 'ffill' 또는 'bfill' 같은 보간 메서드를 적용한다.
isnull	누락된 NA인 값을 알려주는 불리언 값이 저장된 같은 형의 객체를 반환한다.
notnull	isnull과 반대되는 메서드이다.

20-2 기술 통계와 누락 처리

- 누락 처리
  - ✓ 누락된 값이 있는 값을 제거하고 싶을 때는 drop_na 함수를 이용하면 된다.
  - ✓ 누락된 값을 제외시키지 않고 메우고 싶은 경우 fillna 함수를 활용하면 된다.

```
import pandas as pd
import numpy as np

data = pd.Series([1,np.nan, 3.5, np.nan, 7])
print(data)
print(data.dropna())

np.random.seed(12345)
frame = pd.DataFrame(np.random.randn(7,3))
frame.iloc[:4, 1] = np.nan
frame.iloc[:2, 2] = np.nan
print(frame)
print(frame.fillna(0))
```

[결과]

```
0    1.0
1    NaN
2    3.5
3    NaN
4    7.0
dtype: float64

0    1.0
2    3.5
4    7.0
dtype: float64

          0         1         2
0 -0.204708       NaN       NaN
1 -0.555730       NaN       NaN
2  0.092908       NaN  0.769023
3  1.246435       NaN -1.296221
4  0.274992  0.228913  1.352917
5  0.886429 -2.001637 -0.371843
6  1.669025 -0.438570 -0.539741

          0         1         2
0 -0.204708  0.000000  0.000000
1 -0.555730  0.000000  0.000000
2  0.092908  0.000000  0.769023
3  1.246435  0.000000 -1.296221
4  0.274992  0.228913  1.352917
5  0.886429 -2.001637 -0.371843
6  1.669025 -0.438570 -0.539741
```