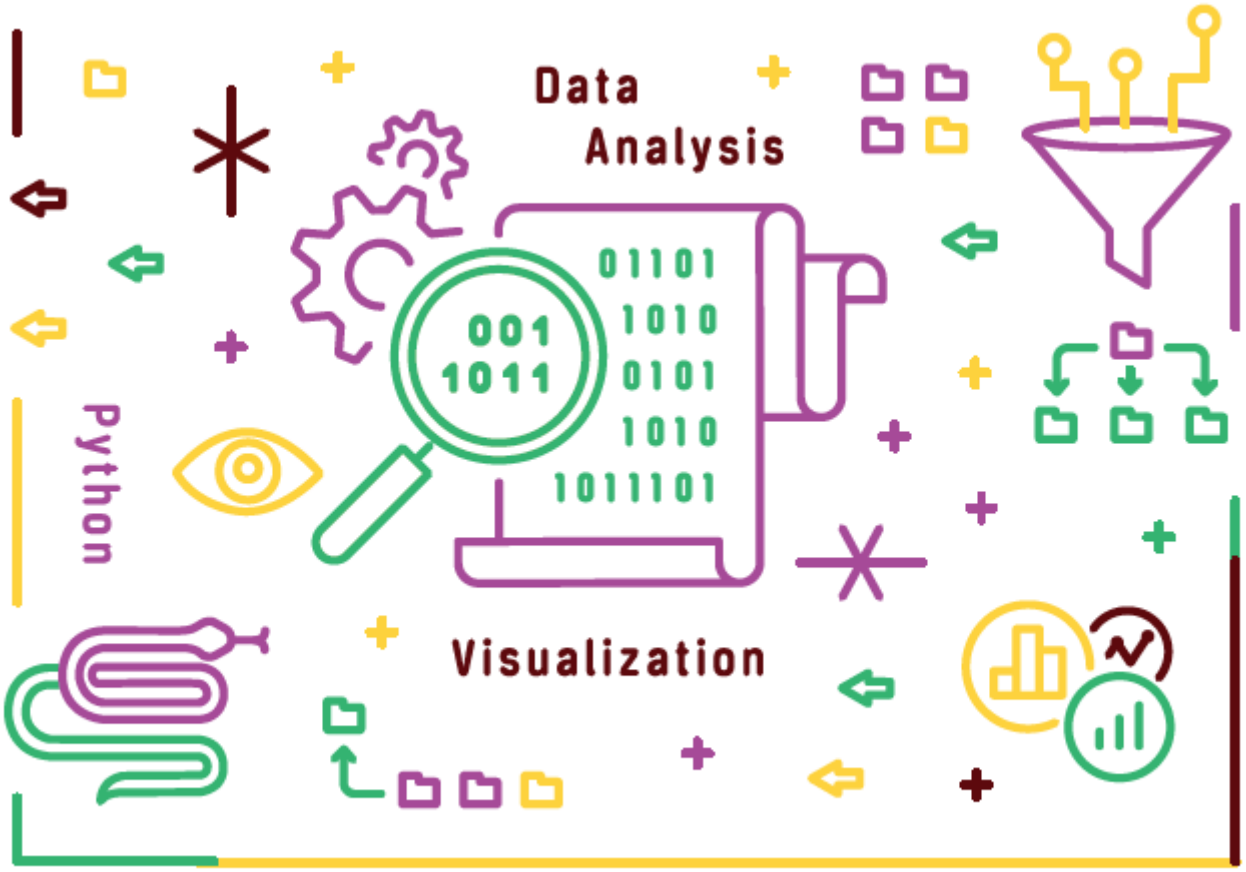


<5장> 데이터 처리가 쉬운 판다스

학습 목표

- 판다스 개념 및 특징을 익힌다.
- 판다스 객체 생성하는 방법을 익힌다.
- 데이터를 확인하는 다양한 방법을 익힌다.
- 데이터를 가공하고 그룹핑 하는 방법을 익힌다.
- 결측데이터를 처리하는 방법을 익힌다.
- 데이터를 병합하는 방법을 익힌다.

목차

01

판다스 개념 및 특징

1. 판다스 개념 및 특징

판다스

- 고수준의 자료구조와 빠르고 쉬운 데이터 분석 도구를 제공하는 파이썬의 라이브러리

판다스(Pandas) 특징

- 자동적/명시적으로 축의 이름에 따라 데이터를 정렬할 수 있는 데이터구조
- 잘못 정렬된 데이터에 의한 오류를 방지하고, 다양한 방식으로 색인된 데이터를 다룰 수 있는 기능
- 통합된 시계열 기능
- 시계열 데이터와 비시계열 데이터를 함께 다룰 수 있는 통합 자료구조
- 산술 연산과 한 축의 모든 값을 더하는 등 데이터 축약 연산은 축의 이름 같은 메타데이터로 전달될 수 있어야 함
- 누락된 데이터를 유연하게 처리할 수 있는 기능
- SQL 같은 일반 데이터베이스처럼 데이터를 합치고 관계 연산을 수행하는 기능

