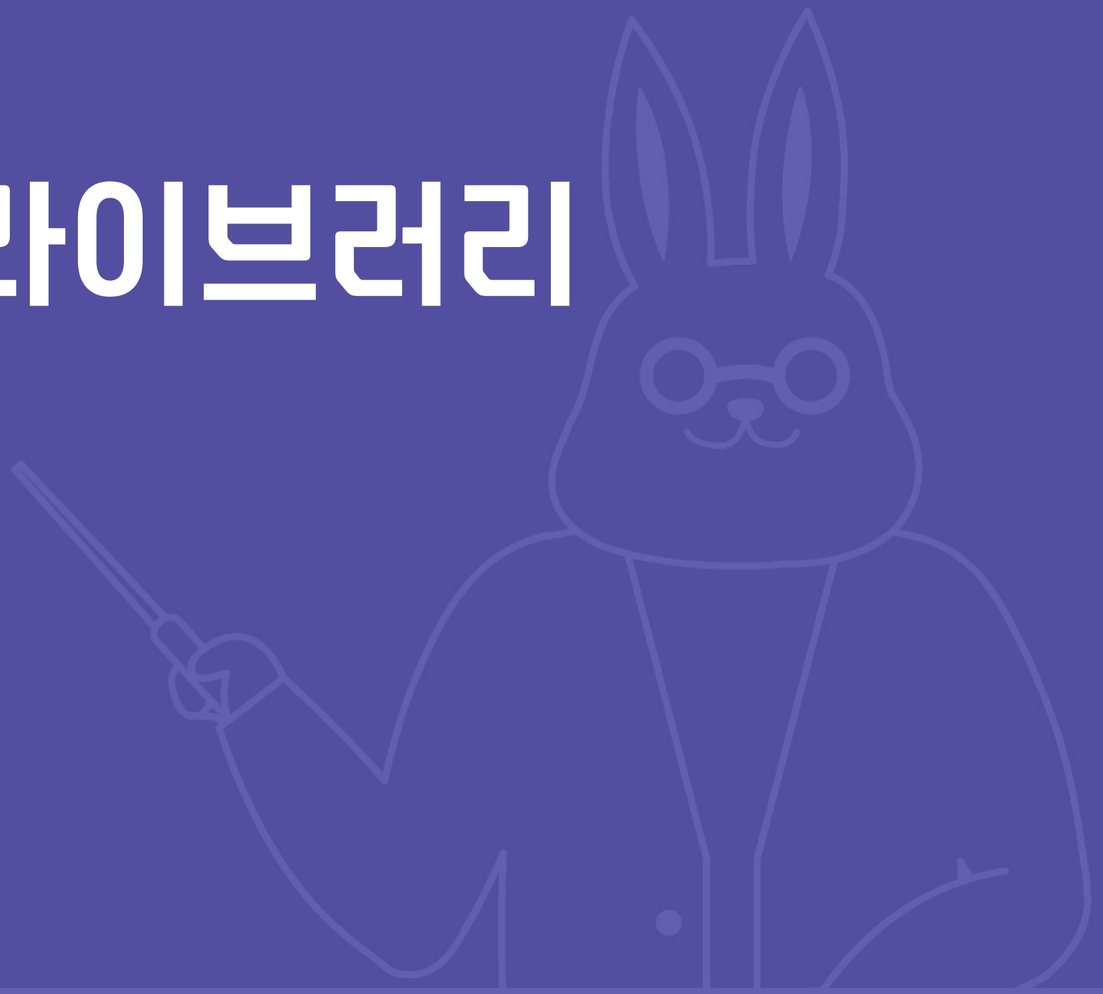
데이터 분석을 위한 라이브러리

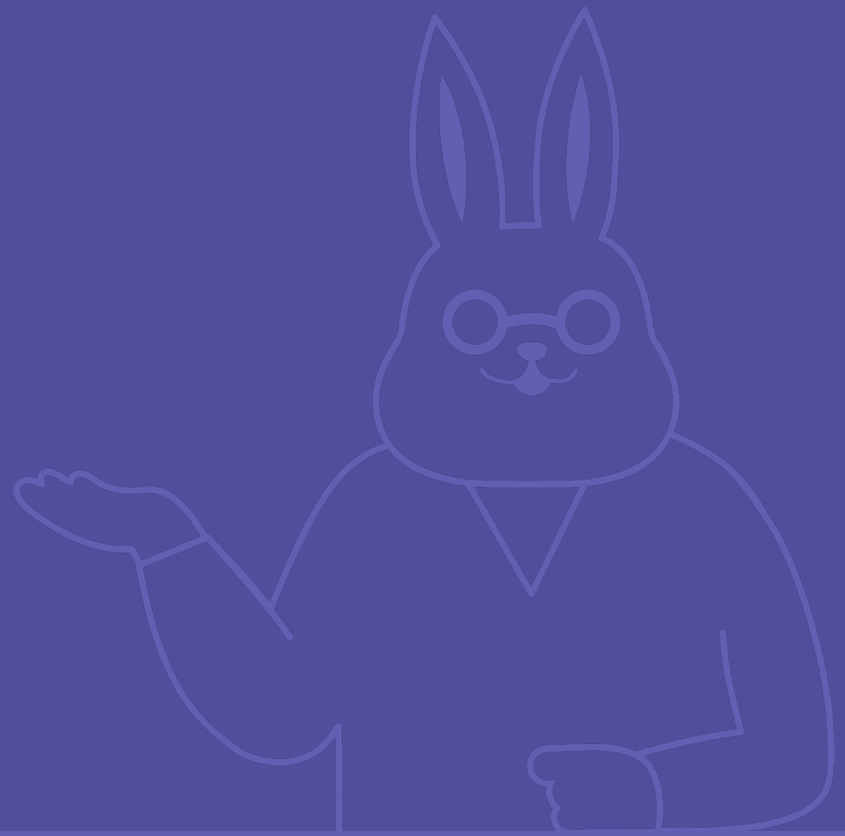
04 데이터 조작 및 분석을 위한 Pandas 심화



/* elice */

01

데이터프레임 정렬하기



Index 값 기준으로 정렬하기

axis = 0 : 행 인덱스 기준 정렬(Default 오름차순)

```
df = df.sort_index(axis = 0)
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df 원본

	col1	col2	col3
1	1	A	1
0	2	A	0
2	9	B	9
4	7	D	2
3	8	NaN	4
5	4	C	3

결과값

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

Index 값 기준으로 정렬하기

axis = 1 : 열 인덱스 기준 내림차순 정렬

```
df.sort_index(axis = 1, ascending = False)
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df 원본

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

결과값

	col3	col2	col1
0	0	A	2
1	1	A	1
2	9	B	9
3	4	NaN	8
4	2	D	7
5	3	C	4

Column 값 기준으로 정렬하기

col1 컬럼 기준 정렬(Default 오름차순)

```
df.sort_values('col1', ascending = True)
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df 원본

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

