데이터시각화이해와실습

Lecture 10. 판다스 활용

외부 라이브러리 pandas

1-1. python(for, if, 자료형) --> numpy | pandas --> matplotlib, seaborn

과거 기록 pandas와 seaborn은 호환성이 높다

미래

--> 모형화(모델링)

AI

- scikit-learn(기계학습 라이브러리)
- TensorFlow(딥러닝 라이브러리)
- Pytorch(딥러닝 라이브러리)

동덕여자대학교
데이터사이언스 전공
권 범

목차

❖ 01. 데이터 처리가 쉬운 판다스

❖ 02. 붓꽃 데이터 분석

❖ 03. 타이타닉 데이터 분석

시각 지능: 이미지 or 동영상 --> dlib, KerasCV, OpenCV

언어 지능: 텍스트 --> 자연어 처리(NLP)

01. 데이터 처리가 쉬운 판다스

02. 붓꽃 데이터 분석
03. 타이타닉 데이터 분석

01. 데이터 처리가 쉬운 판다스

❖ 판다스 개념 및 특징 (1/3)

- 판다스(Pandas)는 고수준의 자료구조와 빠르고 쉬운 데이터 분석 도구를 제공하는 파이썬의 라이브러리임
- 넘파이(NumPy) 기반에서 개발되어 넘파이를 사용하는 애플리케이션(Application)에서 쉽게 사용 가능하고, 여러 형태의 데이터를 분석하고 정리할 때 유용하게 사용됨

Q. 왜 둘 함께 사용?

A.

numpy는 numerical 벡터, 행렬 --> 배열 핸들링에 손쉬운 메서드 제공

배열의 결측치, 이상치 처리, 서로 다른 파일(csv, xlsx)의 데이터 합병(merge) --> pandas

01. 데이터 처리가 쉬운 판다스

❖ 판다스 개념 및 특징 (2/3)

- 온전히 통계 분석을 위해 고안된 R과 다르게, 파이썬은 일반적인(General Purpose) 프로그래밍 언어임
- 따라서 파이썬으로 데이터 분석을 하기 위해서는 여러 가지 라이브러리를 활용할 수밖에 없음
- R의 데이터프레임(data.frame) 데이터 타입을 참고하여 만든 것이 바로 판다스 데이터프레임(DataFrame)임

01. 데이터 처리가 쉬운 판다스

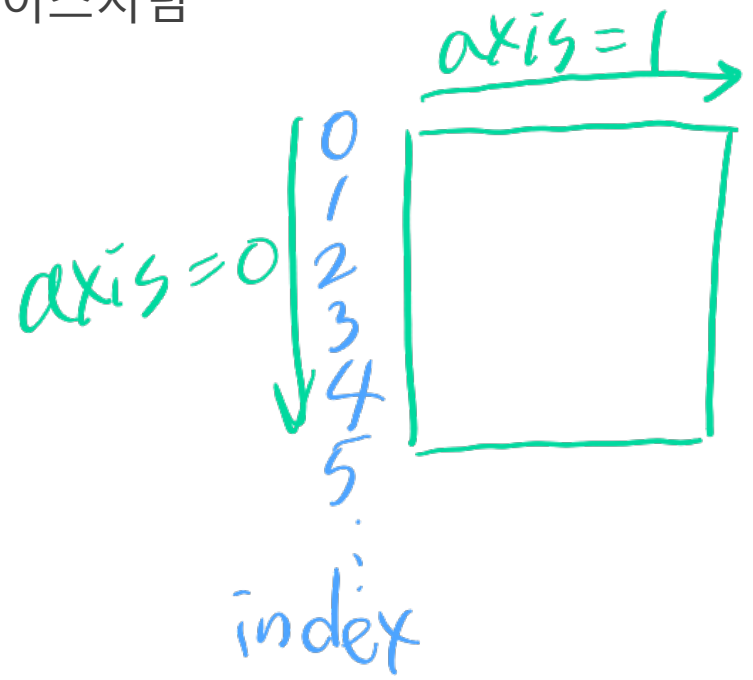
❖ 판다스 개념 및 특징 (3/3)

- 데이터프레임을 자유롭게 가공하는 것은 데이터 과학자에게 중요한 스킬임
- 물론 판다스의 문법을 외우지 않고, 필요할 때마다 책이나 웹에서 찾아가면서 해도 좋지만, 자주 사용하는 조작법을 외우고 있다면 데이터 핸들링을 빠르게 작업할 수 있음

01. 데이터 처리가 쉬운 판다스

❖ 판다스의 주요 특징 axis

- 자동적/명시적으로 축의 이름에 따라 데이터를 정렬할 수 있는 데이터 구조
- 잘못 정렬된 데이터에 의한 오류를 방지하고, 다양한 방식으로 색인된 데이터를 다룰 수 있는 기능
- 통합된 시계열 기능 datetime
- 시계열 데이터와 비시계열 데이터를 함께 다룰 수 있는 통합 자료구조
- 누락된 데이터를 유연하게 처리할 수 있는 기능 결측값, 결측치 --> fillna(), dropna()
- SQL(Structured Query Language) 같은 일반 데이터베이스처럼
  데이터를 합치고, 관계 연산을 수행하는 기능



01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성

- 판다스의 기본 객체인 Series와 DataFrame을 생성하는 방법에 대해 알아보자
  - ◆ Series(시리즈): 레이블을 갖는 1차원 배열
  - ◆ DataFrame(데이터프레임): 레이블을 갖는 행과 열을 갖는 2차원 배열

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ① Series와 DataFrame

- Series는 다양한 자료형을 담을 수 있는 1차원의 배열임
- 즉, Series는 엑셀 문서의 하나의 열(Column)과 같으며, Index라는 레이블을 가짐
- DataFrame은 행과 열을 갖는 2차원의 자료형으로, 여러 개의 Series가 모이면 DataFrame을 구성함
- DataFrame은 Index와 Key, 그리고 Value로 구성됨

	Brand
0	Samsung
1	LG
2	Apple
3	Apple

Series

+

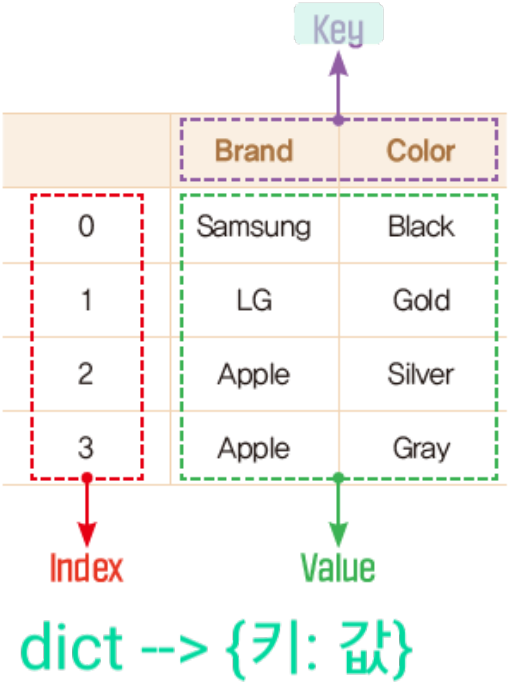
	Color
0	Black
1	Gold
2	Silver
3	Gray

Series

=

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

DataFrame



[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ② Series 만들기 (1/3)

- Series는 동일한 유형의 데이터를 저장하는 1차원 배열임
- Series( ) 함수를 이용하면, 리스트를 쉽게 Series로 만들 수 있음
- 여러 개의 정수 값으로 구성된 리스트를 Series로 만들어 보자

```
1 import pandas as pd
2
3 s = pd.Series([95, 90, 85, 90, 95])
4 s
```

실행결과

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ② Series 만들기 (2/3)

- 이번에는 여러 개의 실수 값으로 구성된 리스트를 Series로 만들어 보자

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

실행결과

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

0, 1, 2, … 와 같은 인덱스가 아닌,
사용자가 원하는 인덱스를 지정하기 위해서는 어떻게 해야 할까?