1. 판다스 개념 및 특징

판다스

- 패키지 설치 명령문

```
pip install pandas
```

- 판다스 import 코드

```
import pandas as pd
```

02

판다스 객체 생성

2. 판다스 객체 생성

Series와 DataFrame

- Series : 다양한 자료형을 담을 수 있는 1차원의 배열
- DataFrame : 행과 열을 갖는 2차원의 자료형

	Brand
0	Samsung
1	LG
2	Apple
3	Apple

Series

+

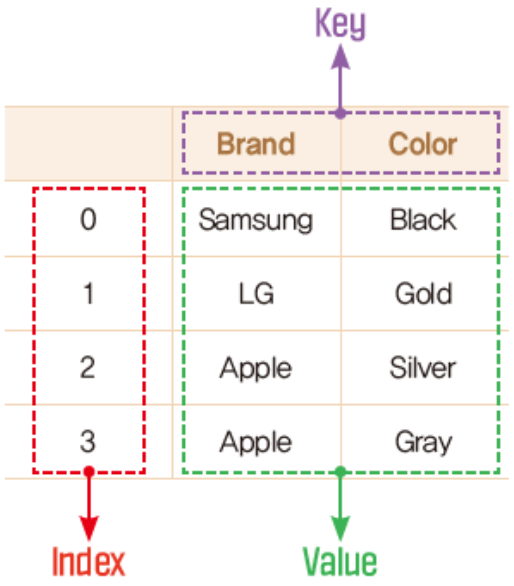
	Color
0	Black
1	Gold
2	Silver
3	Gray

Series

=

	Brand	Color
0	Samsung	Black
1	LG	Gold
2	Apple	Silver
3	Apple	Gray

DataFrame



2. 판다스 객체 생성

Series 만들기

실수형 Series 만들기

```
1
2
```

```
s=pd.Series([4, 3.5, 3.8, 3, 3.7])
s
```

〈실행결과〉

```
0    4.0
1    3.5
2    3.8
3    3.0
4    3.7
dtype: float64
```

정수형 Series 만들기

```
1
2
3
```

```
import pandas as pd
score=pd.Series([95, 90, 85, 90, 95])
score
```

〈실행결과〉

```
0    95
1    90
2    85
3    90
4    95
dtype: int64
```

인덱스 지정

```
1
2
```

```
s=pd.Series([90,80,95],index=['A','B','C'])
s
```

〈실행결과〉

```
A    90
B    80
C    95
dtype: int64
```

2. 판다스 객체 생성

DataFrame 만들기

- 넘파이 배열과 딕셔너리 구조를 이용하여 데이터 프레임을 생성한다.

넘파이로 배열 생성

〈코드〉

```
1  import numpy as np
2  import pandas as pd
3  arr=np.arange(12).reshape(4,3)
4  print(arr)
5  print()
6  df=pd.DataFrame(arr)
7  print(df)
```

〈실행결과〉

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]

   0   1   2
0  0   1   2
1  3   4   5
2  6   7   8
3  9  10  11
```

딕셔너리로 배열 생성

〈코드〉

```
1  df=pd.DataFrame({'name':['장은실','오경선','양숙희'],
2                   'score':[90,80,95],
3                   'dept':['com','eng','math']})
4  df
```

〈실행결과〉

	name	score	dept
0	장은실	90	com
1	오경선	80	eng
2	양숙희	95	math

2. 판다스 객체 생성

csv 활용

- 데이터를 DataFrame으로 불러올 수 있다.

csv 파일 불러오기

```
1  df=pd.read_csv('mobile.csv')
2  df
```

〈실행결과〉

	Brand	Color	price
0	Samsung	Black	100
1	LG	Gold	70
2	Apple	Silver	150
3	Apple	Gray	120

csv 파일 불러오기 : 인덱스 지정