결과값

	col1	col2	col3
1	1	A	1
0	2	A	0
5	4	C	3
4	7	D	2
3	8	NaN	4
2	9	B	9

Column 값 기준으로 정렬하기

col1 컬럼 기준 내림차순 정렬

```
df.sort_values('col1', ascending = False)
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df 원본

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

결과값

	col1	col2	col3
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3
0	2	A	0
1	1	A	1

Column 값 기준으로 정렬하기

col2 컬럼 기준 오름차순 정렬 후 col2 컬럼 기준 내림차순 정렬

```
df.sort_values(['col2', 'col1'], ascending = [True, False])
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df 원본

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
3	8	NaN	4
4	7	D	2
5	4	C	3

중간과정

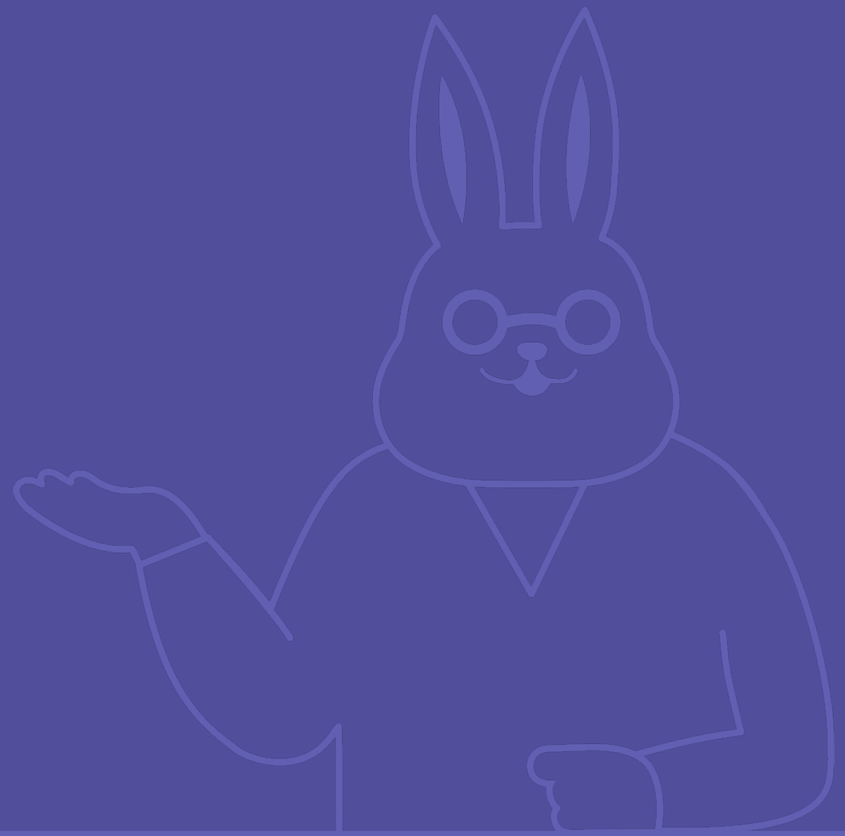
	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
5	4	C	3
4	7	D	2
3	8	NaN	4

결과값

	col1	col2	col3
0	2	A	0
1	1	A	1
2	9	B	9
5	4	C	3
4	7	D	2
3	8	NaN	4

02

데이터프레임 분석용 함수



집계함수 – count

count 메서드 활용하여 데이터 개수 확인 가능

(Default : **NaN값 제외**)