❖ 판다스 객체 생성: ② Series 만들기 (3/3)

- Series( ) 함수의 index 매개변수를 통해서, 원하는 인덱스를 설정할 수 있음

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

실행결과

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ③ DataFrame 만들기 (1/2)

- Series를 모아서 하나의 DataFrame을 만들 수 있음
- 넘파이 배열로부터 DataFrame을 만들어 보자

```
1 import numpy as np
2
3 arr = np.arange(0, 9, 1).reshape(3, 3)
4 print(arr)
5 print()
6 df = pd.DataFrame(arr)
7 print(df)
```

실행결과

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

   0  1  2
0  0  1  2
1  3  4  5
2  6  7  8
```

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ③ DataFrame 만들기 (2/2)

- 이번에는 딕셔너리 구조를 이용하여 DataFrame을 만들어 보자

```
1 df = pd.DataFrame({"name": ["이상해씨", "파이리", "꼬부기"],
2                    "number": [1, 4, 7],
3                    "type": ["Grass", "Fire", "Water"]})
4 df
```

실행결과

	name	number	type
0	이상해씨	1	Grass
1	파이리	4	Fire
2	꼬부기	7	Water

**딕셔너리의 키가 그대로 DataFrame의 키가 되며,
그 결과로 만들어지는 DataFrame의 열은 서로 다른 자료형을 가짐**

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ④ csv 활용 (1/2)

- 판다스가 제공하는 read_csv( ) 함수를 이용하여 데이터를 DataFrame으로 불러올 수 있음

```
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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 객체 생성: ④ csv 활용 (2/2)

- index_col은 인덱스로 사용할 열을 지정함

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

실행결과

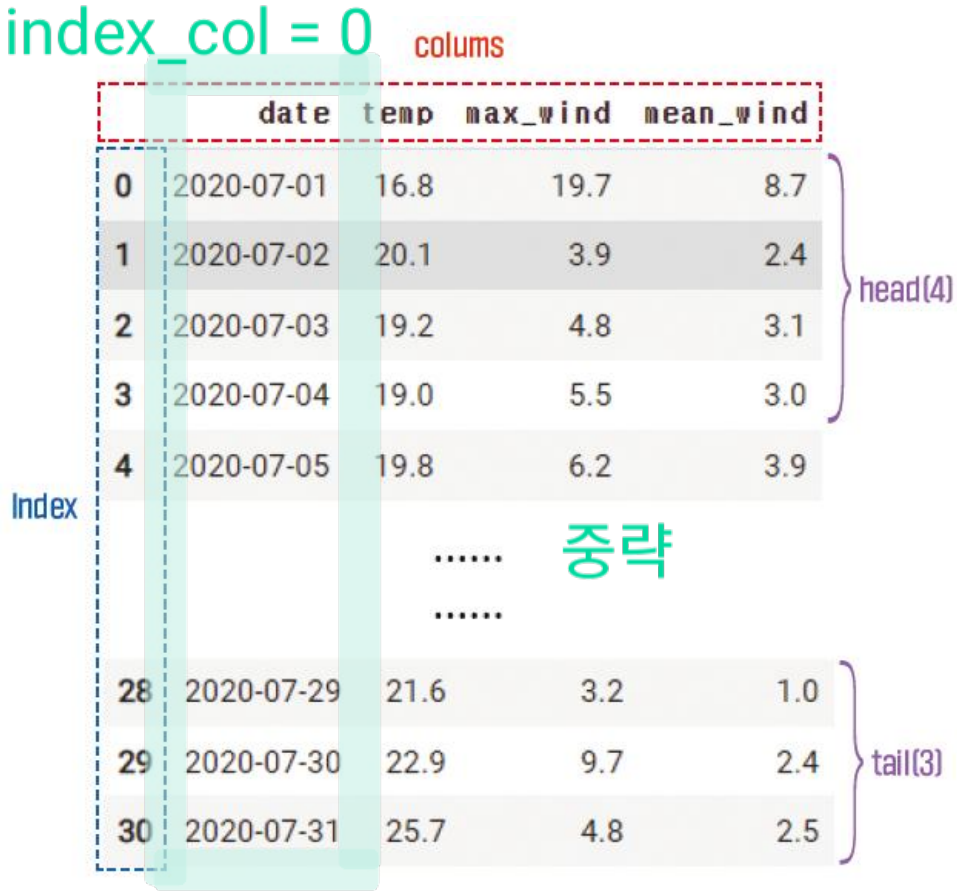
Brand	Color	price
Samsung	Black	100
LG	Gold	70
Apple	Silver	150
Apple	Gray	120

index_col=0을 지정해서,
첫 번째 열이 인덱스가 되었음

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인

- 판다스에는 데이터 분석에 유용한 여러 가지 함수(메소드)와 속성이 미리 정의되어 있음
- 판다스 객체의 데이터를 확인하는 다양한 방법을 알아보자



	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

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ① DataFrame 만들기

- csv 파일을 불러와서 데이터프레임을 생성하자

```
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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ② df.shape

- shape 속성을 이용하면 데이터의 (행, 열) 크기를 확인할 수 있음

```
1 df.shape
```

실행결과

```
(3653, 4)
```

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ③ df.info( )

- info( ) 함수를 이용하면 데이터에 대한 전반적인 정보를 출력할 수 있음
- df를 구성하는 행과 열의 크기, 컬럼명, 컬럼을 구성하는 값의 자료형 등이 출력됨

```
1 df.info()
```

실행결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3653 entries, 0 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
```

✓ 데이터 개수: 3653 entries, 행 인덱스 번호: 0 to 3652
✓ 열 변수 형식(Dtype): 문자열 object, 실수 float64
✓ 결측치 개수: max_wind, mean_wind 변수에 결측치가 있음

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ④ df.head( )

- 데이터를 잘 불러왔는지 확인하기 위해, DataFrame의 앞 부분을 확인해 보자

```
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

head( ) 함수는 데이터프레임의 상위 5개의 행을 출력함

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ⑤ df.tail( )

- 이번에는 DataFrame의 마지막 부분을 확인해 보자

```
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

소괄호( ) 안에 원하는 숫자를 입력하면, 해당 숫자만큼의 행을 출력할 수 있음

tail( ) 함수는 데이터프레임의 하위 5개의 행을 출력함

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ⑥ df.index, df.columns

- 인덱스(행 이름)와 열의 레이블(컬럼 이름)을 출력하려면, index와 columns 속성을 사용하면 됨

```
1 print(df.index)
2 print(df.columns)
```

실행결과

```
RangeIndex(start=0, stop=3653, step=1)
Index(['date', 'temp', 'max_wind', 'mean_wind'], dtype='object')
```

✓ index 속성은 데이터프레임의 인덱스(행 이름)을 반환함
✓ 여기서는 따로 인덱스를 지정하지 않았기 때문에 0부터 시작하는 인덱스가 부여되었음
✓ columns 속성을 이용하면 해당 데이터프레임을 구성하는 컬럼명을 확인할 수 있음
✓ 이 기능은 컬럼명을 변경할 때도 유용하게 사용됨

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ⑦ df.describe( )

- describe( ) 함수는 데이터의 컬럼별 기초 통계량을 출력함
- 물론, mean( ), max( ), median( ) 등 개별 함수를 사용하여, 직접 통계량을 계산할 수도 있음

```
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

예를 들어, df[“temp”].mean( ) 함수를 호출하면, 기온(temp) 컬럼의 평균값을 출력할 수 있음

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ⑧ df.sort_values( )

- sort_values( ) 함수를 사용하면, 데이터의 크기순으로 정렬할 수 있음
- 두 번째 열 즉, 최대풍속(max_wind)의 값의 크기에 따라, DataFrame을 정렬해 보자

```
1 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

sort_values( ) 함수 형식

pandas.DataFrame.sort_values(**by**, **ascending**, **inplace**)

- ✓ **by**: 문자열 또는 리스트로 정렬 기준 컬럼을 지정함
- ✓ **ascending**: 기본값이 True로, 오름차순으로 정렬되며, 내림차순을 원할 경우 False로 지정하면 됨
- ✓ **inplace**: 정렬 결과를, 작업 중인 데이터프레임에 저장하려면 True로 지정함

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ⑨ df.value_counts( ) <---> .describe() : 숫자형 컬럼 --> 기초 통계량

- 범주형 변수의 빈도수를 출력함

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

실행결과

```
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: int64
```

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 확인: ⑩ df.unique( )

- unique( ) 함수를 이용하면, 해당 열의 고유 값을 확인할 수 있음

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

실행결과

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ① 열 선택하기 (1/5)

- DataFrame의 하나의 열을 선택하면, 하나의 Series를 만듦
- 하나의 열을 선택하는 두 가지 방법
  - ◆ df["컬럼명"] <-- 권장
  - ◆ df.컬럼명 <-- 권장 X : 변수명으로 사용하면 안 되는 조건(숫자 시작, 띄워쓰기) 시 사용 불가
- 두 번째 방법의 경우, 열 이름이 숫자로 시작하지 않고 공백이나 특수 문자 등을 포함하지 않는 등의 조건을 만족해야 함
- 또한, iloc( )과 loc( ) 메소드를 사용하면 여러 열을 선택할 수 있음

	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

DataFrame에서 데이터를 선택하는 다양한 방법을 알아보자

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ① 열 선택하기 (2/5)

- 단일 컬럼을 선택해 보자

```
1 df[“temp”]        # df.temp
2 # type(df[“temp”]) --> Series
```

df["temp"].shape --> (3653,)

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
3651    22.9
3652    25.7
Name: temp, Length: 3653, dtype: float64
```

df[["temp"]] --> DataFrame

df[["temp"]].shape --> (3653, 1)

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ① 열 선택하기 (3/5)

- 추출할 열의 이름을 리스트에 저장한 다음, [ ]에 전달하면 DataFrame에서 여러 열을 선택할 수 있음

```
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
...	...	...
3648	2020-07-27	22.1
3649	2020-07-28	21.9
3650	2020-07-29	21.6
3651	2020-07-30	22.9
3652	2020-07-31	25.7

3653 rows × 2 columns

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ① 열 선택하기 (4/5)