```
1  df=pd.read_csv('mobile.csv', index_col=0)
2  df
```

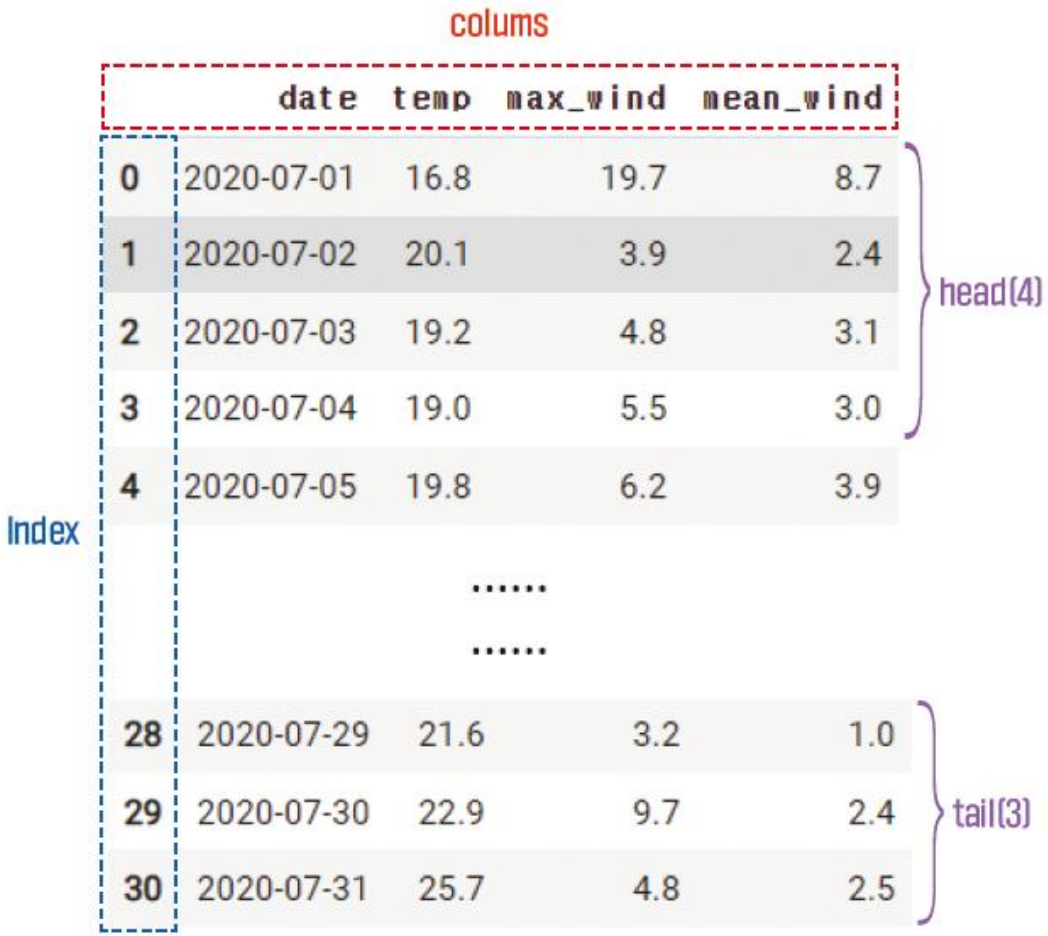
〈실행결과〉

Brand	Color	price
Samsung	Black	100
LG	Gold	70
Apple	Silver	150
Apple	Gray	120

03
판다스 데이터 확인

3. 판다스 데이터 확인

- 판다스 객체의 데이터를 확인하는 다양한 방법



	date	temp	max_wind	mean_wind
0	2020-07-01	16.8	19.7	8.7
1	2020-07-02	20.1	3.9	2.4
2	2020-07-03	19.2	4.8	3.1
3	2020-07-04	19.0	5.5	3.0
4	2020-07-05	19.8	6.2	3.9
	.....			
	.....			
28	2020-07-29	21.6	3.2	1.0
29	2020-07-30	22.9	9.7	2.4
30	2020-07-31	25.7	4.8	2.5

3. 판다스 데이터 확인

DataFrame 만들기

- CSV 파일의 컬럼명이 한글일 경우
  - encoding 속성을 다음과 같이 지정한다.
    - pd.read_csv('파일명', encoding='euc-kr')
    - pd.read_csv('파일명', encoding='cp949')

csv 파일 불러오기

```
1  df=pd.read_csv('weather.csv')
2  df
```

〈실행결과〉

	date	temp	max_wind	mean_wind
0	2010-08-01	28.7	8.3	3.4
1	2010-08-02	25.2	8.7	3.8
2	2010-08-03	22.1	6.3	2.9
3	2010-08-04	25.3	6.6	4.2
4	2010-08-05	27.2	9.1	5.6
...	...	...	...	...
3648	2020-07-27	22.1	4.2	1.7
3649	2020-07-28	21.9	4.5	1.6
3650	2020-07-29	21.6	3.2	1.0
3651	2020-07-30	22.9	9.7	2.4
3652	2020-07-31	25.7	4.8	2.5

3653 rows × 4 columns

3. 판다스 데이터 확인

- **df.shape/df.info( )**
  - 데이터의 (행, 열) 크기 확인
  - 데이터에 대한 전반적인 정보

(행, 열) 크기 확인

	〈코드〉	〈실행결과〉
1	df.shape	(3653, 4)

데이터의 전체적인 구조 출력

	〈코드〉	〈실행결과〉
1	df.info()	<class 'pandas.core.frame.DataFrame'> RangeIndex: 3653 entrles, D to 3652 Data columns (total 4 columns) : # column Non-Null Count Dtype --- ------ -------------- ---- 0 date 3653 non-null object 1 temp 3653 non-null float64 2 max_wind 3649 non-null float64 3 mean_wind 3647 non-null float64 dtypes: float64(3), object(1) memory usage: 114.3+ KB

3. 판다스 데이터 확인

- **df.head( )/df.tail( )**
  - 앞 부분과 마지막 부분을 확인

상위 N행 살펴보기