```
data = {
    'korean': [50, 60, 70],
    'math': [10, np.nan, 40]
}
df = pd.DataFrame(data, index = ['a','b','c'])
df.count(axis = 0) # 열 기준
df.count(axis = 1) # 행 기준
```

df

	korean	math
a	50	10.0
b	60	NaN
c	70	40.0

열 기준

```
korean    3
math      2
dtype: int64
```

(Series)

행 기준

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

(Series)

집계함수 – max, min

max, min 메서드 활용하여 최대, 최소값 확인 가능

(Default : **열 기준, NaN값 제외**)

```
data = {
    'korean': [50, 60, 70],
    'math': [10, np.nan, 40]
}
df = pd.DataFrame(data, index = ['a','b','c'])
df.max()     # 최댓값
df.min()     # 최솟값
```

df

	korean	math
a	50	10.0
b	60	NaN
c	70	40.0

최댓값

```
korean    70.0
math      40.0
dtype: float64
```

(Series)

최솟값

```
korean    50.0
math      10.0
dtype: float64
```

(Series)

집계함수 – sum, mean

sum, mean 메서드 활용하여 합계 및 평균 계산

(Default : 열 기준, NaN값 제외)

```
data = {
    'korean': [50, 60, 70],
    'math': [10, np.nan, 40]
}
df = pd.DataFrame(data, index = ['a','b','c'])
df.sum()     # 합계
df.mean()    # 평균
```

df

	korean	math
a	50	10.0
b	60	NaN
c	70	40.0

합계

```
korean    180.0
math       50.0
dtype: float64
```

(Series)

평균

```
korean    60.0
math      25.0
dtype: float64
```

(Series)

집계함수 – sum, mean

axis, skipna 인자 활용하여 합계 및 평균 계산

(행 기준, NaN값 포함 시)

```
data = {
    'korean': [50, 60, 70],
    'math': [10, np.nan, 40]
}
df = pd.DataFrame(data, index = ['a','b','c'])
df.sum(axis = 1) # 합계
df.mean(axis = 1, skipna = False) # 평균
```

df원본

	korean	math
a	50	10.0
b	60	NaN
c	70	40.0

합계

```
a     60.0
b     60.0
c    110.0
dtype: float64
```

(Series)

평균

```
a    30.0
b     NaN
c    55.0
dtype: float64
```

(Series)

집계함수 – sum, mean

NaN값이 존재하는 **column의 평균** 구하여 **NaN값 대체**