- loc( ) 메소드를 사용하면 컬럼명을 통해 열을 선택할 수 있음

```
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
...	...	...
3648	2020-07-27	22.1
3649	2020-07-28	21.9
3650	2020-07-29	21.6
3651	2020-07-30	22.9
3652	2020-07-31	25.7

3653 rows × 2 columns

❖ 판다스 데이터 선택: ① 열 선택하기 (5/5)

- 열 인덱스를 사용해 열을 선택하려면, iloc( ) 메소드를 이용하면 됨

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

integer location

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
...	...	...
3648	2020-07-27	22.1
3649	2020-07-28	21.9
3650	2020-07-29	21.6
3651	2020-07-30	22.9
3652	2020-07-31	25.7

3653 rows × 2 columns

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ② 행 선택하기

- 특정 행의 범위를 슬라이싱(Slicing)을 통해 선택할 수 있음

```
1 df[0:3] <-- 0, 1, 2 (3은 미포함)
```

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

01. 데이터 처리가 쉬운 판다스

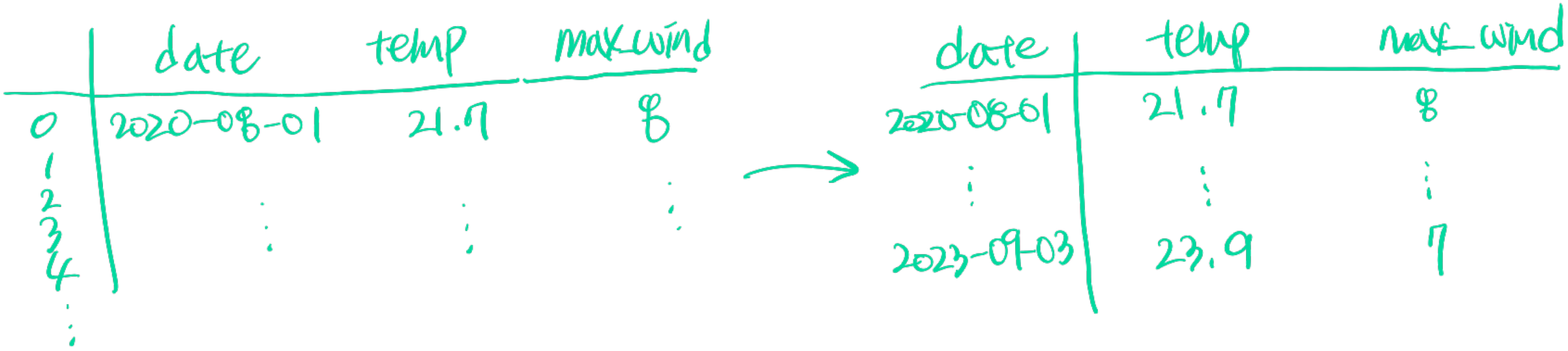
❖ 판다스 데이터 선택: ③ 레이블로 선택하기 – df.loc[ ]

- 날짜(date)를 데이터프레임의 인덱스로 지정하고,
  원하는 날짜의 기온(temp), 평균풍속(mean_wind) 값을 추출해 보자

```
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
```



01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ④ 위치로 선택하기 – df.iloc[ ]

- iloc[n]과 같이 대괄호 [ ] 안에 인덱스 n을 입력하면, 원하는 행을 선택할 수 있음

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

실행결과

```
date          2010-08-04
temp                25.3
max_wind             6.6
mean_wind            4.2
Name: 2010-08-04, dtype: object
```

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

특정 행, 특정 열을 동시에 선택하려면 어떻게 해야 할까?

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ④ 위치로 선택하기 – df.iloc[ ]

- 슬라이싱을 이용하면 특정 행, 특정 열을 선택할 수 있음

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

실행결과

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

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ⑤ 부울린(Boolean) 인덱싱 (1/3)

True/False <-- 브로드 캐스팅

- 조건에 맞는 데이터만을 선택할 수 있음

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

마스크(Mask), 필터(Filter)

실행결과

	date	temp	max_wind	mean_wind
date				
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

기온(temp) 값이 30보다 크거나 같은 행만 출력되는 것을 확인할 수 있음

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ⑤ 부울린(Boolean) 인덱싱 (2/3)

- 가장 더웠던 날의 모든 정보를 출력해 보자

```
1 w = df[“temp”] == df[“temp”].max()
2 df[w]
```

실행결과

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 선택: ⑤ 부울린(Boolean) 인덱싱 (3/3)

- 기온이 30도 이상이고, 최대풍속이 9 이상인 데이터를 모두 추출해 보자

```
1 w = (df[“temp”] >= 30) & (df[“max_wind”] > 9)
2 df[w]
```

실행결과

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

조건이 2개 이상인 경우에는
각 조건을 소괄호 ( )로 묶어서 표현하자!

01. 데이터 처리가 쉬운 판다스

❖ 판다스 결측 데이터 처리: ① 결측 데이터 확인하기 (1/2)

- 판다스는 누락된 데이터를 표시할 때, NaN(Not a Number)으로 표기하며, 연산에는 포함되지 않음
- DataFrame에 누락된 데이터를 확인하고, 이를 처리하는 방법에 대해 알아보자
- isna( ) 메소드를 사용하면, 결측 데이터를 확인할 수 있음

```
1 df[df["max_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
2019-04-19	2019-04-19	7.8	NaN	2.3

isna( ) 함수는 값이 NaN일 때 True, 그렇지 않으면 False 값을 반환함

isna( ) 함수와 isnull( ) 함수는 동일한 기능을 수행함

01. 데이터 처리가 쉬운 판다스

❖ 판다스 결측 데이터 처리: ① 결측 데이터 확인하기 (2/2)

- isna( ) 함수와 sum( ) 함수를 활용하면, 컬럼별 결측 데이터 개수를 확인할 수 있음

```
1 df.isna().sum()
```

실행결과

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

.info()를 사용해도 됨

01. 데이터 처리가 쉬운 판다스

❖ 판다스 결측 데이터 처리: ② 결측 데이터 삭제하기 (1/2)

- 결측 데이터를 다루는 가장 간단한 방법은 결측 데이터를 가진 행이나 열을 삭제하는 것임
- 판다스의 dropna( ) 함수를 이용하면 결측 데이터를 삭제할 수 있음

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

실행결과

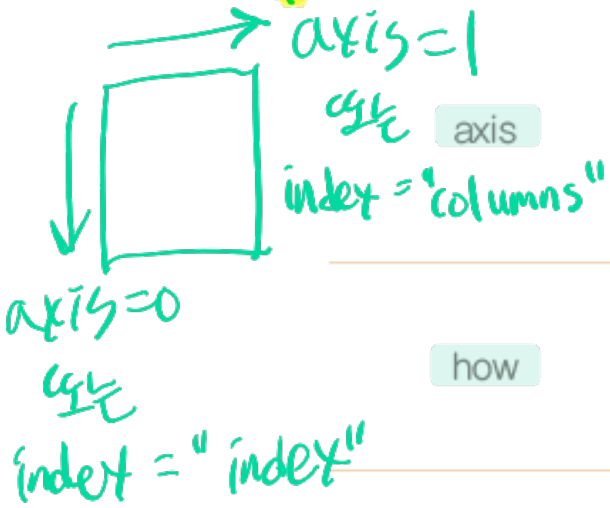
```
date          0
temp          0
max_wind      0
mean_wind     0
dtype: int64
```

✓ 행 데이터 중 어느 한 변수에도 결측치가 있는 경우, 해당 행은 삭제됨
✓ 결측치가 제거된 데이터프레임을 사용하기 위해, 다른 이름으로 저장함

01. 데이터 처리가 쉬운 판다스

❖ 판다스 결측 데이터 처리: ② 결측 데이터 삭제하기 (2/2)

- dropna( ) 함수의 매개변수에 대해서 알아보자



함수 형식: pandas.DataFrame.dropna(axis, how, 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.

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 처리가 쉬운 판다스

❖ 판다스 결측 데이터 처리: ③ 결측 데이터 대체하기

- fillna( ) 함수는 결측 데이터를 특정 값으로 채움

```
1 df[“temp”] = df[“temp”].fillna(df[“temp”].mean())
2 df[“max_wind”] = df[“max_wind”].fillna(df[“max_wind”].mean())
3 df[“mean_wind”] = df[“mean_wind”].fillna(df[“mean_wind”].mean())
4 df.isna().sum()
```

실행결과

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ① 엑셀 파일 불러오기

- dust.xlsx 파일을 불러와서 데이터프레임을 생성하자

```
1 dust = pd.read_excel("dust.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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ② 컬럼 삭제하기

- 데이터 분석에 필요 없는 '지역', '망', '측정소코드' 컬럼을 삭제하자
- axis의 기본값은 axis=0으로, 행이 삭제됨
- 컬럼을 삭제하기 위해서는 axis=1을 사용해야 함

columns

```
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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ③ 컬럼 생성하기

- 도시(city) 컬럼을 새롭게 생성하자

```
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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ④ 컬럼 이름 변경하기 (1/2)

- 한글로 작성된 컬럼명을 영어로 변경하자

```
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')
```

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ④ 컬럼 이름 변경하기 (2/2)

- 컬럼 이름을 변경하는 두 가지 방법
- pandas.DataFrame.columns = [“새이름1”, “새이름2”, …]
  - ◆ 전체 변수 이름을 재설정함
  - ◆ 변수명을 차례로 재설정함
  - ◆ 변수가 많은 경우 적절하지 않음
- pandas.DataFrame.rename(columns={“기존이름”: “새이름”}, inplace=True) <-- 권장
  - ◆ 원하는 변수의 이름만 수정함
  - ◆ 딕셔너리 구조로 정의함
  - ◆ 즉, 이전 열 이름을 키로 지정하고, 새 이름을 값으로 지정함

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑤ 데이터 형 변환 (1/4)

- dust 데이터프레임의 컬럼별 데이터 자료형을 확인해 보자