```
1  df.head()
```

〈실행결과〉

	date	temp	max_wind	mean_wind
0	2010-08-01	28.7	8.3	3.4
1	2010-08-02	25.2	8.7	3.8
2	2010-08-03	22.1	6.3	2.9
3	2010-08-04	25.3	6.6	4.2
4	2010-08-05	27.2	9.1	5.6

하위 N행 살펴보기

```
1  df.tail(3)
```

〈실행결과〉

	date	temp	max_wind	mean_wind
3650	2020-07-29	21.6	3.2	1.0
3651	2020-07-30	22.9	9.7	2.4
3652	2020-07-31	25.7	4.8	2.5

3. 판다스 데이터 확인

df.index/df.columns

- 인덱스(행 이름)와 열의 레이블(컬럼 이름) 출력

인덱스 출력

```
1  df.index
```

〈실행결과〉

```
RangleIndex(start=0, stop=3653, step=1)
```

컬럼 출력

```
1  df.columns
```

〈실행결과〉

```
Index([ ' date ' ,  ' temp ' ,  ' max_wind ' , mean_wind ' ], dtype= ' object ' )
```

3. 판다스 데이터 확인

- **df.describe( )**
  - 데이터의 컬럼별 요약 통계량

요약 통계량 확인

```
1  df.describe()
```

〈실행결과〉

	temp	max_wind	mean_wind
count	3653.000000	3649.000000	3647.000000
mean	12.942102	7.911099	3.936441
std	8.538507	3.029862	1.888473
min	-9.000000	2.000000	0.200000
25%	5.400000	5.700000	2.500000
50%	13.800000	7.600000	3.600000
75%	20.100000	9.700000	5.000000
max	31.300000	26.000000	14.900000

기온(temp) 컬럼의 평균 출력

〈코드〉

```
1  df['temp'].mean()
```

〈실행결과〉

```
12.942102381604148
```

3. 판다스 데이터 확인

df.sort_values( )

- 함수 형식
  - DataFrame.sort_values(by = ['정렬변수','정렬변수2',], ascending = False, inplace = True)
- by = [ ] : by = 을 사용하지 않아도 된다.
- inplace = True : 정렬 결과가 동일 데이터프레임 이름으로 저장된다.
- ascending = True : 디폴트이므로 오름차순 정렬이면 사용하지 않아도 된다.

최대풍속(max_wind) 컬럼의 값 크기에 따라 오름차순으로 정렬

1

〈코드〉

```
df.sort_values(by='max_wind')
```

〈실행결과〉

	date	temp	max_wind	mean_wind
1514	2014-09-23	20.7	2.0	1.0
1134	2013-09-08	20.4	2.1	0.8
421	2011-09-26	18.7	2.1	0.3
1512	2014-09-21	20.4	2.2	1.2
1005	2013-05-02	7.1	2.2	0.8
...	...	...	...	...
2988	2018-10-06	19.4	26.0	7.0
559	2012-02-11	-0.7	NaN	NaN
560	2012-02-12	0.4	NaN	NaN
561	2012-02-13	4.0	NaN	NaN
3183	2019-04-19	7.8	NaN	2.3

3653 rows × 4 columns

최대풍속(max_wind) 컬럼의 값 크기에 따라 내림차순으로 정렬

1

〈코드〉

```
df.sort_values(by='max_wind',
ascending=False)
```

〈실행결과〉

	date	temp	max_wind	mean_wind
2988	2018-10-06	19.4	26.0	7.0
3340	2019-09-23	15.0	25.8	11.0
1850	2015-08-25	20.1	25.3	14.9
3339	2019-09-22	15.7	23.1	11.9
1851	2015-08-26	17.4	22.6	8.1
...	...	...	...	...
1514	2014-09-23	20.7	2.0	1.0
559	2012-02-11	-0.7	NaN	NaN
560	2012-02-12	0.4	NaN	NaN
561	2012-02-13	4.0	NaN	NaN
3183	2019-04-19	7.8	NaN	2.3

3653 rows × 4 columns

3. 판다스 데이터 확인

- **df.value_counts( )**
  - 범주형 변수의 빈도분석 결과를 출력

빈도 분석 출력

```
1  bank=pd.read_csv('bank.csv')
2  bank['job'].value_counts()  # 내림차순
3  bank['job'].value_counts(ascending=True) # 오름차순
```

〈실행결과〉

