pandas, numpy module 불러오기

In [1]:
```
import pandas as pd
import numpy as np
```

7-1. 통계 함수

주요 통계 함수 목록

메소드	설명
count	NA 값을 제외한 값의 수를 반환
Describe	Series나 DataFrame의 각 컬럼에 대한 요약 통계를 계산
min, max	최소, 최대값 반환
sum	합을 계산
mean	평균을 계산
median	중위값(50% 분위)을 계산
var	표본 분산의 값 계산
Std	표본 정규분포 계산
Skew	비대칭도의 값 계산
cumsum	누적 합을 구함
diff	1차 산술 차 계산

주요 함수 옵션

인자	설명
axis	연산을 수행할 축. 기본값 = 0 만약 0이면, 컬럼 단위로 수행. 만약 1이면, 로우 단위로 수행
skipna	누락된 값을 제외할 것인지 정함. 기본값 = True
level	계산하려는 축이 계층적 색인일 때 레벨에 따라 묶어서 계산 (추후 설명)

**** axis는 통계함수 뿐만 아니라, 많은 함수에서 동일하게 적용됨. ****

In [2]:
```
# 샘플 데이터 생성
np.random.seed(321)

df = pd.DataFrame(np.random.randint(50, 100, (5, 4)),
                  columns = [np.repeat([2016, 2017], 2), ['영어', '수학'] * 2],
                  index = ['Kim','Park','Lee','Jung','Moon'])

df.index.set_names('학생명', inplace=True)
df.columns.set_names(['연도','과목'], inplace=True)
df
```

Out[2]:

연도	2016		2017	
과목	영어	수학	영어	수학
학생명				
Kim	76	81	91	58
Park	67	90	76	74
Lee	87	58	69	71
Jung	51	77	76	63
Moon	93	81	60	87

In [3]:
```
# Kim 학생의 2016년 영어 성적을 NA로 처리하기
df.loc['Kim', (2016, '영어')] = np.nan
```

In [4]:
```
# df 확인
df
```

Out[4]:

연도	2016		2017	
과목	영어	수학	영어	수학
학생명				
Kim	NaN	81	91	58
Park	67.0	90	76	74
Lee	87.0	58	69	71
Jung	51.0	77	76	63
Moon	93.0	81	60	87

In [6]:
```
# 2016년 성적만 선택해서 df2016에 저장
# 가장 상위 컬럼이므로 그대로 작성
df2016 = df[2016]
df2016
```

Out[6]:

과목	영어	수학
학생명		
Kim	NaN	81
Park	67.0	90
Lee	87.0	58
Jung	51.0	77
Moon	93.0	81

In [7]:
```
# 1) count() - NaN 값을 제외한 데이터의 갯수
df.count()
```

Out[7]:
```
연도    과목
2016  영어    4
      수학    5
2017  영어    5
      수학    5
dtype: int64
```

In [8]:
```
# 원래 값으로 되돌려 놓기
df.loc['Kim', (2016, '영어')] = 76
```

In [9]:
```
# 2) describe() - 각 컬럼에 대한 요약 통계량을 확인해주는 함수
df.describe()
```

Out[9]:

연도	2016		2017	
과목	영어	수학	영어	수학
count	5.000000	5.00000	5.000000	5.000000
mean	74.800000	77.40000	74.400000	70.600000
std	16.649324	11.84483	11.371016	11.148991
min	51.000000	58.00000	60.000000	58.000000
25%	67.000000	77.00000	69.000000	63.000000
50%	76.000000	81.00000	76.000000	71.000000
75%	87.000000	81.00000	76.000000	74.000000
max	93.000000	90.00000	91.000000	87.000000

In [10]:
```
# 3) sum() - 값의 합을 계산
# 기본적으로 함수는 각 컬럼에 있는 로우(row)들의 값들로 수행 (default: axis =0)
# 원래 column 접근할 때는 1인데, 계산할 때는 반대
df2016.sum(axis = 0, skipna = True)
# 기본값 이므로 = df2016.sum()
```

Out[10]:
```
과목
영어    298.0
수학    387.0
dtype: float64
```

In [12]:
```
# 4) mean() - 값의 평균을 계산
# 각 학생들의 과목 평균 구하기
df2016.mean(axis = 1)
```

Out[12]:
```
학생명
Kim      81.0
Park     78.5
Lee      72.5
Jung     64.0
Moon     87.0
dtype: float64
```