```
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
```

String

date 컬럼이 숫자형(int64)으로 저장되어 있음.
head( ) 함수로 확인해 보면, 년도, 월, 일, 시간이 결합된 형태임!

	name	date	SO2	CO	O3	NO2	PM10	PM25	addr	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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑤ 데이터 형 변환 (2/4)

- 숫자 형식으로 구성된 date 컬럼을 날짜형으로 변환해 보자
- 우선, 숫자 형식을 문자 형식으로 변환하고, 날짜 형식 8자리(년도: 4자리, 월: 2자리, 일: 2자리)에 적합하게 str.slice( )를 사용하여 첫 8자리를 슬라이싱하자

```
1 dust["date"] = dust["date"].astype(str)
2 dust["date"] = dust["date"].str.slice(0, 8)
3 print(dust["date"].dtypes)
4 dust.head(3)
```

문자열로 바꾸기

실행결과

```
object
```

	name	date	SO2	CO	O3	NO2	PM10	PM25	addr	city
0	송파구	20210401	0.004	1.0	0.002	0.066	50	18	서울 송파구 백제고분로 236	서울
1	송파구	20210401	0.004	0.8	0.002	0.058	48	20	서울 송파구 백제고분로 236	서울
2	송파구	20210401	0.004	0.8	0.002	0.055	44	20	서울 송파구 백제고분로 236	서울

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑤ 데이터 형 변환 (3/4)

- 이제 to_datetime( ) 함수를 활용해 date 컬럼을 날짜형으로 변환해 보자

```
1 dust["date"] = pd.to_datetime(dust["date"])
2 print(dust["date"].dtypes)
3 dust.head(3)
```

실행결과

```
datetime64[ns]
```

	name	date	SO2	CO	O3	NO2	PM10	PM25	addr	city
0	송파구	2021-04-01	0.004	1.0	0.002	0.066	50	18	서울 송파구 백제고분로 236	서울
1	송파구	2021-04-01	0.004	0.8	0.002	0.058	48	20	서울 송파구 백제고분로 236	서울
2	송파구	2021-04-01	0.004	0.8	0.002	0.055	44	20	서울 송파구 백제고분로 236	서울

날짜 자료형은 두 데이터프레임을 병합할 때, 고유값으로 활용되는 경우가 종종 있음

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑤ 데이터 형 변환 (4/4)

- pandas.Series.dt.날짜형식(year, month, day)을 사용하여 년도, 월, 일 컬럼을 생성해 보자

```
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

날짜 자료형은 두 데이터프레임을 병합할 때, 고유값으로 활용되는 경우가 종종 있음

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑥ 데이터 병합하기 (1/5)

- 데이터를 분석할 때, 데이터가 서로 다른 파일에 저장되어 있을 수 있음
- 각 파일에서 읽은 데이터프레임을 병합할 수 있다면, 데이터 분석이 용이해 질 것임

어떻게 두 데이터프레임을 병합할 수 있을까? merge()

	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

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑥ 데이터 병합하기 (2/5)

- 판다스의 merge( ) 함수를 이용하면 데이터프레임을 병합할 수 있음
- merge( ) 함수는 각 데이터에 존재하는 고유값(Key)을 기준으로 두 데이터프레임을 병합함

pandas.merge( ) 함수 형식

pandas.merge(**left**, **right**, **how**, **on**)

기준

- ✓ **left**: 병합할 DataFrame 또는 Series를 전달함
- ✓ **right**: 병합할 DataFrame 또는 Series를 전달함
- ✓ **how**: 병합하는 방법을 지정하며, 기본값은 "inner"임.
  그 밖의 병합 방법에는 "left", "right", "outer", "cross"가 있음
- ✓ **on**: 병합에 사용할 열 또는 인덱스 이름을 지정함.
  만약 on을 None으로 지정하고 인덱스에서 병합하지 않는 경우,
  두 데이터프레임에서 컬럼의 이름을 기준으로 교집합이 사용됨

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑥ 데이터 병합하기 (3/5)

- nation.xlsx 파일을 불러와서 데이터프레임 s1을 생성하자

```
1 s1 = pd.read_excel("nation.xlsx")
2 s1
```

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑥ 데이터 병합하기 (4/5)

- code.xlsx 파일을 불러와서 데이터프레임 s2를 생성하자

```
1 s2 = pd.read_excel("code.xlsx")
2 s2
```

실행결과

	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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 가공: ⑥ 데이터 병합하기 (5/5)

- 공통 컬럼인 '국적코드'를 기준으로 두 데이터프레임을 병합해 보자

```
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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 그룹핑 (1/3)

- 판다스의 groupby( ) 함수를 사용하면, 지정한 열을 기준으로 데이터를 묶는 것이 가능함
- 또한, mean( ), std( ), var( ), max( ), min( ), mode( ) 함수 등, 통계량과 관련된 함수와 함께 활용됨
- 참고로, mode( ) 함수는 최빈값을 반환해 주는 함수임

groupby(무엇을 수단으로)
["대분류", "중분류", "소분류"]

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 그룹핑 (2/3)

- groupby( ) 함수를 사용해서 '국적코드' 열을 기준으로 그룹화하고, sum( ) 함수를 적용해 그룹별 합계를 계산해 보자

```
1 s1.groupby("국적코드").sum()
```

실행결과

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

01. 데이터 처리가 쉬운 판다스

❖ 판다스 데이터 그룹핑 (3/3)

- 이번에는 '국적코드', '성별' 열을 기준으로 그룹화하고, sum( ) 함수를 적용해 그룹별 합계를 계산해 보자

```
1 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

여러 열을 그룹으로 묶을 경우, 계층적인 인덱스를 구성함

02. Iris 붓꽃 데이터 분석

01. 데이터 처리가 쉬운 판다스
03. 타이타닉 데이터 분석

02. 붓꽃 데이터 분석

❖ 데이터 설명

50개 50개 50개

- 아이리스(붓꽃) 데이터에는 3가지 종(Setosa, Versicolor, Virginica)이 각각의 특성에 맞게 분류되어 있음
- 꽃잎의 꽃받침(Sepal), 꽃잎(Petal) 부분의 너비와 길이 등을 측정한 데이터이며, 150개의 샘플로 구성되어 있음

붓꽃의 3가지 종

Iris Versicolor | Iris Setosa | Iris Virginica

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

02. 붓꽃 데이터 분석

❖ 필드의 이해

- 데이터 세트에 포함된 5개의 변수에 대해 살펴보자

변수명	변수 설명
Sepal Length	꽃받침의 길이 정보
Sepal Width	꽃받침의 너비 정보
Petal Length	꽃잎의 길이 정보
Petal Width	꽃잎의 너비 정보
Species	꽃의 종류 정보, Setosa/Versicolor/Virginica 3종류로 구분

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

02. 붓꽃 데이터 분석

❖ 붓꽃 데이터 읽어오기

- iris.csv 파일로부터 붓꽃 데이터를 읽고, 처음 5줄의 데이터를 출력해 보자

```
1 import pandas as pd
2
3 iris = pd.read_csv("iris.csv")
4 iris.head()
```

실행결과

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

02. 붓꽃 데이터 분석

❖ 붓꽃 데이터의 기본 정보 출력

- info( ) 함수를 이용해, 데이터프레임의 기본 정보를 확인해 보자

```
1 iris.info()
```

실행결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   SepalLength  150 non-null    float64
 1   SepalWidth   150 non-null    float64
 2   PetalLength  150 non-null    float64
 3   PetalWidth   150 non-null    float64
 4   Species      150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

02. 붓꽃 데이터 분석

❖ 붓꽃 데이터의 기초 통계량 출력

- describe( ) 함수를 이용해, 붓꽃 데이터의 기초 통계량을 확인해 보자

```
1 iris.describe()
```

실행결과

	SepalLength	SepalWidth	PetalLength	PetalWidth
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

02. 붓꽃 데이터 분석

❖ 붓꽃 품종별 개수 구하기

- value_counts( ) 함수를 이용해, 품종별 데이터 개수를 확인해 보자

```
1 count = pd.DataFrame(iris[“Species”].value_counts())
2 count
```

실행결과

	Species
setosa	50
versicolor	50
virginica	50

02. 붓꽃 데이터 분석

❖ 붓꽃 데이터 전처리

- 데이터를 수집하는 과정에서 데이터가 누락되거나 또는 데이터가 중복되는 경우가 있을 수 있음
- 결측 데이터 또는 중복 데이터가 포함된 상태에서 데이터 분석을 수행하게 될 경우,
  올바른 분석 결과를 기대하기 어려움

결측 데이터와 중복 데이터를
확인하고 처리하는 방법을 알아보자

02. 붓꽃 데이터 분석

❖ 결측치 확인하기

- 데이터프레임에 isna( ) 함수를 적용하여, 각 원소가 결측값(Null)인지 여부를 체크할 수 있음
- 결측값이면 True, 아니면 False를 반환함
- 이 상태에서 sum( ) 함수를 적용하면, 각 열의 결측값의 개수를 반환함

```
1 iris.isna().sum()
```