```
management     1560
blue-collar    1499
technician     1206
admin.          834
services        661
retired         351
self-employed   256
entrepreneur    239
unemployed      223
housemaid       208
student         153
Name: job, dtype: int64
```

```
student         153
housemaid       208
unemployed      223
entrepreneur    239
self-employed   256
retired         351
services        661
admin.          834
technician     1206
blue-collar    1499
management     1560
Name: job, dtype: int 64
```

3. 판다스 데이터 확인

- **df.unique( )**
  - 해당 열의 고유값 확인

column의 고유값 출력

```
1   bank['job'].unique()
```

〈실행결과〉

```
array([ ' management ' ,  ' technician ' ,  ' blue-collar ' ,  ' retired ' ,  ' services ' ,
        ' admin. ' ,  ' entrepreneur ' ,  ' self-employed ' ,  ' unemployed ' ,  ' student ' ,
        nan,  ' housemaid ' ], dtype=object)
```

04

판다스 데이터 선택

4. 판다스 데이터 선택

데이터 선택

- Pandas DataFrame에서 데이터를 선택하는 다양한 방법 알아보기

	date	temp	max_wind	mean_wind
0	2020-07-01	16.8	19.7	8.7
1	2020-07-02	20.1	3.9	2.4
2	2020-07-03	19.2	4.8	3.1
3	2020-07-04	19.0	5.5	3.0
4	2020-07-05	19.8	6.2	3.9

df[1:3]

df['temp'] or df.temp

4. 판다스 데이터 선택

열 선택하기

- 열을 선택하면, 하나의 Series를 만든다.
- df['컬럼명'] = df.컬럼명

단일 컬럼 선택하기

〈코드〉

```
1  df['temp'] # df.temp
2  #type(df['temp'])
```

〈실행결과〉

```
0                                  28.7
1                                  25.2
2                                  22.1
3                                  25.3
4                                  27.2
                                   ...
3648                               22.1
3649                               21.9
3650                               21.6
2651                               22.9
3652                               25.7
Name: temp, Length: 3653, dtype: float64
```

여러 컬럼 선택

```
1  df[['date','temp']]
```

〈실행결과〉

	date	temp
0	2010-08-01	28.7
1	2010-08-02	25.2
2	2010-08-03	22.1
3	2010-08-04	25.3
4	2010-08-05	27.2
...	...	...

4. 판다스 데이터 선택

열 선택하기

- 열 이름을 사용하여 열을 선택하려는 경우
- 열 인덱스를 사용하여 추출하는 경우

여러 컬럼 선택 : loc()

```
1  df.loc[:,['date','temp']]
```

〈실행결과〉

	date	temp
0	2010-08-01	28.7
1	2010-08-02	25.2
2	2010-08-03	22.1
3	2010-08-04	25.3
4	2010-08-05	27.2
...	...	...

여러 컬럼 선택 : iloc()

```
1  df.iloc[:,[0,1]]
```

〈실행결과〉

	date	temp
0	2010-08-01	28.7
1	2010-08-02	25.2
2	2010-08-03	22.1
3	2010-08-04	25.3
4	2010-08-05	27.2
...	...	...

4. 판다스 데이터 선택

행 선택하기

- 인덱스 숫자를 사용하면 행을 슬라이스

특정 행 범위 선택

```
1  df[0:3]
```

〈실행결과〉

	date	temp	max_wind	mean_wind
0	2010-08-01	28.7	8.3	3.4
1	2010-08-02	25.2	8.7	3.8
2	2010-08-03	22.1	6.3	2.9

4. 판다스 데이터 선택

- 레이블로 선택하기(df.loc)

특정 날짜에 해당하는 열 선택

```
1  df.index=df['date']
2  df.loc['2010-08-01',['temp','mean_wind']]
```

〈실행결과〉

```
temp        28.7
mean_wind    3.4
Name: 2010-08-01, dtype: object
```

4. 판다스 데이터 선택

■ 위치로 선택하기(df.iloc)

- 정수를 입력하면 해당 행을 선택

특정 행 선택

```
1  df.iloc[3]
```

〈실행결과〉

```
date         2010-08-04
temp             28.3
max_wind          6.6
mean_wind         4.2
Name: 2010-08-04, dtype: object
```

특정 행과 열 선택 : 슬라이싱

```
1  df.iloc[1:3,0:2]
```

〈실행결과〉

	date	temp	max_wind	mean_wind
date				
2010-08-01	2010 08 01	28.7	8.3	3.4
2010-08-02	2010-08-02	25.2	8.7	3.8
2010-08-03	2010-08-03	22.1	6.3	2.9
2010-08-04	2010-08-04	25.3	6.6	4.2
2010-08-05	2010-08-05	27.2	9.1	5.6

	date	temp
date		
2010-08-02	2010-08-02	25.2
2010-08-03	2010-08-03	22.1

4. 판다스 데이터 선택

■ 위치로 선택하기(df.iloc)

- 정수를 입력하면 해당 행을 선택

특정 행 선택