```
…
B_avg = df['math'].mean()
print(B_avg)  # 25.0

# NaN값 대체
df['math'] = df['math'].fillna(B_avg)
# 평균
df.mean(axis = 1, skipna = False)
```

df원본

	korean	math
a	50	10.0
b	60	NaN
c	70	40.0

NaN값 대체

	korean	math
a	50	10.0
b	60	25.0
c	70	40.0

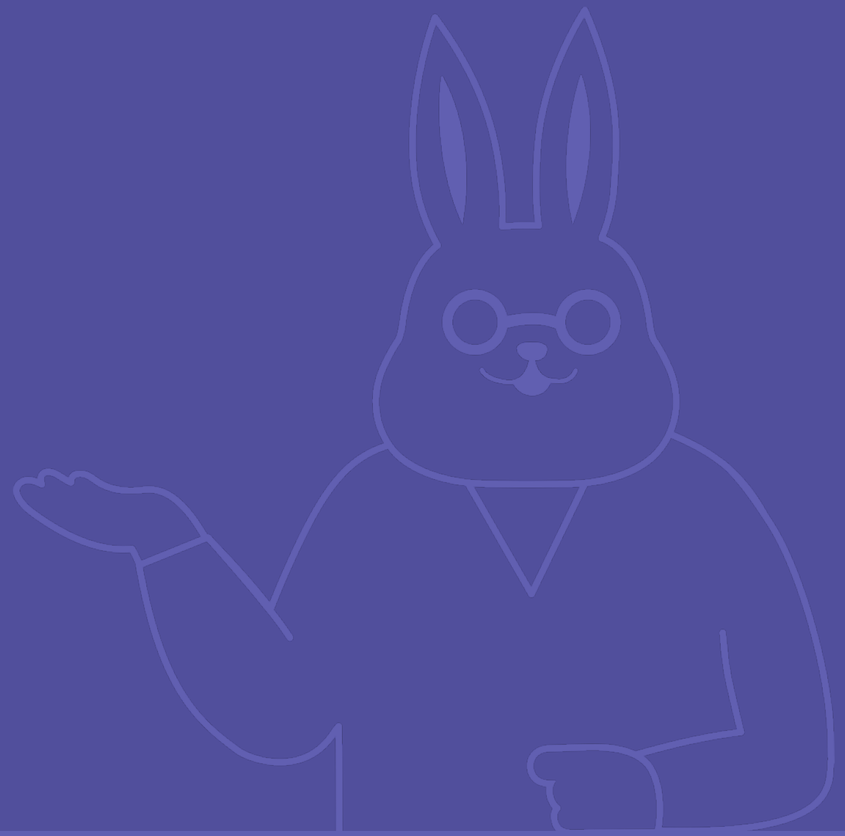
평균

a	30.0
b	42.5
c	55.0
dtype: float64	

(Series)

03

그룹으로 묶기



group by

간단한 집계를 넘어서서 **조건부로 집계**하고 싶은 경우

```
df = pd.DataFrame({
'data1' : range(6),
'data2' : [4,4,6,0,6,1],
'key':['A','B','C','A','B','C']
})
df.groupby('key').sum() #1번
df.groupby(['key','data1']).sum() #2번
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

1번

key	data1	data2
A	3	4
B	5	10
C	7	7

2번

key	data1	data2
A	0	4
	3	0
B	1	4
	4	6
C	2	6
	5	1

aggregate

groupby를 통해서 집계를 한번에 계산하는 방법

```
df.groupby('key').aggregate(['min', np.median, max]) #1번
df.groupby('key').aggregate({'data1': 'min', 'data2': np.sum}) #2번
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

1번

	data1			data2		
	min	median	max	min	median	max
key						
A	0	1.5	3	0	2.0	4
B	1	2.5	4	4	5.0	6
C	2	3.5	5	1	3.5	6

2번

	data1	data2
key		
A	0	4
B	1	10
C	2	7

filter

groupby를 통해서 그룹 속성을 기준으로 데이터 필터링

```
def filter_by_mean(x):
    return x['data2'].mean() > 3
df.groupby('key').mean()  #1번
df.groupby('key').filter(filter_by_mean)  #2번
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

1번

	data1	data2
key		
A	1.5	2.0
B	2.5	5.0
C	3.5	3.5

2번

	data1	data2	key
1	1	4	B
2	2	6	C
4	4	6	B
5	5	1	C

apply, lambda

groupby를 통해서 묶인 데이터에 함수 적용

```
df.groupby('key').apply(lambda x: x.max() - x.min())
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

\# 1번

key	data1	data2
A	3	4
B	3	2
C	3	5

get_group

groupby로 묶인 데이터에서 key값으로 데이터를 가져올 수 있다

```
df = pd.read_csv("./univ.csv")
# 상위 5개 데이터
df.head()
# 데이터 추출
df.groupby("시도").get_group("충남")
len(df.groupby("시도").get_group("충남"))
# 94
```

*pandas 라이브러리는 이미 import 해둔 것으로 가정

df 상단 5개

	시도	학교명
0	충남	충남도립청양대학
1	경기	한국복지대학교
2	경북	가톨릭상지대학교
3	전북	군산간호대학교
4	경남	거제대학교

데이터 추출

	시도	학교명
0	충남	충남도립청양대학
44	충남	신성대학교
60	충남	백석문화대학교
67	충남	혜전대학교
92	충남	아주자동차대학
112	충남	천안연암대학

크레딧

/* elice */

코스 매니저
하주희

콘텐츠 제작자
임원균, 하주희

강사
황지영

감수자
장석준

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io