실행결과

```
SepalLength    0
SepalWidth     0
PetalLength    0
PetalWidth     0
Species        0
dtype: int64
```

02. 붓꽃 데이터 분석

❖ 중복 데이터 확인하기 (1/3)

- 데이터프레임에 duplicated( ) 함수를 적용하여, 동일한 샘플이 중복되어 존재하는지 확인할 수 있음
- 중복되면 True, 아니면 False를 반환함
- 이 상태에서 sum( ) 함수를 적용하면, 중복된 샘플의 개수를 반환함

```
1 iris.duplicated().sum()
```

실행결과

```
1
```

**중복 데이터 존재 여부는 확인할 수 있지만,
어느 행의 데이터가 중복된 것인지 알 수 없음.
어떻게 해야 할까?**

02. 붓꽃 데이터 분석

❖ 중복 데이터 확인하기 (2/3)

- 부울린 인덱싱(Boolean Indexing)을 이용하면, 실제 어느 행의 데이터가 중복인지 확인할 수 있음
- loc[ ]의 행 위치에 중복 데이터 여부를 표시하는 bool 값을 입력해 보자
- 중복인 행은 True이므로, 중복 데이터만 표시됨

```
1 index = iris.duplicated()
2 iris.loc[index, :]
```

실행결과

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
142	5.8	2.7	5.1	1.9	virginica

142번 행은 어느 행과 중복되는 걸까?
확인해 보자

❖ 중복 데이터 확인하기 (3/3)

- 앞서 출력해 얻은 결과를 이용해, 어떤 데이터 행끼리 중복되는지 확인해 보자

```
1 result = (iris["SepalLength"]==5.8) & (iris["PetalWidth"]==1.9)
2 iris.loc[result,:]
```

실행결과

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
101	5.8	2.7	5.1	1.9	virginica
142	5.8	2.7	5.1	1.9	virginica

101번 행과 142번 행이 중복됨을 알 수 있음

02. 붓꽃 데이터 분석

❖ 중복 데이터 삭제하기

- drop_duplicates( ) 함수를 이용하면, 중복된 데이터를 삭제할 수 있음

```
1 iris = iris.drop_duplicates()
2 print(iris.duplicated().sum())
3 iris.loc[result, :]
```

실행결과

```
0
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
101	5.8	2.7	5.1	1.9	virginica

142번 행 즉, 중복 데이터가 삭제됨!

02. 붓꽃 데이터 분석

❖ 붓꽃 데이터 그룹핑: ① 품종 열을 기준으로 합계 구하기

- 붓꽃의 품종 별로 각 특성의 합계를 계산해 보자

```
1 iris.groupby("Species").sum()
```

실행결과

Species	SepalLength	SepalWidth	PetalLength	PetalWidth
setosa	250.3	171.4	73.1	12.3
versicolor	296.8	138.5	213.0	66.3
virginica	323.6	146.0	272.5	99.4

02. 붓꽃 데이터 분석

❖ 붓꽃 데이터 그룹핑: ② 품종 열을 기준으로 평균 구하기

- 이번엔 붓꽃의 품종 별로 각 특성의 평균값을 계산해 보자

```
1 iris.groupby(“Species”).mean()
```

실행결과

Species	SepalLength	SepalWidth	PetalLength	PetalWidth
setosa	5.006000	3.428000	1.462000	0.246000
versicolor	5.936000	2.770000	4.260000	1.326000
virginica	6.604082	2.979592	5.561224	2.028571

02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화

- 판다스의 시리즈나 데이터프레임은 'plot'이라는 시각화 메소드를 내장하고 있음
- plot( )은 matplotlib을 내부에서 임포트하여 사용함
- plot( ) 메소드의 kind라는 매개변수를 통해서, 여러 가지 그래프를 그릴 수 있음

plot( ) 메소드 내 kind 매개변수에 지정할 수 있는 값

옵션	종류	옵션	종류
line	선 그래프	kde	커널 밀도 그래프
bar	막대 그래프 – 수직	area	면적 그래프
barh	막대 그래프 – 수평	pie	원형 그래프
hist	히스토그램 그래프	scatter	산점도 그래프
box	박스 그래프	hexbin	고밀도 산점도 그래프

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화: ① 막대 그래프 그리기 (1/3)

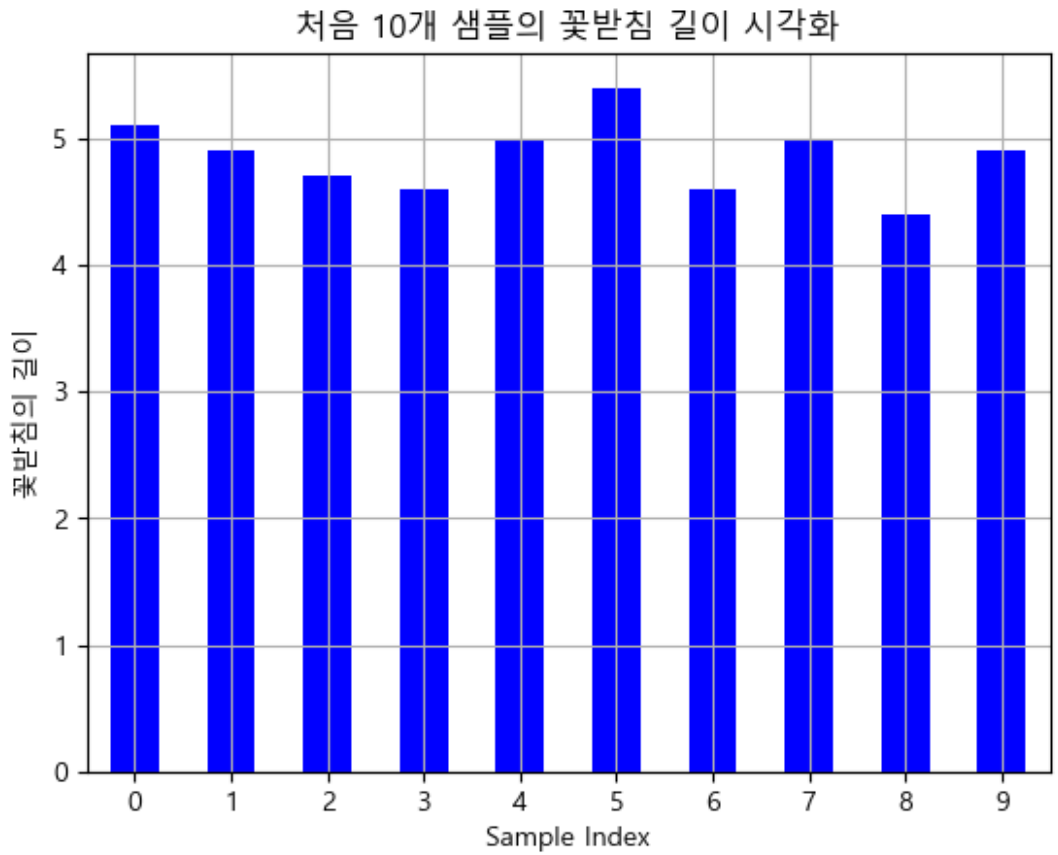
- kind="bar"를 지정하여, 처음 10개 샘플의 꽃받침 길이에 대한 수직 막대 그래프를 그려보자

```
1  import matplotlib.pyplot as plt
2
3  plt.rc("font", family="Malgun Gothic")
4  plt.rcParams["axes.unicode_minus"] = False
5
6  iris.SepalLength[:10].plot(kind="bar", rot=0, color="blue")
7  plt.title("처음 10개 샘플의 꽃받침 길이 시각화")
8  plt.xlabel("Sample Index")
9  plt.ylabel("꽃받침의 길이")
10 plt.grid()
11 plt.show()
```

02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화: ① 막대 그래프 그리기 (2/3)

실행결과



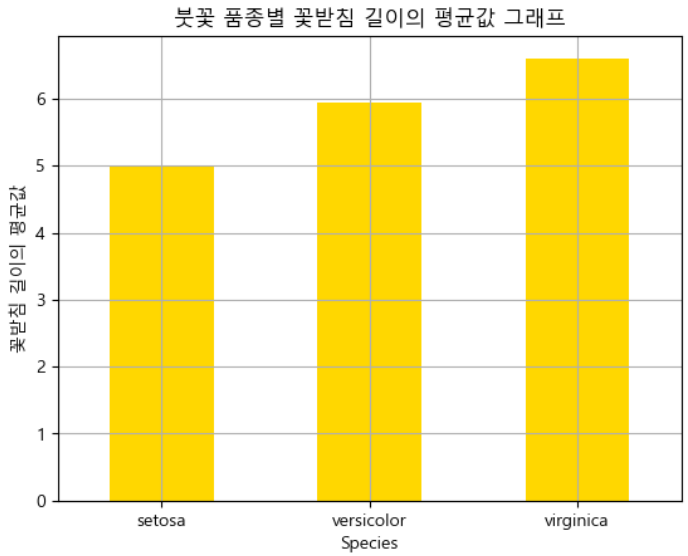
02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화: ① 막대 그래프 그리기 (3/3)

- 붓꽃 종류별 꽃받침 길이의 평균값을 막대 그래프로 시각화 해보자