```
1  df.iloc[3]
```

〈실행결과〉

```
date         2010-08-04
temp             28.3
max_wind          6.6
mean_wind         4.2
Name: 2010-08-04, dtype: object
```

특정 행과 열 선택 : 슬라이싱

```
1  df.iloc[1:3,0:2]
```

〈실행결과〉

	date	temp	max_wind	mean_wind
date				
2010-08-01	2010 08 01	28.7	8.3	3.4
2010-08-02	2010-08-02	25.2	8.7	3.8
2010-08-03	2010-08-03	22.1	6.3	2.9
2010-08-04	2010-08-04	25.3	6.6	4.2
2010-08-05	2010-08-05	27.2	9.1	5.6

	date	temp
date		
2010-08-02	2010-08-02	25.2
2010-08-03	2010-08-03	22.1

4. 판다스 데이터 선택

불 인덱싱

- 하나의 열의 값을 기준으로 데이터를 선택

조건에 맞는 데이터 추출

```
1  w=df['temp']>=30
2  df[w]
```

〈실행결과〉

date	date	temp	max_wind	mean_wind
2013-08-08	2013-08-08	31.3	7.8	4.6
2013-08-09	2013-08-09	30.6	9.9	6.4
2013-08-10	2013-08-10	30.6	7.4	3.8
2018-07-23	2018-07-23	30.5	6.5	1.6
2018-08-04	2018-08-04	30.3	5.8	3.0

조건에 맞는 데이터 추출

```
1  # 최고로 더웠던 날의 모든 정보를 출력하세요.
2  w=df['temp']==df['temp'].max()
3  df[w]
```

〈실행결과〉

date	date	temp	max_wind	mean_wind
2013-08-08	2013-08-08	31.3	7.8	4.6

조건이 2개 이상인 경우 추출

```
1  # 기온이 30도 이상이고 최대풍속이 9 이상인 데이터
2  w=(df['temp']>=30) & (df['max_wind']>=9)
3  df[w]
```

〈실행결과〉

date	date	temp	max_wind	mean_wind
2013-08-09	2013-08-09	30.6	9.9	6.4

05
판다스 결측데이터 처리

5. 판다스 결측데이터 처리

■ 결측데이터 확인하기

결측데이터 확인

```
1  # 최대풍속이 측정되지 않은 모든 데이터 출력
2  df[df['mean_wind'].isna()]
```

〈실행결과〉

	date	temp	max_wind	mean_wind
date				
2012-02-11	2012-02-11	-0.7	NaN	NaN
2012-02-12	2012-02-12	0.4	NaN	NaN
2012-02-13	2012-02-13	4.0	NaN	NaN
2015-03-22	2015-03-22	10.1	11.6	NaN
2015-04-01	2015-04-01	7.3	12.1	NaN

결측데이터 개수 확인 : 컬럼별 결측데이터 개수 확인

```
1
2  df.isnull().sum()
```

〈실행결과〉

```
date        0
temp        0
max_wind    4
mean_wind   6
dtype :  int64
```

5. 판다스 결측데이터 처리

■ 결측데이터 삭제하기

- 결측데이터를 다루는 가장 간단한 방법

표 5-1. 함수 설명 : DataFrame.dropna(axis, how, thresh, subset, inplace)

매개변수	설명
axis	• 축을 행 또는 열로 결정한다. • 0 또는 'index'이면 누락된 값이 포함 된 행을 삭제한다. • 1 또는 'columns'면 누락된 값이 포함 된 열을 삭제한다. • 기본적으로 값은 0으로 설정되어 있다.
how	• any는 null 값이 있는 경우 행 또는 열을 삭제한다. • all은 모든 값이 누락된 경우 행 또는 열을 삭제한다. • 기본적으로 any로 설정한다.
inplace	• True로 설정하면 호출자 DataFrame을 변경하는 부울 값이다. • 기본적으로 값은 False로 설정되어 있다.

결측데이터가 있는 행 삭제 후 확인

〈코드〉

```
1  df2=df.dropna()
2  df2.isnull().sum()
```

〈실행결과〉

```
date          0
temp          0
max_wind      0
mean_wind     0
dtype :  int64
```

5. 판다스 결측데이터 처리

■ 결측데이터 대체하기

결측데이터 대체 : 평균값으로 대체

```
1  df.fillna(df.mean(),inplace=True)
```

결측데이터 대체 후 확인

1

〈코드〉

```
df.isnull().sum()
```

〈실행결과〉

```
date          0
temp          0
max_wind      0
mean_wind     0
dtype :  int64
```

06
판다스 데이터 가공

6. 판다스 데이터 가공

데이터 가공

- 엑셀(.xlsx)파일을 불러와서 데이터프레임을 생성한다.

엑셀 파일 불러오기