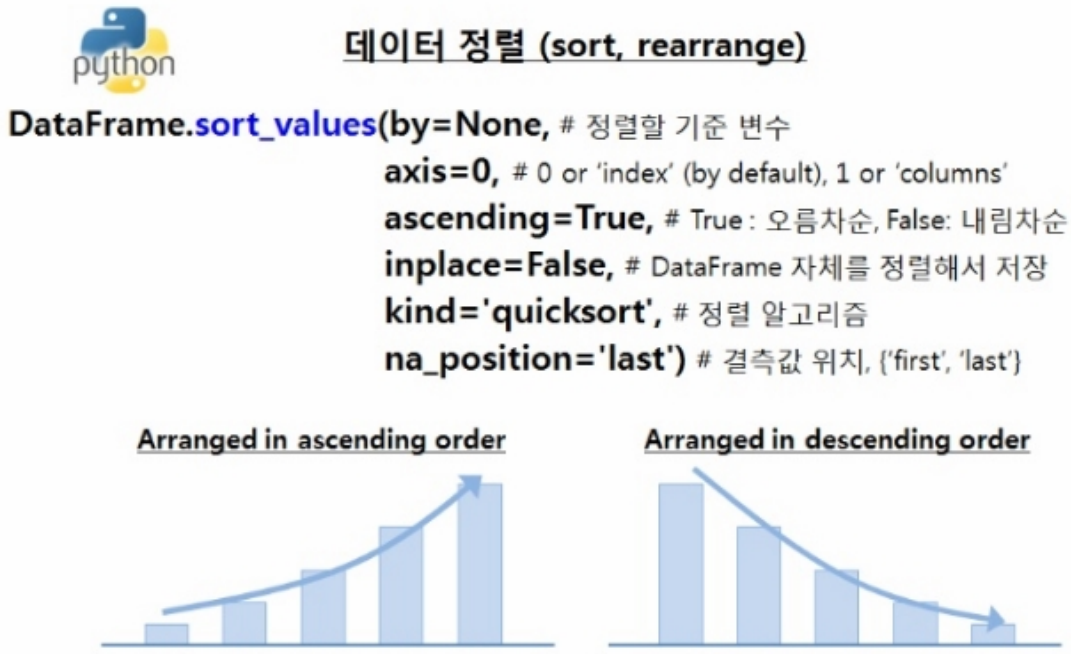
In [13]:
```
# 5) cumsum() - cumulative sum
df2016.cumsum(axis = 1)
```

Out[13]:

과목	영어	수학
학생명		
Kim	NaN	81.0
Park	67.0	157.0
Lee	87.0	145.0
Jung	51.0	128.0
Moon	93.0	174.0

7-2. 정렬

* 데이터 정렬 : sort_values()



* 인덱스 정렬 : sort_index() - 인자는 sort_values()와 동일

1) Series 정렬

In [15]:
```
# Series 정렬을 위한 샘플 데이터
sr = pd.Series([3, 5, 2, 1, 7, np.nan], index = list('bcafed'))
sr
```

Out[15]:
```
b    3.0
c    5.0
a    2.0
f    1.0
e    7.0
d    NaN
dtype: float64
```

A. index 정렬

In [16]:
```
# 오름차순 정렬, 기본값 = True
sr.sort_index(ascending = True)
```

Out[16]:
```
a    2.0
b    3.0
c    5.0
d    NaN
e    7.0
f    1.0
dtype: float64
```

In [17]:
```
# 내림차순 정렬
sr.sort_index(ascending = False)
```

Out[17]:
```
f    1.0
e    7.0
d    NaN
c    5.0
b    3.0
a    2.0
dtype: float64
```

B. value 정렬

In [18]:
```
# NA값을 맨 마지막으로 오름차순 정렬, 기본값 = True
sr.sort_values(ascending = True, na_position = 'last')
```

Out[18]:
```
f    1.0
a    2.0
b    3.0
c    5.0
e    7.0
d    NaN
dtype: float64
```

In [20]:
```
# NA값을 맨 처음으로 내림차순 정렬
sr.sort_values(ascending = False, na_position = 'first')
```

Out[20]:
```
d    NaN
e    7.0
c    5.0
b    3.0
a    2.0
f    1.0
dtype: float64
```

2) DataFrame 정렬

In [21]:
```
# sample data 생성
dic = {'영어':[10, 60, 80, 30, 40, 50, 20, 70, 30, 90],
       '수학':[90, 80, 70, 60, 50, 50, 40, 30, 30, 20]}

df2 = pd.DataFrame(dic)
df2
```

Out[21]:

	영어	수학
0	10	90
1	60	80
2	80	70
3	30	60
4	40	50
5	50	50
6	20	40
7	70	30
8	30	30
9	90	20