```
1 df2 = iris.groupby(iris[“Species”]).mean()
2 df2.SepalLength.plot(kind=“bar”, rot=0, color=“gold”)
3 plt.title(“붓꽃 품종별 꽃받침 길이의 평균값 그래프”)
4 plt.ylabel(“꽃받침 길이의 평균값”)
5 plt.grid()
6 plt.show()
```

실행결과



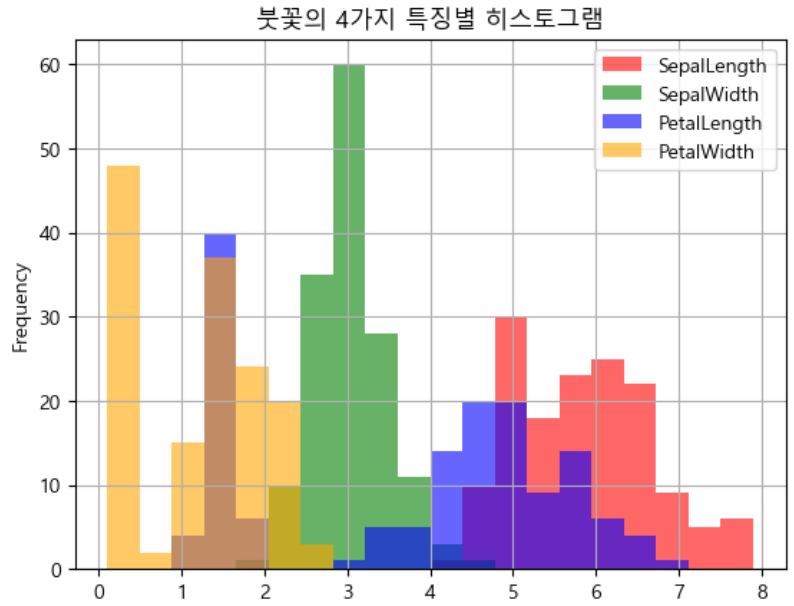
02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화: ② 히스토그램 그리기

- kind="hist"를 지정하여, 붓꽃 데이터의 4가지 특징별 히스토그램을 그려보자

```
1 iris.plot(kind="hist", bins=20, alpha=0.6,
2           color=["red", "green", "blue", "orange"])
3 plt.title("붓꽃의 4가지 특징별 히스토그램")
4 plt.grid()
5 plt.show()
```

실행결과



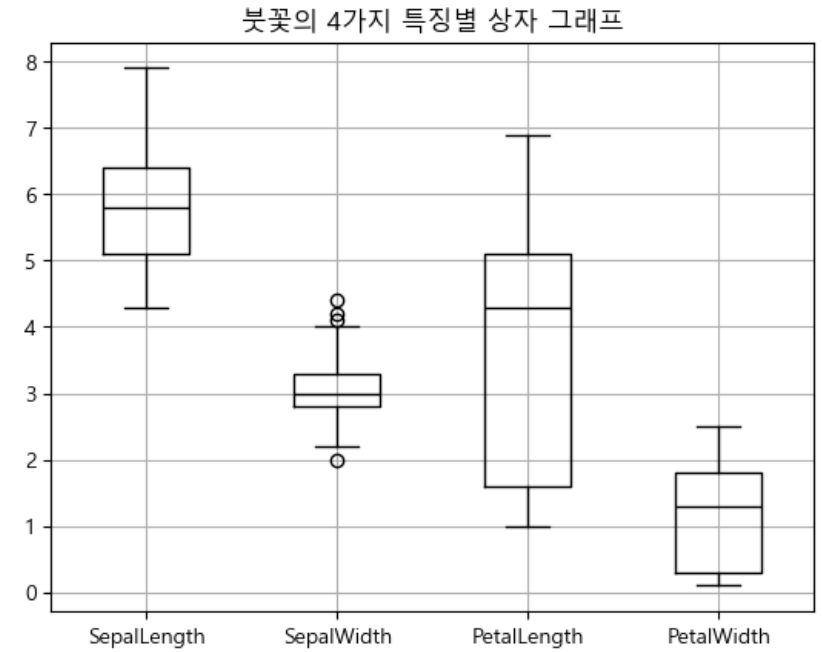
02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화: ③ 상자 그래프 그리기

- 이번엔 kind="box"를 지정하여, 붓꽃 데이터의 4가지 특징별 상자 그래프를 그려보자

```
1 iris.plot(kind="box", color="black")
2 plt.title("붓꽃의 4가지 특징별 상자 그래프")
3 plt.grid()
4 plt.show()
```

실행결과



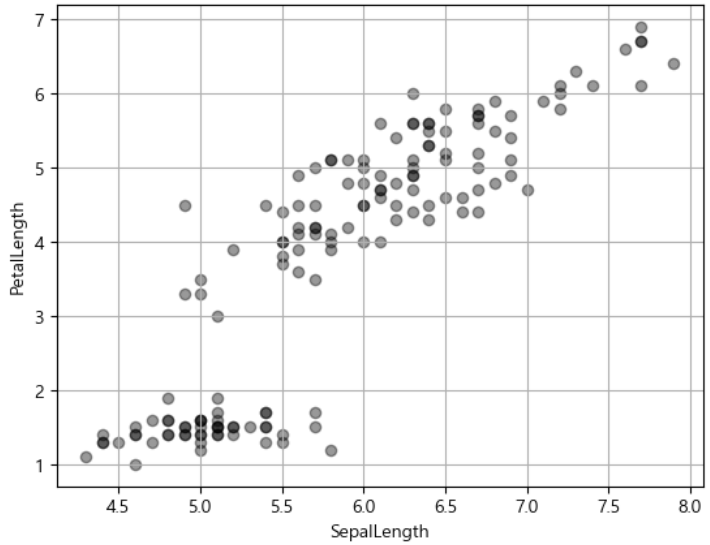
02. 붓꽃 데이터 분석

❖ 판다스의 데이터 시각화: ④ 산점도 그리기

- kind="scatter"를 지정하여, x축을 꽃받침의 길이(SepalLength) 그리고 y축을 꽃잎의 길이(PetalLength)로 하는 산점도를 그려보자

```
1 iris.plot(kind="scatter", x="SepalLength", y="PetalLength",
2            color="black", s=35, alpha=0.4)
3 plt.grid()
4 plt.show()
```

실행결과



**꽃받침이 길어질수록,
꽃잎도 길어지는 경향을 확인할 수 있음**

03. 타이배타닉 데이터 분석

01. 데이터 처리가 쉬운 판다스
02. 붓꽃 데이터 분석

03. 타이타닉 데이터 분석

❖ 데이터 설명

- 타이타닉 데이터는 데이터 사이언스나 머신러닝 분야에서 입문자용으로 널리 사용되는 데이터임
- 실제로 타이타닉 데이터는 데이터 분석 경연 사이트인 캐글(Kaggle)에서 입문자용으로 사용되고 있음
- 타이타닉 데이터를 활용하여 공부하면 데이터 분석의 전반적인 과정을 습득하는 데,
  도움을 받을 수 있음

타이타닉 호 침몰 사건 당시, 사망자와 생존자를 구분하는 요인 분석을 통해 승객들의 생존 여부를 예측해 보자

03. 타이타닉 데이터 분석

❖ 필드의 이해

- 타이타닉 데이터 세트에 포함된 12개의 변수에 대해 살펴보자

변수명	변수 설명
PassengerId	승객 번호
Survived	생존 여부 : 0=사망, 1=생존
Pclass	객실 등급 : 1=등급, 2=등급, 3=등급
Name	승객 이름
Sex	성별
Age	나이
SibSp	함께 탑승한 형제와 배우자의 수
Parch	함께 탑승한 부모, 아이의 수
Ticket	티켓 번호
Fare	탑승 요금
Cabin	객실 번호
Embarked	탑승 항구 : C=Cherbourg/Q=Queenstown/S=Southampton

[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 읽어오기

- titanic.csv 파일로부터 타이타닉 데이터를 읽고, 처음 5줄의 데이터를 출력해 보자

```
1 import pandas as pd
2
3 titanic = pd.read_csv("titanic.csv")
4 titanic.head()
```

실행결과

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터의 기본 정보 출력

- info( ) 함수를 이용해, 데이터프레임의 기본 정보를 확인해 보자

```
1 titanic.info()
```

실행결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

샘플 수는 891개로, 결측 데이터를 포함한 컬럼도 존재하는 것을 확인할 수 있음

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터의 기초 통계량 출력

- describe( ) 함수를 이용해, 타이타닉 데이터의 기초 통계량을 확인해 보자

```
1 titanic.describe()
```

실행결과

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

평균 생존률이 약 38.4%임을 알 수 있음

03. 타이타닉 데이터 분석

❖ 요금 기준으로 내림차순하기

- sort_values( ) 함수를 사용해서, 요금(Fare)을 기준으로 데이터를 내림차순해 보자