```
1  dust=pd.read_excel('dust1.xlsx')
2  dust.head()
```

〈실행결과〉

	지역	망	측정소코드	측정소명	측정일시	SO2	CO	O3	NO2	PM10	PM25	주소
0	서울 송파구	도시대기	111273	송파구	2021040101	0.004	1.0	0.002	0.066	50	18	서울 송파구 백제고분로 236
1	서울 송파구	도시대기	111273	송파구	2021040102	0.004	0.8	0.002	0.058	48	20	서울 송파구 백제고분로 236
2	서울 송파구	도시대기	111273	송파구	2021040103	0.004	0.8	0.002	0.055	44	20	서울 송파구 백제고분로 236
3	서울 송파구	도시대기	111273	송파구	2021040104	0.003	0.8	0.002	0.055	40	20	서울 송파구 백제고분로 236
4	서울 송파구	도시대기	111273	송파구	2021040105	0.004	0.8	0.002	0.053	38	17	서울 송파구 백제고분로 236

6. 판다스 데이터 가공

■ 컬럼(변수) 삭제/생성하기

- 데이터 분석에 필요 없는 컬럼을 삭제하고 특정값으로
- 새로운 컬럼을 생성

컬럼 삭제

```
1  dust=dust.drop(['지역','망','측정소코드'], axis=1)
2  dust.head()
```

〈실행결과〉

	측정소명	측정일시	SO2	CO	O3	NO2	PM10	PM25	주소
0	송파구	2021040101	0.004	1.0	0.002	0.066	50	18	서울 송파구 백제고분로 236
1	송파구	2021040102	0.004	0.8	0.002	0.058	48	20	서울 송파구 백제고분로 236
2	송파구	2021040103	0.004	0.8	0.002	0.055	44	20	서울 송파구 백제고분로 236
3	송파구	2021040104	0.003	0.8	0.002	0.055	40	20	서울 송파구 백제고분로 236
4	송파구	2021040105	0.004	0.8	0.002	0.053	38	17	서울 송파구 백제고분로 236

컬럼 생성

```
1  dust['city']='서울'
2  dust.head()
```

〈실행결과〉

	측정소명	측정일시	SO2	CO	O3	NO2	PM10	PM25	주소	city
0	송파구	2021040101	0.004	1.0	0.002	0.066	50	18	서울 송파구 백제고분로 236	서울
1	송파구	2021040102	0.004	0.8	0.002	0.058	48	20	서울 송파구 백제고분로 236	서울
2	송파구	2021040103	0.004	0.8	0.002	0.055	44	20	서울 송파구 백제고분로 236	서울
3	송파구	2021040104	0.003	0.8	0.002	0.055	40	20	서울 송파구 백제고분로 236	서울
4	송파구	2021040105	0.004	0.8	0.002	0.053	38	17	서울 송파구 백제고분로 236	서울

6. 판다스 데이터 가공

컬럼 이름 변경하기

- DataFrame.columns = [‘새이름1’, ‘새이름2’, ...]
  - •전체 변수 이름을 재설정한다.
  - •변수명을 차례로 재설정한다.
- DataFrame.rename(columns = {‘기존이름’:‘새이름’}, inplace = True)
  - 원하는 변수 이름만 수정한다.
  - 딕셔너리 구조로 정의한다.

컬럼 이름 변경

```
1  dust.rename(columns={'측정소명':'name',
2                       '측정일시':'date',
3                       '주소':'addr'}, inplace=True)
4  dust.columns
```

〈실행결과〉

```
Index(['name', 'date', 'SO2', 'CO', 'O3', 'NO2', 'PM10', 'PM25', 'addr', 'city'], dtype='object'
```

6. 판다스 데이터 가공

■ 데이터 형 변환

- 숫자 형식은 문자 형식으로 변환

데이터형으로 확인

〈코드〉

```
1 dust.dtypes
```

〈실행결과〉

```
name    object
date     int64
SO2    float64
CO     float64
O3     float64
NO2    float64
PM10     int64
PM25     int64
addr    object
city    object
dtype: object
```

숫자 int형을 문자열형으로 변환

〈코드〉

```
1 dust['date']=dust['date'].astype(str)
2 dust['date']=dust['date'].str.slice(0,8)
3 dust.dtypes
```

〈실행결과〉

```
name    object
date    object
SO2    float64
CO     float64
O3     float64
NO2    float64
PM10     int64
PM25     int64
addr    object
city    object
dtype: object
```

6. 판다스 데이터 가공

■ 데이터 형 변환

문자열형을 날짜형으로 변환

〈코드〉