A. index 정렬

In [22]:
```
# row index 정렬 (default: axis=0)
df2.sort_index(axis = 0)
```

Out[22]:

	영어	수학
0	10	90
1	60	80
2	80	70
3	30	60
4	40	50
5	50	50
6	20	40
7	70	30
8	30	30
9	90	20

In [23]:
```
# column index 정렬
df2.sort_index(axis = 1)
```

Out[23]:

	수학	영어
0	90	10
1	80	60
2	70	80
3	60	30
4	50	40
5	50	50
6	40	20
7	30	70
8	30	30
9	20	90

B. value 정렬

In [24]:
```
# 수학 성적으로 오름차순 정렬
df2.sort_values(by = '수학', ascending = True)
```

Out[24]:

	영어	수학
9	90	20
7	70	30
8	30	30
6	20	40
4	40	50
5	50	50
3	30	60
2	80	70
1	60	80
0	10	90

In [25]:
```
# 영어 성적으로 내림차순 정렬
df2.sort_values(by = '영어', ascending = False)
```

Out[25]:

	영어	수학
9	90	20
2	80	70
7	70	30
1	60	80
5	50	50
4	40	50
3	30	60
8	30	30
6	20	40
0	10	90

In [26]:
```
# column index 정렬하고 수학을 기준으로 내림차순 정렬
df2.sort_index(axis = 1, inplace = True)
df2.sort_values(by = '수학', ascending = False)
```

Out[26]:

	수학	영어
0	90	10
1	80	60
2	70	80
3	60	30
4	50	40
5	50	50
6	40	20
7	30	70
8	30	30
9	20	90

In [27]:
```
# 2개의 기준으로 정렬
# 수학을 기준으로 내림차순 정렬하고 수학 점수가 같은 경우 영어 성적으로 내림차순 정렬
df2.sort_values(by = ['수학', '영어'], ascending = [False, False])
# = df2.sort_values(by = ['수학', '영어'], ascending = False)
```

Out[27]:

	수학	영어
0	90	10
1	80	60
2	70	80
3	60	30
5	50	50
4	50	40
6	40	20
7	30	70
8	30	30
9	20	90

In [28]:
```
# 수학을 기준으로 내림차순 정렬하고 수학 점수가 같은 경우 영어 성적으로 오름차순 정렬
df2.sort_values(by = ['수학', '영어'], ascending = [False, True])
```

Out[28]:

	수학	영어
0	90	10
1	80	60
2	70	80
3	60	30
4	50	40
5	50	50
6	40	20
8	30	30
7	30	70
9	20	90

3) 계층 색인 DataFrame 정렬

In [29]:
```
# data frame확인
df
```

Out[29]:

연도	2016		2017	
과목	영어	수학	영어	수학
학생명				
Kim	76.0	81	91	58
Park	67.0	90	76	74
Lee	87.0	58	69	71
Jung	51.0	77	76	63
Moon	93.0	81	60	87

In [30]:
```
# 2017년 수학 성적 기준으로 내림차순 정렬
df.sort_values(by = (2017, '수학'), ascending = False)
```

Out[30]:

연도	2016		2017	
과목	영어	수학	영어	수학
학생명				
Moon	93.0	81	60	87
Park	67.0	90	76	74
Lee	87.0	58	69	71
Jung	51.0	77	76	63
Kim	76.0	81	91	58

In [31]:
```
# 2016년 평균을 구하고 2016년 평균을 기준으로 내림차순 정렬
df[2016].mean(axis = 1)
```

Out[31]:
```
학생명
Kim      78.5
Park     78.5
Lee      72.5
Jung     64.0
Moon     87.0
dtype: float64
```

In [33]:
```
df[(2016, '평균')] = df[2016].mean(axis = 1)
df
```

Out[33]:

연도	2016		2017		2016
과목	영어	수학	영어	수학	평균
학생명					
Kim	76.0	81	91	58	78.5
Park	67.0	90	76	74	78.5
Lee	87.0	58	69	71	72.5
Jung	51.0	77	76	63	64.0
Moon	93.0	81	60	87	87.0

In [34]:
```
df.sort_values(by = (2016, '평균'), ascending = False)
```

Out[34]:

연도	2016		2017		2016
과목	영어	수학	영어	수학	평균
학생명					
Moon	93.0	81	60	87	87.0
Kim	76.0	81	91	58	78.5
Park	67.0	90	76	74	78.5
Lee	87.0	58	69	71	72.5
Jung	51.0	77	76	63	64.0