```
1 titanic.sort_values("Fare", ascending=False)
```

실행결과

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	C
737	738	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	C
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
...	...	...	...	...	...	...	...	...	...	...	...	...
633	634	0	1	Parr, Mr. William Henry Marsh	male	NaN	0	0	112052	0.0000	NaN	S
413	414	0	2	Cunningham, Mr. Alfred Fleming	male	NaN	0	0	239853	0.0000	NaN	S
822	823	0	1	Reuchlin, Jonkheer. John George	male	38.0	0	0	19972	0.0000	NaN	S
732	733	0	2	Knight, Mr. Robert J	male	NaN	0	0	239855	0.0000	NaN	S
674	675	0	2	Watson, Mr. Ennis Hastings	male	NaN	0	0	239856	0.0000	NaN	S

891 rows × 12 columns

03. 타이타닉 데이터 분석

❖ 사망자와 생존자 수 확인하기

- value_counts( ) 함수를 사용해서, 사망자와 생존자 수를 확인해 보자

```
1 titanic[“Survived”].value_counts()
```

실행결과

```
0    549
1    342
Name: Survived, dtype: int64
```

Survived 컬럼 (0: 사망, 1: 생존)

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 전처리: ① 결측치 확인하기

- 타이타닉 데이터의 컬럼별 결측 데이터 수를 확인해 보자

```
1 titanic.isna().sum()
```

실행결과

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

나이(Age), 객실 번호(Cabin), 탑승 항구(Embarked) 컬럼에 결측 데이터가 존재함!

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 전처리: ② 객실 번호(Cabin) 컬럼 삭제하기

- 결측 데이터가 너무 많으면, 데이터 분석을 제대로 수행할 수 없으므로 객실 번호(Cabin) 컬럼을 삭제함

```
1 titanic.drop(["Cabin"], axis=1, inplace=True)
2 titanic.columns
```

실행결과

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

객실 번호(Cabin) 컬럼이 삭제됨!

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 전처리: ③ 탑승 항구(Embarked) 컬럼 내 결측치 처리 (1/2)

- 탑승 항구(Embarked) 컬럼에는 2개의 결측치가 있음
- 컬럼 제거보다는 결측 데이터를 어떤 값으로 대체하는 것이 적절해 보임
- 최빈값으로 결측 데이터를 대체해 보자
- 우선, value_counts( ) 함수를 통해 최빈값을 찾아보자

```
1 titanic[“Embarked”].value_counts()
```

실행결과

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

최빈값이 ‘S’라는 것을 확인함!
‘S’로 2개의 결측 데이터를 대체해 보자!

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 전처리: ③ 탑승 항구(Embarked) 컬럼 내 결측치 처리 (2/2)

- fillna( ) 함수에 찾은 최빈값 'S'를 입력해, 결측 데이터를 대체해 보자

```
1 titanic["Embarked"] = titanic["Embarked"].fillna('S')
2 titanic.isna().sum()
```

실행결과

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         0
dtype: int64
```

이제 남은 것은 나이(Age)임!

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 전처리: ④ 나이(Age) 컬럼 내 결측치 처리

- 나이는 결측치가 많고, 생존 여부와 상관이 있을 것으로 판단됨
- 나이 컬럼 내, 결측 데이터는 평균값으로 대체해 보자

```
1 avg = titanic[“Age”].mean()
2 titanic[“Age”] = titanic[“Age”].fillna(avg)
3 titanic.isna().sum()
```

실행결과

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

결측 데이터가 있던 3개의 컬럼 모두 처리 완료함!

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 그룹핑하기: ① 항구별 Pclass 컬럼의 평균값 계산

- 3개의 항구에서 타이타닉호에 탑승한, 탑승자의 객실 등급(Pclass) 평균값을 확인해 보자

```
1 titanic[“Pclass”].groupby(titanic[“Embarked”]).mean()
```

실행결과

```
Embarked
C    1.886905
Q    2.909091
S    2.346749
Name: Pclass, dtype: float64
```

03. 타이타닉 데이터 분석

❖ 타이타닉 데이터 그룹핑하기: ② 객승 등급(Pclass)과 성별(Sex)을 기준으로 평균값 계산

- 탑승자의 객실 등급(Pclass)과 성별(Sex)에 따른, 각 컬럼의 평균값을 확인해 보자

```
1 titanic.drop(["Name", "Ticket", "Embarked"], axis=1, inplace=True)
2 titanic.groupby(["Pclass", "Sex"]).mean()
```

실행결과

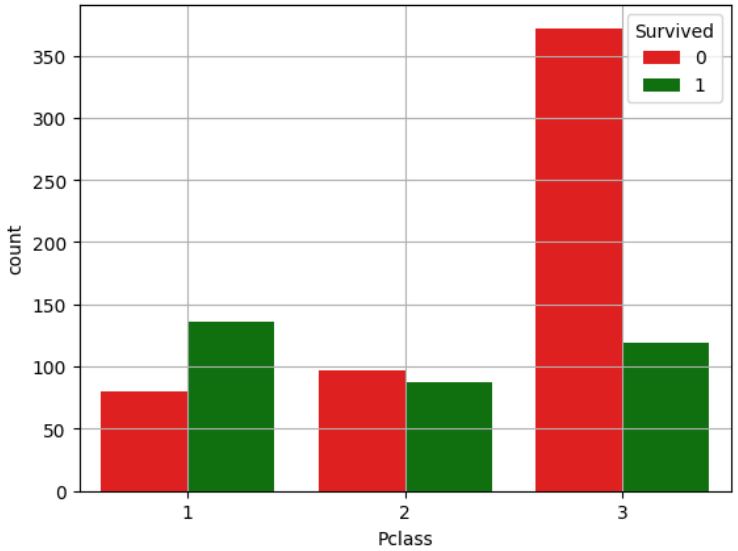
Pclass	Sex	PassengerId	Survived	Age	SibSp	Parch	Fare
1	female	469.212766	0.968085	34.141405	0.553191	0.457447	106.125798
	male	455.729508	0.368852	39.287717	0.311475	0.278689	67.226127
2	female	443.105263	0.921053	28.748661	0.486842	0.605263	21.970121
	male	447.962963	0.157407	30.653908	0.342593	0.222222	19.741782
3	female	399.729167	0.500000	24.068493	0.895833	0.798611	16.118810
	male	455.515850	0.135447	27.372153	0.498559	0.224784	12.661633

03. 타이타닉 데이터 분석

❖ 판다스 데이터 시각화: ① 객실 등급(Pclass)별 생존자 확인 (1/2)

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 sns.countplot(data=titanic, x="Pclass", hue="Survived", palette=["red", "green"],
5                stat="count") # stat의 기본값은 count
6 plt.grid()
7 plt.show()
```

실행결과



백분율로 표시하려면 어떻게 해야 할까?

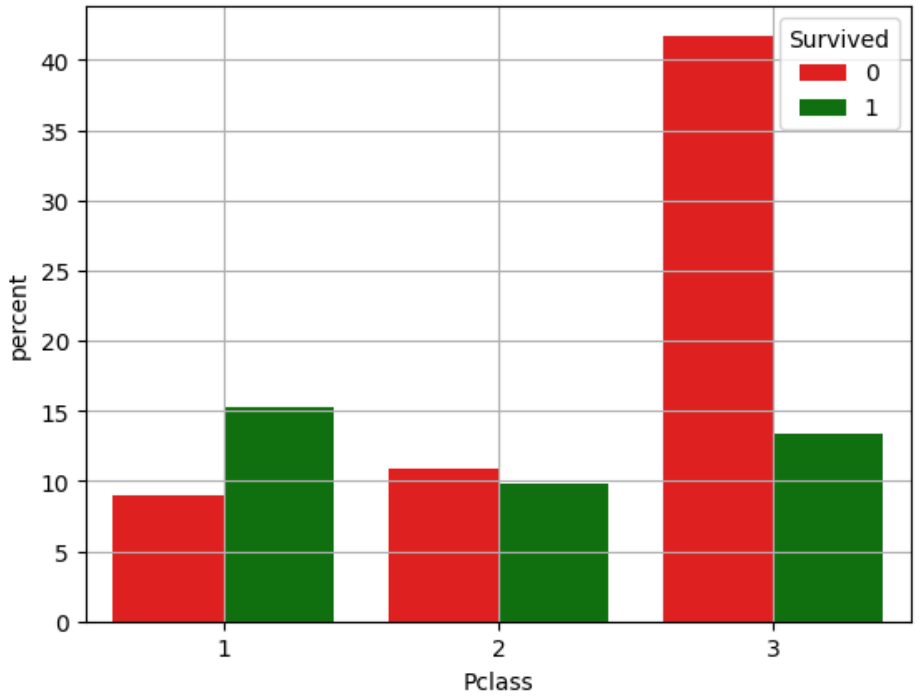
- hue
  (명사) 빛깔, 색조
  (명사) 색깔
- palette
  (명사) 팔레트
  (명사) (특정 화가가 쓰는) 색깔들
- Statistic
  (명사) 통계
  (동사) 통계학
  (동사) 통계 자료

03. 타이타닉 데이터 분석

❖ 판다스 데이터 시각화: ① 객실 등급(Pclass)별 생존자 확인 (2/2)