```
1  dust['date']=pd.to_datetime(dust['date'])
2  dust.dtypes
```

〈실행결과〉

```
name              object
date      datetime64[ns]
SO2              float64
CO               float64
O3               float64
NO2              float64
PM10               int64
PM25               int64
addr              object
city              object
dtype : object
```

날짜 형식 활용 : Series.dt. 형식

```
1  dust['year']=dust['date'].dt.year
2  dust['month']=dust['date'].dt.month
3  dust['day']=dust['date'].dt.day
4  dust.head()
```

〈실행결과〉

	name	date	SO2	CO	O3	NO2	PM10	PM25	addr	city	year	month	day
0	송파구	2021-04-01	0.004	1.0	0.002	0.066	50	18	서울 송파구 백제고분로 236	서울	2021	4	1
1	송파구	2021-04-01	0.004	0.8	0.002	0.058	48	20	서울 송파구 백제고분로 236	서울	2021	4	1
2	송파구	2021-04-01	0.004	0.8	0.002	0.055	44	20	서울 송파구 백제고분로 236	서울	2021	4	1
3	송파구	2021-04-01	0.003	0.8	0.002	0.055	40	20	서울 송파구 백제고분로 236	서울	2021	4	1
4	송파구	2021-04-01	0.004	0.8	0.002	0.053	38	17	서울 송파구 백제고분로 236	서울	2021	4	1

6. 판다스 데이터 가공

■ 데이터 병합하기

	국적코드	성별	입국객수	증가수
0	A01	남성	125000	8000
1	A01	여성	130000	10000
2	A05	남성	300	10
3	A05	여성	200	50
4	A06	남성	158912	24486
5	A06	여성	325000	63466

+

	국적코드	국적명
0	A01	필리핀
1	A01	일본
2	A05	미국
3	A05	중국
4	A06	호주
5	A06	베트남

→

	국적코드	성별	입국객수	증가수	국적명
0	A01	남성	125000	8000	필리핀
1	A01	여성	130000	10000	필리핀
2	A05	남성	300	10	호주
3	A05	여성	200	50	호주
4	A06	남성	158912	24486	베트남
5	A06	여성	325000	63466	베트남

▪ 함수 형식

- pd.merge(df_left, df_right, how = 'inner', on = None)

- 아무런 옵션도 적용하지 않으면, on = None이므로 두 데이터의 공통 열 이름(id)을 기준으로
- Inner(교집합)를 조인
- Outer 옵션을 적용하여 id를 기준으로 합치는데, 어느 한쪽에라도 없는 데이터가 있는 경우
- NaN 값이 지정
- 왼쪽에 입력한 데이터프레임 기준(how = 'left')으로, 각각의 key 값에 해당하는 열을 지정
- 오른쪽에 입력한 데이터프레임 기준(how = 'right')으로, 각각의 key 값에 해당하는 열을 지정

6. 판다스 데이터 가공

데이터 병합하기

병합할 원본 데이터 확인

```
1
2
3
4
```

```
s1=pd.read_excel('nation.xlsx')
s2=pd.read_excel('code.xlsx')
```

〈실행결과〉

	국적코드	성별	입국객수	증가수
0	A01	남성	125000	8000
1	A01	여성	130000	10000
2	A05	남성	300	10
3	A05	여성	200	50
4	A06	남성	158912	24486
5	A06	여성	325000	63466

	국적코드	국적명
0	A01	필리핀
1	A02	일본
2	A03	미국
3	A04	중국
4	A05	호주
5	A06	베트남
6	A07	스위스
7	A99	기타

데이터 병합 : 공통 컬럼을 기준

```
1
```

```
pd.merge(s1,s2,on='국적코드')
```

〈실행결과〉

	국적코드	성별	입국객수	증가수	국적명
0	A01	남성	125000	8000	필리핀
1	A01	여성	130000	10000	필리핀
2	A05	남성	300	10	호주
3	A05	여성	200	50	호주
4	A06	남성	158912	24486	베트남
5	A06	여성	325000	63466	베트남

07
판다스 데이터 그룹핑

7. 판다스 데이터 그룹핑

데이터 그룹핑

- 데이터를 특정한 값에 기반에 묶는 기능
- 통계량을 요약

한 열을 기준으로 그룹화

```
1  s1=pd.read_excel('nation.xlsx')
2  s1.groupby('국적코드').sum()
```

〈실행결과〉

국적코드	입국객수	증가수
A01	255000	18000
A05	500	60
A06	483912	87952

여러 열을 기준으로 그룹화

```
1  s1=pd.read_excel(filename)
2  s1.groupby(['국적코드','성별']).sum()
```

〈실행결과〉

국적코드	성별	입국객수	증가수
A01	남성	125000	8000
	여성	130000	10000
A05	남성	300	10
	여성	200	50
A06	남성	158912	24486
	여성	325000	63466