```
1 sns.countplot(data=titanic, x="Pclass", hue="Survived", palette=["red", "green"],
2               stat="percent")
3 plt.grid()
4 plt.show()
```

실행결과



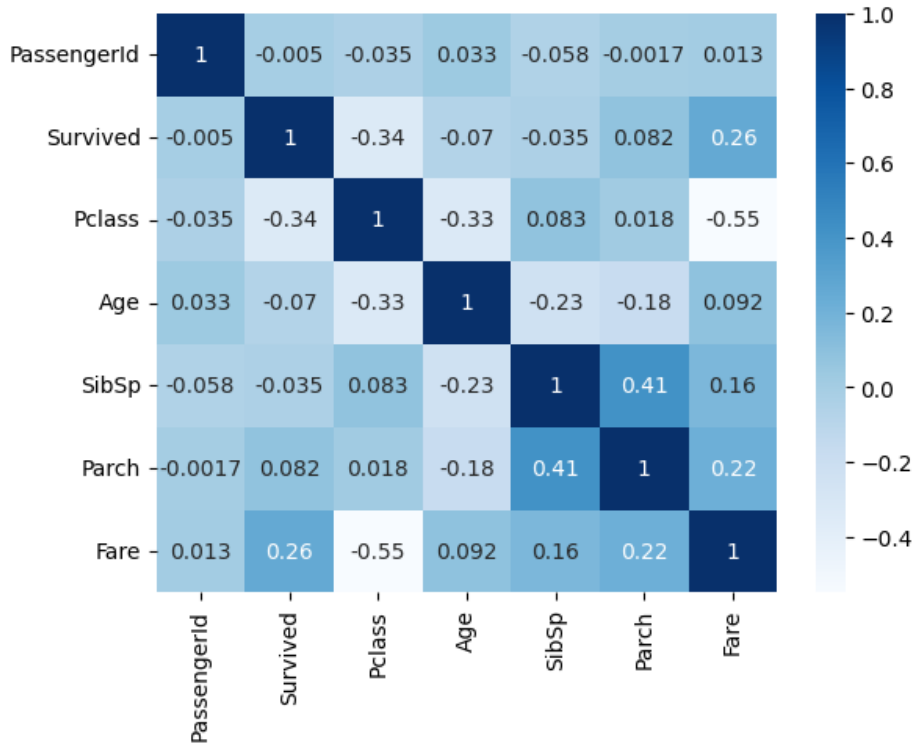
생존율은 1등급 객실의 탑승자가 높았으며, 사망률은 3등급 객실의 탑승자가 높았음

03. 타이타닉 데이터 분석

❖ 판다스 데이터 시각화: ② 상관관계 확인

```
1 titanic.drop(["Sex"], axis=1, inplace=True)
2 sns.heatmap(data=titanic.corr(), annot=True, cmap="Blues")
3 plt.show()
```

실행결과



❖ 상관 분석에 대한 이해 (1/3)

- 상관 분석 개념

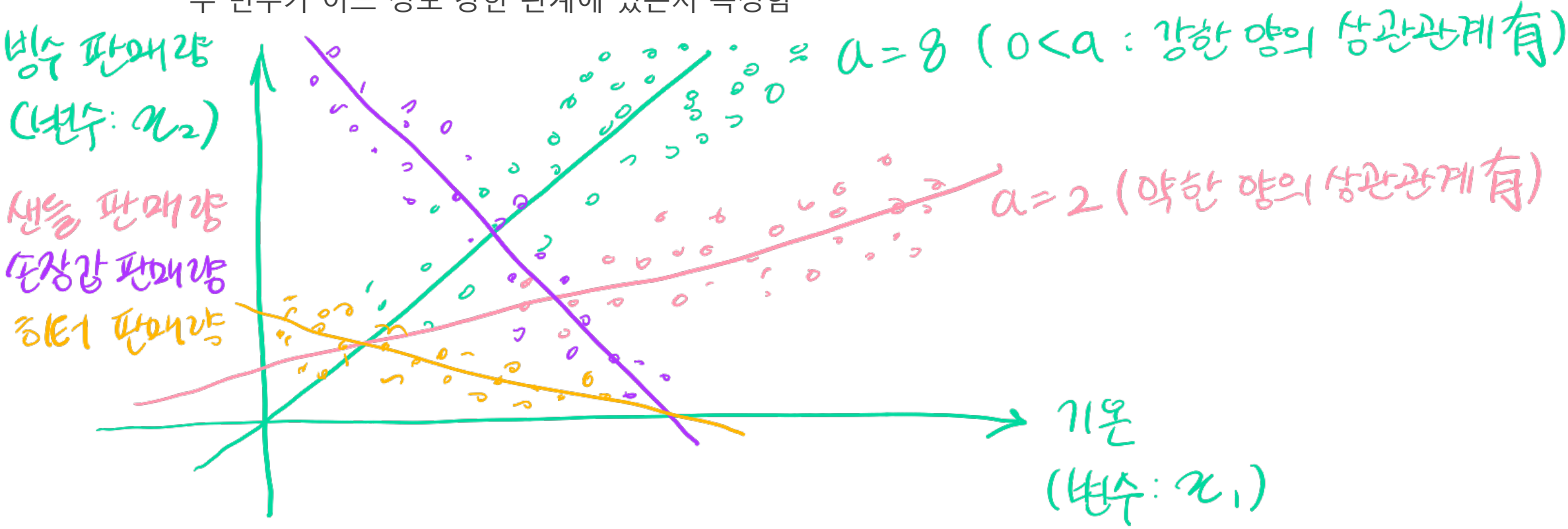
전제 $y = ax + b$

$0 < a$ /

$a < 0$ \

◆ 두 변수가 어떤 선형적 관계에 있는지 분석하는 방법

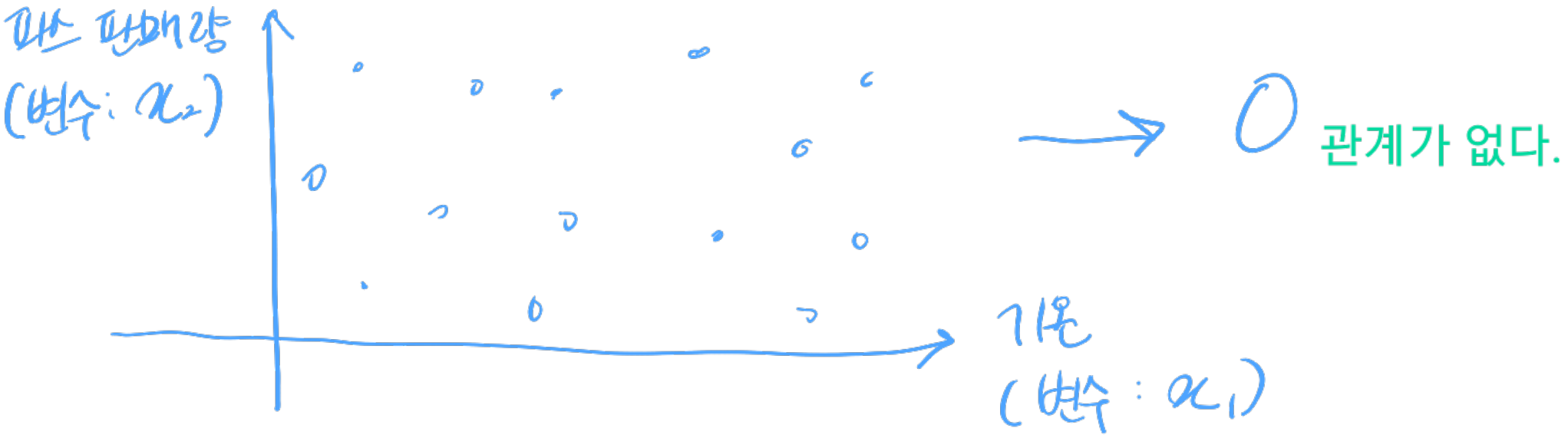
◆ 두 변수는 서로 독립적이거나 상관된 관계일 수 있는데,
두 변수가 어느 정도 강한 관계에 있는지 측정함



03. 타이타닉 데이터 분석

❖ 상관 분석에 대한 이해 (2/3)

- 상관 계수
  - ◆ 변수 간 관계의 정도(0~1)와 방향(+, -)을 하나의 수치로 요약하는 지수
  - ◆ -1과 1사이의 값을 가짐 1에 가까울수록 강한 양의 상관관계, -1에 가까울수록 강한 음의 상관관계
  - ◆ 상관 계수가 +면 양의 상관관계이며, 한 변수가 증가할 때 다른 변수도 증가함
  - ◆ 상관 계수가 –면 음의 상관관계이며, 한 변수가 증가할 때 다른 변수는 감소함
  - ◆ 상관 계수는 데이터프레임의 corr( ) 함수를 이용하면 구할 수 있음



03. 타이타닉 데이터 분석

❖ 상관 분석에 대한 이해 (3/3)

- 상관 분석 결과의 시각화

  ◆ 두 변수의 관계를 보여 주는, 산점도(Scatter Plot)이나 히트맵(Heatmap)을 주로 사용함

seaborn

끝맺음

- ❖ 01. 데이터 처리가 쉬운 판다스
- ❖ 02. 붓꽃 데이터 분석
- ❖ 03. 타이타닉 데이터 분석

THANK YOU!
Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr