

# 파이썬으로 시작하는 데이터 분석

02 pandas



# 목차

- 
- 01 Pandas 라이브러리 소개
  - 02 데이터 살펴보기
  - 03 데이터 변환하기 1
  - 04 데이터 변환하기 2



# 목차

05 데이터 요약하기

06 데이터 추출하기

07 데이터 정제하기

08 데이터 병합하기

01

# Pandas 라이브러리 소개

## ✓ Pandas



데이터 조작 및 분석을 위한 파이썬 라이브러리

시리즈와 데이터프레임이라는 데이터분석에 유용한 데이터 구조 제공

엑셀의 파이썬 버전

## ✓ Pandas를 배워야 하는 이유



대용량의 데이터를 처리하기 용이

프로그래밍을 통한 반복적인 작업 및 자동화에 유리

머신러닝/딥러닝 모델에 필요한 데이터 처리

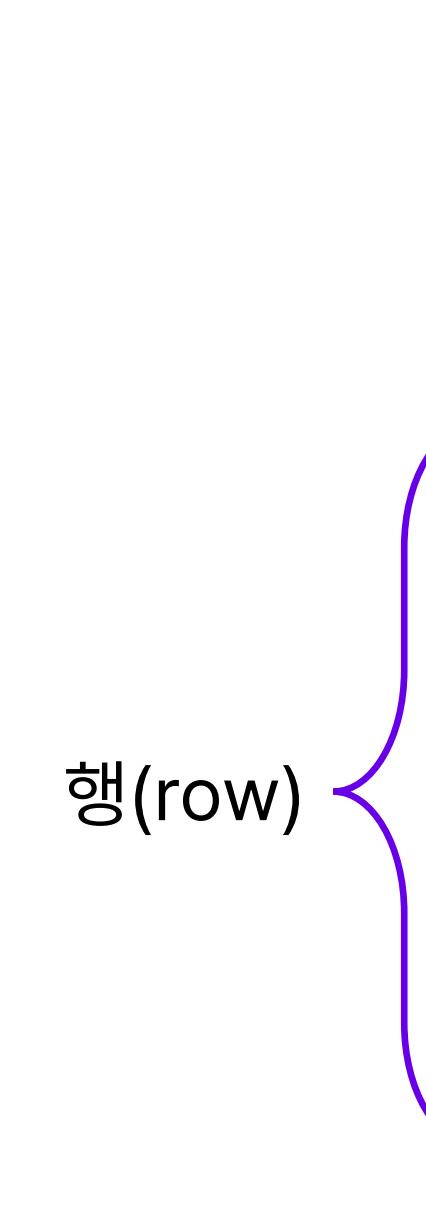
## ⑤ 데이터

- 대부분의 데이터는 엑셀과 유사한 표 형태로 저장

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256

## ✓ DataFrame

- Pandas 라이브러리에서 지원하는 자료구조
- 2차원의 행렬 데이터를 표 형태로 저장
- 가로인 행(row)과 세로인 열(column)으로 이루어짐
- 각 행과 열은 인덱스를 가지고 있어  
데이터를 쉽게 검색하고 필터링 가능



index	국어	수학	영어
민수	95	88	96
도윤	72	72	68
서아	83	93	85
연우	89	80	94
하준	100	95	93

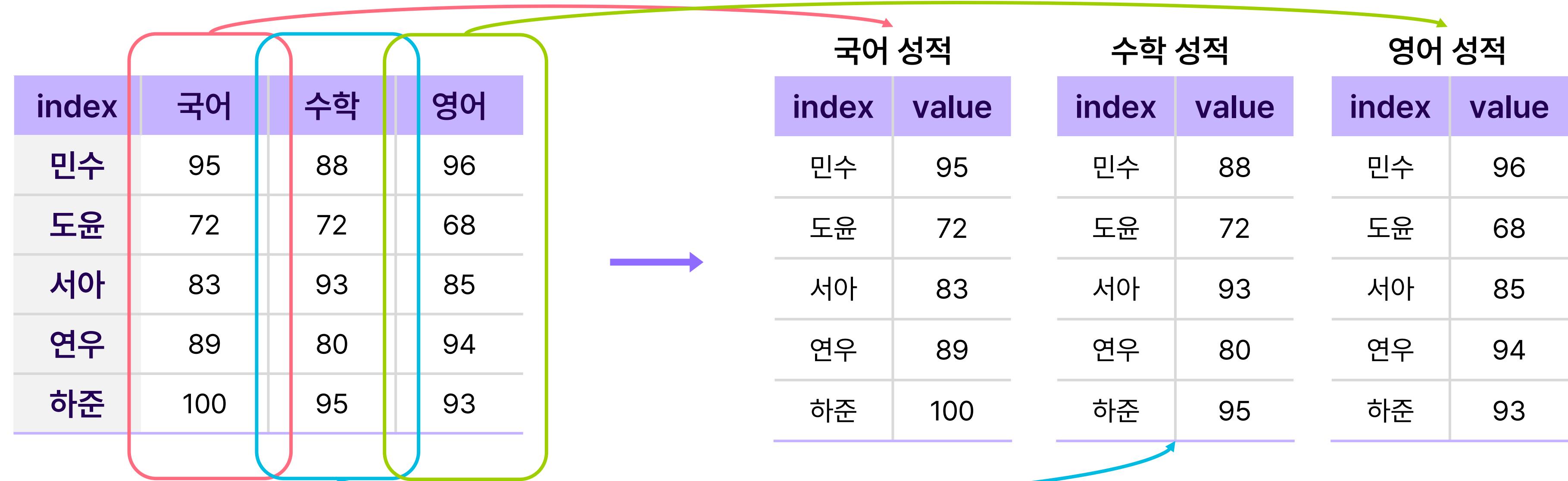
## ✓ Series

- Pandas 라이브러리에서 지원하는 자료구조
- 각 원소는 인덱스와 값으로 이루어짐
- 인덱스는 숫자 또는 문자열로 지정 가능

index	value	index	value
0	95	민수	95
1	72	도윤	72
2	83	서아	83
3	89	연우	89
4	100	하준	100

## ✓ 시리즈와 데이터프레임

- 데이터프레임에서 하나의 열 데이터를 추출하면 시리즈



## ⑤ Pandas 사용하기

- Pandas 라이브러리를 사용하기 위해서는 불러오는 작업 필요

```
import pandas as pd
```

- pandas 라이브러리를 불러와서, pd라는 별명으로 사용한다는 의미

```
df=pd.read_csv('data/seoul_park.csv')
```

pd(pandas)에 있는 read\_csv() 함수를 사용

02

## 데이터 살펴보기

### ⑤ 데이터 파일

- Pandas는 다양한 형식의 데이터 파일을 불러올 수 있음
- 우리가 실제로 사용할 데이터 대부분은 .csv 파일 혹은 .xlsx 파일 형식

## ⑤ 강의에 활용할 데이터

- 2016년 1월 1일부터 2019년 3월 31일까지의 서울대공원 입장객 데이터
- 조류독감으로 인해 2016년 12월 18일부터 2017년 3월 27일까지 폐쇄되어 기록이 없음
- 입장객의 수와 분류, 공휴일 여부와 날씨 등이 기록되어 있음
- data라는 폴더 안에 “seoul\_park.csv”파일로 저장되어 있음

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256



seoul\_park.csv

## ⑤ 데이터 불러오기: `read_csv()`, `read_excel()`

- 파일로 저장 되어있는 데이터를 불러와서 데이터프레임으로 저장
- .csv 파일로 저장 되어있는 데이터는 `read_csv()` 사용
- .xlsx 등의 파일로 저장되어 있는 데이터는 `read_excel()` 사용

```
df=pd.read_csv('data/seoul_park.csv')
```

data 폴더 안에 있는 seoul\_park.csv파일을 불러와 변수 df에 데이터프레임 형태로 저장

## ✓ 데이터 일부 확인: head(), tail()

- 데이터프레임의 앞 뒤 일부 데이터를 확인하는데 사용

df.head()

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256

df.tail()

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
1081	2019-03-27	X	구름 많음	504	464	10	30	21	0	613	1117
1082	2019-03-28	X	구름 많음	761	687	46	28	35	108	904	1665
1083	2019-03-29	X	구름 조금	1644	1447	120	77	14	188	1226	2870
1084	2019-03-30	O	흐림	1539	1326	44	169	29	115	913	2452
1085	2019-03-31	O	구름 조금	3061	2563	111	387	53	0	1357	4418

## ✓ 데이터 정보 확인: info()

- 데이터의 정보를 확인하는데 사용
- 어떤 컬럼이 있는지, 몇 개의 값이 있는지, 어떤 형태로 저장되어 있는지 확인 가능

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1086 entries, 0 to 1085
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   날짜      1086 non-null   object 
 1   공휴일    1086 non-null   object 
 2   날씨      946 non-null   object 
 3   유료합계  1086 non-null   object 
 4   어른      1086 non-null   object 
 5   청소년    1081 non-null   object 
 6   어린이    1086 non-null   object 
 7   외국인    1086 non-null   object 
 8   단체      1086 non-null   object 
 9   무료합계  1086 non-null   object 
 10  총계     1086 non-null   object 
dtypes: object(11)
memory usage: 93.5+ KB
```

## ⑤ 데이터프레임에서 특정 컬럼 추출

- 데이터프레임에서 컬럼이름을 활용해 특정 컬럼을 시리즈 형태로 추출 가능

```
df["컬럼이름"]
```

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256
...	...	...	...	...	...	...	...	...	...	...	...
1081	2019-03-27	X	구름 많음	504	464	10	30	21	0	613	1117
1082	2019-03-28	X	구름 많음	761	687	46	28	35	108	904	1665
1083	2019-03-29	X	구름 조금	1644	1447	120	77	14	188	1226	2870
1084	2019-03-30	O	흐림	1539	1326	44	169	29	115	913	2452
1085	2019-03-31	O	구름 조금	3061	2563	111	387	53	0	1357	4418

```
df["어른"]
```

0	2799
1	4370
2	2571
3	602
4	319
...	...
1081	464
1082	687
1083	1447
1084	1326
1085	2563

## ⑤ 데이터프레임에서 특정 컬럼 추출

- 리스트를 활용하면 여러 개의 컬럼을 데이터프레임 형태로 추출 가능

컬럼의 이름들을 리스트로 묶어주는 대괄호

```
df[["컬럼 이름1", "컬럼 이름2"]]
```

df에서 추출할 컬럼들을 지정하는 대괄호

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256
...	...	...	...	...	...	...	...	...	...	...	...
1081	2019-03-27	X	구름 많음	504	464	10	30	21	0	613	1117
1082	2019-03-28	X	구름 많음	761	687	46	28	35	108	904	1665
1083	2019-03-29	X	구름 조금	1644	1447	120	77	14	188	1226	2870
1084	2019-03-30	O	흐림	1539	1326	44	169	29	115	913	2452
1085	2019-03-31	O	구름 조금	3061	2563	111	387	53	0	1357	4418

```
df[["공휴일", "어른"]]
```

	공휴일	어른
0	O	2799
1	O	4370
2	O	2571
3	X	602
4	X	319
...	...	...
1081	X	464
1082	X	687
1083	X	1447
1084	O	1326
1085	O	2563

## ✓ 데이터 숫자 세기: value\_counts()

- 해당 컬럼에 값들이 몇 개씩 저장되어 있는지 알고 싶을 때 사용

```
df["컬럼이름"].value_counts()
```

```
df["날씨"].value_counts()
```



구름	많음	277
구름	조금	236
맑음		222
비		101
흐림		100
눈		6
눈/비		4

Name: 날씨, dtype: int64

03

## 데이터 변환하기 1

## ⑤ 데이터 타입 변환의 필요성

- 데이터는 항상 원하는 타입으로 되어있지 않다!

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1086 entries, 0 to 1085
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   날짜      1086 non-null   object 
 1   공휴일    1086 non-null   object 
 2   날씨      946 non-null   object 
 3   유료합계  1086 non-null   object 
 4   어른      1086 non-null   object 
 5   청소년    1081 non-null   object 
 6   어린이    1086 non-null   object 
 7   외국인    1086 non-null   object 
 8   단체      1086 non-null   object 
 9   무료합계  1086 non-null   object 
 10  총계      1086 non-null   object 
dtypes: object(11)
memory usage: 93.5+ KB
```

- 실습에 사용할 데이터에 있는 숫자들이 정수 타입이 아닌 텍스트(object) 타입으로 저장되어 있음
- 텍스트(object) 타입일 경우 덧셈이나 뺄셈 등의 연산이 불가능

## ✓ 데이터 타입 변환: astype()

- 해당 컬럼의 데이터 타입을 원하는 타입으로 변환할 때 사용

```
df["컬럼이름"].astype(변환할타입)
```

```
df["어른"] = df["어른"].astype(int)
```

df의 "어른" 컬럼 데이터의 타입을 int로 변환하여

df의 "어른" 컬럼에 저장(덮어씌우기)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1086 entries, 0 to 1085
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   날짜      1086 non-null   object 
 1   공휴일    1086 non-null   object 
 2   날씨      946 non-null   object 
 3   유료합계  1086 non-null   object 
 4   어른      1086 non-null   int64 
 5   청소년    1081 non-null   object 
 6   어린이    1086 non-null   object 
 7   외국인    1086 non-null   object 
 8   단체      1086 non-null   object 
 9   무료합계  1086 non-null   object 
 10  총계     1086 non-null   object 
dtypes: int64(1), object(10)
memory usage: 93.5+ KB
```

## ✓ 데이터프레임 변환 시 유의사항

- Pandas의 메서드는 원본 데이터프레임(시리즈)을 바로 변환하지 않고,  
변환된 데이터프레임(시리즈)을 반환
- 원본 데이터프레임을 변환하려면 덮어씌워주는 작업이 필요

```
df["어른"].astype(int)
```

df의 "어른" 컬럼 데이터의 타입을 int로 변환한 시리즈 그 자체

```
df["어른"] = df["어른"].astype(int)
```

df의 "어른" 을 astype으로 변환한 시리즈를 df의 "어른" 컬럼에  
덮어씌움으로서 원본 데이터프레임인 df가 변환됨

## ✓ 데이터 타입 변환: to\_numeric()

- 해당 컬럼의 데이터 타입을 숫자 타입으로 변환할 때 사용

```
pd.to_numeric(df["컬럼 이름"])
```

```
df["유료합계"] = pd.to_numeric(df["유료합계"])
```

df의 "유료합계" 컬럼의 데이터 타입을 숫자로 변환하여

df의 "유료합계" 컬럼에 저장(덮어씌우기)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1086 entries, 0 to 1085
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   날짜      1086 non-null   object 
 1   공휴일    1086 non-null   object 
 2   날씨      946 non-null   object 
 3   유료합계  1086 non-null   int64 
 4   어른      1086 non-null   int64 
 5   청소년    1081 non-null   object 
 6   어린이    1086 non-null   object 
 7   외국인    1086 non-null   object 
 8   단체      1086 non-null   object 
 9   무료합계  1086 non-null   object 
 10  총계      1086 non-null   object 
dtypes: int64(2), object(9)
memory usage: 93.5+ KB
```

## ⑤ 한꺼번에 타입 변환하기

- 변환해야 하는 컬럼들을 리스트로 만들고, for문을 활용해 한꺼번에 타입 변환 가능

```
columns=['유료합계', '어른', '청소년', '어린이', '외국인', '단체', '무료합계', '총계']
for i in columns:
    df[i]=pd.to_numeric(df[i])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1086 entries, 0 to 1085
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   날짜      1086 non-null   object 
 1   공휴일    1086 non-null   object 
 2   날씨      946 non-null   object 
 3   유료합계  1086 non-null   int64  
 4   어른      1086 non-null   int64  
 5   청소년    1081 non-null   float64
 6   어린이    1086 non-null   int64  
 7   외국인    1086 non-null   int64  
 8   단체      1086 non-null   int64  
 9   무료합계  1086 non-null   int64  
 10  총계      1086 non-null   int64  
dtypes: float64(1), int64(7), object(3)
memory usage: 93.5+ KB
```

## ✓ 데이터 타입 변환: to\_datetime()

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2016-01-01	O	구름 조금	3359	2799	141	419	47	0	1023	4382
1	2016-01-02	O	구름 많음	5173	4370	203	600	100	111	2092	7265
2	2016-01-03	O	구름 많음	3008	2571	128	309	91	0	1549	4557
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690
4	2016-01-05	X	구름 많음	416	319	35	62	43	47	840	1256

- 날짜/시간으로 변환 가능한 문자열, 정수, 실수를 **시간 타입**으로 변환할 때 사용

```
pd.to_datetime(df["컬럼 이름"])
```

```
df["날짜"] = pd.to_datetime(df["날짜"])
```

df의 “날짜” 컬럼의 텍스트(ex. “2016-01-01”)을  
시간 타입(e.x 2016년 1월 1일 시간 자료)으로 변환하여  
df의 “날짜”컬럼에 저장(덮어씌우기)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1086 entries, 0 to 1085
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   날짜      1086 non-null   datetime64[ns] 
 1   공휴일    1086 non-null   object  
 2   날씨      946 non-null   object  
 3   유료합계  1086 non-null   int64  
 4   어른      1086 non-null   int64  
 5   청소년    1081 non-null   float64 
 6   어린이    1086 non-null   int64  
 7   외국인    1086 non-null   int64  
 8   단체      1086 non-null   int64  
 9   무료합계  1086 non-null   int64  
 10  총계      1086 non-null   int64  
dtypes: datetime64[ns](1), float64(1), int64(7), object(2)
memory usage: 93.5+ KB
```

## ✓ 시간 타입을 활용한 데이터 변환: dt

- 시간 타입은 날짜와 시간에 대한 정보가 담겨있어 다양한 처리가 가능

```
df['연']=df['날짜'].dt.year
```

dt.year를 사용하여 년도 정보가 저장된

df["연"] 컬럼을 생성

```
df['요일']=df['날짜'].dt.dayofweek
```

dt.dayofweek를 사용하여 요일 정보가 저장된

df["요일"] 컬럼을 생성(요일은 숫자로 저장 (월~일:0~6))

```
df['월']=df['날짜'].dt.month
```

```
df['일']=df['날짜'].dt.day
```

	연	월	일	요일
2016	1	1	4	
2016	1	2	5	
2016	1	3	6	
2016	1	4	0	
2016	1	5	1	

04

## 데이터 변환하기 2

## ⑤ 데이터 변환

- 데이터 분석의 정확성을 높이기 위해 데이터의 값 변환이 필요한 경우가 많음
- 알아보기 쉬운 데이터로 바꾸거나, 단위를 통일하는 등 다양한 변환 가능

연	월	일	요일
2016	1	1	4
2016	1	2	5
2016	1	3	6
2016	1	4	0
2016	1	5	1

## ⑤ 시리즈 연산을 통한 변환

- 데이터프레임의 각 열은 시리즈로, 연산이 가능함
- 다양한 컬럼의 연산을 활용해 데이터를 변환하거나 새로운 컬럼 생성 가능

```
df["어른"] = df["어른"] + 200
```

df의 "어른"컬럼의 전체 값에 200을 더해 덮어씌우기

```
df["동물원매출"] = df["어른"] * 5000 + df["청소년"] * 3000 + df["어린이"] * 2000
```

df의 "어른", "청소년", "어린이" 컬럼의 입장객 수에 해당하는 입장요금을 곱하고  
전체를 더해 df에 "동물원매출"컬럼 생성

해당 변환들은 적용하지 않고 이후 이론 강의 진행

## ✓ 시리즈 연산을 통한 변환

- 데이터프레임의 각 열은 시리즈로, 연산이 가능함
- 다양한 컬럼의 연산을 활용해 데이터를 변환하거나 새로운 컬럼 생성 가능

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일	동물원매출
0	2016-01-01	O	구름 조금	3359	3199	141.0	419	47	0	1023	4382	2016	1	1	4	15256000.0
1	2016-01-02	O	구름 많음	5173	4770	203.0	600	100	111	2092	7265	2016	1	2	5	23659000.0
2	2016-01-03	O	구름 많음	3008	2971	128.0	309	91	0	1549	4557	2016	1	3	6	13857000.0
3	2016-01-04	X	구름 많음	890	1002	Nan	235	51	223	800	1690	2016	1	4	0	Nan
4	2016-01-05	X	구름 많음	416	719	35.0	62	43	47	840	1256	2016	1	5	1	1824000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1081	2019-03-27	X	구름 많음	504	864	10.0	30	21	0	613	1117	2019	3	27	2	2410000.0
1082	2019-03-28	X	구름 많음	761	1087	46.0	28	35	108	904	1665	2019	3	28	3	3629000.0
1083	2019-03-29	X	구름 조금	1644	1847	120.0	77	14	188	1226	2870	2019	3	29	4	7749000.0
1084	2019-03-30	O	흐림	1539	1726	44.0	169	29	115	913	2452	2019	3	30	5	7100000.0
1085	2019-03-31	O	구름 조금	3061	2963	111.0	387	53	0	1357	4418	2019	3	31	6	13922000.0

해당 변환들은 적용하지 않고 이후 이론 강의 진행

### ✓ 열 전체 변환: map()

- 데이터프레임의 특정 열 전체를 변환하는데 사용

```
df["컬럼이름"].map(딕셔너리 등)
```

- 딕셔너리 등을 사용하여 변환 방식을 지정하고 해당 컬럼에 적용

## ❸ 데이터프레임에 map() 적용하기

df의 “요일” 컬럼의 값들을

딕셔너리 week의 Key와 Value를 활용하여  
숫자로 표시되어 있는曜일을 글자로 변경

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일
0	2016-01-01	O	구름 조금	3359	2799	141.0	419	47	0	1023	4382	2016	1	1	4
1	2016-01-02	O	구름 많음	5173	4370	203.0	600	100	111	2092	7265	2016	1	2	5
2	2016-01-03	O	구름 많음	3008	2571	128.0	309	91	0	1549	4557	2016	1	3	6
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690	2016	1	4	0
4	2016-01-05	X	구름 많음	416	319	35.0	62	43	47	840	1256	2016	1	5	1

```
week={0:'월', 1:'화', 2:'수', 3:'목', 4:'금',
5:'토', 6:'일'}
```

```
df['요일']=df['요일'].map(week)
```

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일
0	2016-01-01	O	구름 조금	3359	2799	141.0	419	47	0	1023	4382	2016	1	1	금
1	2016-01-02	O	구름 많음	5173	4370	203.0	600	100	111	2092	7265	2016	1	2	토
2	2016-01-03	O	구름 많음	3008	2571	128.0	309	91	0	1549	4557	2016	1	3	일
3	2016-01-04	X	구름 많음	890	602	Nan	235	51	223	800	1690	2016	1	4	월
4	2016-01-05	X	구름 많음	416	319	35.0	62	43	47	840	1256	2016	1	5	화

## ✓ 데이터에 함수 적용: apply()

- 데이터프레임에 함수를 적용할 때 사용

```
df["컬럼이름"].apply(함수, axis = 0 또는 1)
```

axis가 0이면 열 단위로, 1이면 행 단위로 함수 적용  
(생략되어 있으면 기본값은 0)

- map과 달리 복수의 컬럼에 사용 가능

## ✓ 데이터프레임에 apply() 적용하기

'눈'과 '비'를 '눈/비'로 합치는 weather함수를 선언

```
def weather(e):
    if e=='눈' or e=='비':
        return '눈/비'
    else:
        return e

df['날씨']=df['날씨'].apply(weather)
```

df의 "날씨" 컬럼의 값들에 weather함수를 적용하여 변환한 뒤

df의 "날씨" 컬럼에 저장(덮어씌우기)

날씨	개수	
구름	많음	277
구름	조금	236
맑음		222
비		101
흐림		100
눈		6
눈/비		4

Name: 날씨, dtype: int64

날씨	개수	
구름	많음	277
구름	조금	236
맑음		222
눈/비		111
흐림		100

Name: 날씨, dtype: int64

 참고: lambda 함수

- 간단하게 함수를 선언하는 문법
- 한 줄로 표현되며, 콜론(:) 왼쪽에는 인자를, 오른쪽에는 반환값을 작성

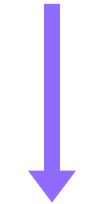
```
def times2(x):  
    return 2*x  
  
times2 = lambda x : 2*x
```

```
def add(x, y):  
    return x + y  
  
add = lambda x, y: x + y
```

## ✓ lambda함수를 활용한 apply 활용

```
def weather(e):
    if e=='눈' or e=='비':
        return '눈/비'
    else:
        return e

df['날씨']=df['날씨'].apply(weather)
```



```
df['날씨']=df['날씨'].apply(lambda e : '눈/비' if e=='눈' or e=='비' else e)
```

05

## 데이터 요약하기

### ⑤ 데이터 요약

- 데이터의 정보를 요약해서 확인하기 위한 과정
- 데이터에 대한 정보를 미리 파악하여 추후 분석 방향 판단 가능

✓ 데이터 통계값 확인: `mean()`, `min()`, `max()`, `median()`

```
df.mean()
```

```
df["컬럼 이름"].mean()
```

```
유료합계      4595.651934  
어른          3568.507366  
청소년        463.038853  
어린이        550.260589  
외국인          75.399632  
단체          892.831492  
무료합계        2090.390424  
총계          6686.042357  
연            2017.177716  
월            6.438306  
일            15.670350  
dtype: float64
```

```
df["어른"].mean()
```



3568.5073664825045

`min()`, `max()`, `median()` 등 다른 집계함수들도 같은 문법으로 활용 가능

## ✓ 데이터 전체 통계: describe()

- 데이터프레임이나 시리즈의 간단한 통계 정보를 요약해서 확인 가능
- 평균(mean), 표준편차(std), 최소값(min), 25/50/75% 분위수(25%/50%/75%), 최대값(max) 출력

df.describe()

	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일
count	1086.000000	1086.000000	1081.000000	1086.000000	1086.000000	1086.000000	1086.000000	1086.000000	1086.000000	1086.000000	1086.000000
mean	4595.651934	3568.507366	463.038853	550.260589	75.399632	892.831492	2090.390424	6686.042357	2017.177716	6.438306	15.670350
std	6461.199603	5456.322438	1424.486882	1013.065306	103.030042	2155.583046	2299.475934	8469.956231	0.980313	3.415790	8.817452
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	2.000000	2016.000000	1.000000	1.000000
25%	606.750000	462.500000	24.000000	43.000000	19.000000	0.000000	667.500000	1322.250000	2016.000000	3.000000	8.000000
50%	1697.000000	1282.000000	80.000000	150.000000	43.000000	73.000000	1182.000000	3030.000000	2017.000000	6.000000	16.000000
75%	5130.500000	3390.750000	298.000000	604.000000	87.000000	691.500000	2867.750000	8146.500000	2018.000000	9.000000	23.000000
max	40219.000000	38511.000000	17345.000000	20277.000000	1174.000000	18420.000000	22789.000000	58688.000000	2019.000000	12.000000	31.000000

## ✓ 데이터 그룹화: groupby()

- 데이터를 그룹으로 묶어 분석할 때 사용

```
df.groupby("컬럼이름1")["컬럼이름2"].집계함수
```

```
df.groupby("날씨")["총계"].mean()
```

df의 "날씨" 컬럼을 기준으로 그룹화하여  
"총계" 값의 평균을 집계



날씨		
구름 많음	6234.844765	
구름 조금	7409.122881	
눈/비	3038.603604	
맑음	7756.225225	
흐림	6056.450000	
Name:	총계	, dtype: float64

groupby()를 적용한 결과는 DataFrame 형태가 아닌 DataFrameGroupBy 형태

## ✓ 데이터 그룹화: groupby()

- 2개 이상의 컬럼을 기준으로 집계도 가능

```
df.groupby(["컬럼이름1", "컬럼이름2"])[["컬럼이름3"]].mean()
```

```
df.groupby(["날씨", "공휴일"])[["총입장객수"]].mean()
```

df의 “날씨”와 “공휴일” 컬럼을 기준으로

그룹화하여 “총입장객수”컬럼의 평균값을 집계



날씨	공휴일	
구름	많음	0 12994.262500
	x	3489.903553
구름	조금	0 14797.581081
	x	4034.148148
눈/비	o	4239.027778
	x	2462.400000
맑음	o	14658.260274
	x	4374.691275
흐림	o	9374.957447
	x	3113.622642

Name: 총입장객수, dtype: float64

## ✓ 데이터 그룹화: groupby()

- 특정 컬럼을 지정하지 않고 전체 컬럼에 대한 집계도 가능

```
df.groupby("컬럼이름").집계함수
```

```
df.groupby("날씨").mean()
```

df의 "날씨" 컬럼을 기준으로  
모든 컬럼의 평균값을 집계



	유료합계	어른	청소년	어린이	외국인	단체	무료합계	원
<strong>날씨</strong>								
구름 많음	4278.678700	3362.703971	401.698182	506.801444	76.043321	819.425993	1956.166065	6234.8447
구름 조금	5119.974576	4012.063559	485.540426	587.555085	73.389831	883.622881	2289.148305	7409.1228
눈/비	2124.261261	1411.900901	282.765766	421.099099	47.720721	643.918919	914.342342	3038.6036
맑음	5271.693694	4051.324324	641.108597	575.518018	71.684685	1127.621622	2484.531532	7756.2252
흐림	4206.080000	3452.790000	208.650000	542.180000	78.810000	547.390000	1850.370000	6056.4500

06

## 데이터 추출하기

### ⑤ 데이터 추출

- 원하는 구간, 조건에 해당하는 데이터를 추출해야 할 때 사용
- 데이터 분석 결과를 얻기 위해, 적절한 데이터를 선택하고 추출하기 위해 필요한 과정

 참고: pandas 논리연산자

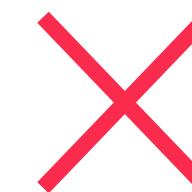
- and → &

$$(A > 30) \ \& \ (A < 50)$$

- or → |

A가 30보다 크고 50보다 작다

- not → ~


$$(30 < A < 50)$$

- 소괄호 ()를 사용하여 우선 순위 명시

부등호는 한 번에 하나의 비교만 가능!

$$(A > 30) \ | \ (B < 50)$$

A가 30보다 크거나 B가 50보다 작다

## ✓ 조건에 따른 인덱싱: Boolean indexing

- 비교 연산자나 논리 연산자를 사용하여 데이터프레임에서 조건에 맞는 행 추출 가능

df[조건식]

df[df["어른"]>1000]



	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일
0	2016-01-01	O	구름 조금	3359	2799	141.0	419	47	0	1023	4382	2016	1	1	금
1	2016-01-02	O	구름 많음	5173	4370	203.0	600	100	111	2092	7265	2016	1	2	토
2	2016-01-03	O	구름 많음	3008	2571	128.0	309	91	0	1549	4557	2016	1	3	일
8	2016-01-09	O	구름 많음	1418	1227	78.0	113	47	80	1222	2640	2016	1	9	토
9	2016-01-10	O	구름 많음	2211	1839	96.0	276	34	61	1393	3604	2016	1	10	일
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1077	2019-03-23	O	눈/비	2097	1760	44.0	293	42	0	1465	3562	2019	3	23	토
1078	2019-03-24	O	맑음	10039	8502	161.0	1376	74	35	2748	12787	2019	3	24	일
1083	2019-03-29	X	구름 조금	1644	1447	120.0	77	14	188	1226	2870	2019	3	29	금
1084	2019-03-30	O	흐림	1539	1326	44.0	169	29	115	913	2452	2019	3	30	토
1085	2019-03-31	O	구름 조금	3061	2563	111.0	387	53	0	1357	4418	2019	3	31	일

df의 “어른” 컬럼의 값이 1000보다 크다 라는 조건식을 만족하는 데이터를 추출

## ✓ 복잡한 조건을 활용한 Boolean indexing

- pandas 논리연산자를 활용하면 복잡한 조건을 추가할 수 있음

```
df[(df["어른"]>1000) & (df["어린이"]>1000)]
```



		날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일
78	2016-03-19	O	구름 조금	26448	22735	422.0	2990	112	178	6683	33131	2016	3	19	토	
79	2016-03-20	O	구름 조금	27093	22947	535.0	3370	176	0	7267	34360	2016	3	20	일	
85	2016-03-26	O	맑음	16958	14114	408.0	2094	192	62	4790	21748	2016	3	26	토	
86	2016-03-27	O	맑음	23610	20041	487.0	2860	205	60	6879	30489	2016	3	27	일	
92	2016-04-02	O	구름 조금	31407	26605	561.0	3643	169	201	6380	37787	2016	4	2	토	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
937	2018-11-03	O	맑음	11573	9817	231.0	1525	152	436	5053	16626	2018	11	3	토	
938	2018-11-04	O	구름 조금	12125	10223	307.0	1595	79	195	3778	15903	2018	11	4	일	
1063	2019-03-09	O	NaN	12602	11023	157.0	1422	52	74	5231	17833	2019	3	9	토	
1071	2019-03-17	O	NaN	12169	10313	199.0	1657	51	30	3549	15718	2019	3	17	일	
1078	2019-03-24	O	맑음	10039	8502	161.0	1376	74	35	2748	12787	2019	3	24	일	

df의 “어른” 컬럼의 값이 1000보다 크고 “어린이”컬럼의 값이 1000보다 크다 라는 조건식을 만족하는 데이터를 추출

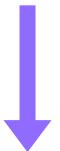
## ✓ 라벨을 활용한 데이터 추출: loc

- 라벨을 사용하여 데이터프레임에서 데이터를 추출하는 방법
- 인덱스이름과 컬럼이름을 입력 받아 그에 해당하는 데이터 추출

```
df.loc["인덱스이름", "컬럼이름"]
```

index가 456인 행의  
"총계" 열 데이터 추출

```
df.loc[456, "총계"]
```



338

index가 24, 67, 126인 행의  
"공휴일", "어른", "어린이" 열 데이터 추출

```
df.loc[[24, 67, 126], ["공휴일", "어른", "어린이"]]
```



	공휴일	어른	어린이
24	X	58	4
67	X	465	8
126	O	4764	1031

index가 35부터 40까지인 행의  
전체 열 데이터 추출

```
df.loc[35:40, :]
```



	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일	매출액
35	2016-02-05	X	구름 조금	304	234	41.0	29	14	0	536	840	2016	2	5	금	3040000
36	2016-02-06	O	맑음	1082	924	53.0	105	40	0	976	2058	2016	2	6	토	10820000
37	2016-02-07	O	맑음	1452	1251	74.0	127	191	0	676	2128	2016	2	7	일	14520000
38	2016-02-08	O	구름 많음	2654	2286	123.0	245	429	0	567	3221	2016	2	8	월	26540000
39	2016-02-09	O	맑음	5453	4610	223.0	620	209	0	1607	7060	2016	2	9	화	54530000
40	2016-02-10	O	맑음	7080	6037	251.0	792	141	0	2869	9949	2016	2	10	수	70800000

## ✓ loc과 Boolean indexing을 활용한 데이터 추출

- loc에서 “인덱스이름”에 Boolean indexing을 활용하면 조건에 맞는 데이터 추출 가능

```
df.loc[(df["어른"]>1000) | (df["어린이"]>1000), ["날짜", "공휴일", "어른", "어린이", "총계"]]
```

df의 “어른” 컬럼 값이 1000보다 크거나,

df의 “어린이” 컬럼 값이 1000보다 큰 행의

날짜, 공휴일, 어른, 어린이, 총계 컬럼 데이터 추출

	날짜	공휴일	어른	어린이	총계
0	2016-01-01	O	2799	419	4382
1	2016-01-02	O	4370	600	7265
2	2016-01-03	O	2571	309	4557
8	2016-01-09	O	1227	113	2640
9	2016-01-10	O	1839	276	3604
...	...	...	...	...	...
1077	2019-03-23	O	1760	293	3562
1078	2019-03-24	O	8502	1376	12787
1083	2019-03-29	X	1447	77	2870
1084	2019-03-30	O	1326	169	2452

## ✓ 순서를 활용한 데이터 추출: iloc

- 행과 열의 위치(순서)를 이용해 데이터를 추출할 때 사용
- 행이나 열의 위치(순서)를 나타내는 정수를 입력 받아 해당 데이터 추출

df.iloc[행의 위치, 열의 위치]

df.iloc[42, 5]

17.0

df.iloc[3:7,4:7]

어른 청소년 어린이

3 602 NaN 235

4 319 35.0 62

5 277 41.0 49

6 250 23.0 36

## ✓ loc vs iloc

- loc은 index 이름을 활용, iloc은 정수형 위치(순서)를 활용하여 인덱싱

index	국어	수학	영어
민수	95	88	96
도윤	72	72	68
서아	83	93	85
연우	89	80	94
하준	100	95	93

```
grade.loc["연우", "수학"]
```

```
grade.iloc[3, 1]
```

## ✓ loc과 iloc을 활용한 값 변환

- loc과 iloc을 활용하여 특정 원소 값 변환 가능
- loc과 iloc을 활용해 바꿀 위치를 지정하고 해당 위치에 넣을 값을 지정

index	국어	수학	영어
민수	95	88	96
도윤	72	72	68
서아	83	93	85
연우	89	80	94
하준	100	95	93

```
df.loc["서아", "국어"] = 100
```

```
df.iloc[1, 1] = 99
```



index	국어	수학	영어
민수	95	88	96
도윤	72	99	68
서아	100	93	85
연우	89	80	94
하준	100	95	93

07

## 데이터 정제하기

### ⑤ 데이터 정제

- 원본(Raw) 데이터는 불완전하거나 부정확한 경우가 많음
- 데이터가 누락되어 있거나, 잘못된 값이 포함되어 있는 경우 등 다양한 문제가 존재
- 데이터를 제거하거나 대체하는 등 다양한 정제과정이 필요

## ✓ 데이터 정렬: sort\_values()

- 데이터프레임의 특정 컬럼의 값을 기준으로 전체 데이터를 정렬할 때 사용

```
df.sort_values("컬럼 이름", ascending = True 또는 False)
```

- ascending이 True면 오름차순, False면 내림차순  
(생략되어 있으면 기본값인 True)

df.sort\_values("총계")

	날짜	공휴일	날씨	유료함계	어른	청소년	어린이	외국인	단체	무료함계	총계	연	월	일	요일	매출액
352	2017-03-28	X	구름 많음	0	0	0.0	0	0	0	0	2	2017	3	28	화	0
353	2017-03-29	X	구름 조금	0	0	0.0	0	0	0	0	7	2017	3	29	수	0
654	2018-01-24	X	맑음	18	16	1.0	1	0	0	11	128	2018	1	24	수	180000
655	2018-01-25	X	맑음	23	23	0.0	0	1	0	13	162	2018	1	25	목	230000
656	2018-01-26	X	맑음	50	31	13.0	6	0	0	17	222	2018	1	26	금	500000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
363	2017-04-08	O	구름 조금	28514	24756	638.0	3120	220	137	1237	40887	2017	4	8	토	285140000
156	2016-06-05	O	구름 조금	33504	28943	1079.0	3482	729	190	823	41743	2016	6	5	일	335040000
390	2017-05-05	O	구름 조금	27046	25824	1222.0	0	616	170	1921	46264	2017	5	5	금	270460000
755	2018-05-05	O	맑음	29521	28592	899.0	30	208	65	22789	52310	2018	5	5	토	295210000
125	2016-05-05	O	Nan	40219	38511	1708.0	0	476	35	18469	58688	2016	5	5	목	402190000

df.sort\_values("총계", ascending=False)

	날짜	공휴일	날씨	유료함계	어른	청소년	어린이	외국인	단체	무료함계	총계	연	월	일	요일	매출액
125	2016-05-05	O	Nan	40219	38511	1708.0	0	476	35	18469	58688	2016	5	5	목	402190000
755	2018-05-05	O	맑음	29521	28592	899.0	30	208	65	22789	52310	2018	5	5	토	295210000
390	2017-05-05	O	구름 조금	27046	25824	1222.0	0	616	170	1921	46264	2017	5	5	금	270460000
156	2016-06-05	O	구름 조금	33504	28943	1079.0	3482	729	190	823	41743	2016	6	5	일	335040000
363	2017-04-08	O	구름 조금	28514	24756	638.0	3120	220	137	1237	40887	2017	4	8	토	285140000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
656	2018-01-26	X	맑음	50	31	13.0	6	0	0	172	222	2018	1	26	금	500000
655	2018-01-25	X	맑음	23	23	0.0	0	1	0	139	162	2018	1	25	목	230000
654	2018-01-24	X	맑음	18	16	1.0	1	0	0	110	128	2018	1	24	수	180000
353	2017-03-29	X	구름 조금	0	0	0.0	0	0	0	7	2017	3	29	수	0	
352	2017-03-28	X	구름 많음	0	0	0.0	0	0	0	2	2017	3	28	화	0	

## ✓ 인덱스 재지정: reset\_index()

- 데이터프레임의 인덱스를 처음부터 다시 지정해줄 때 사용
- 데이터프레임을 정렬하는 등의 작업을 하다 보면 index가 뒤섞이게 되는데,

이 때 `reset_index()`를 활용하여 인덱스 초기화

df.reset_index()																		
	index	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일	매출액	
0	125	2016-05-05	O	NaN	40219	38511	1708.0	0	476	35	18469	58688	2016	5	5	목	402190000	
1	755	2018-05-05	O	맑음	29521	28592	899.0	30	208	65	22789	52310	2018	5	5	토	295210000	
2	390	2017-05-05	O	구름 조금	27046	25824	1222.0	0	616	170	19218	46264	2017	5	5	금	270460000	
3	156	2016-06-05	O	구름 조금	33504	28943	1079.0	3482	729	190	8239	41743	2016	6	5	일	335040000	
4	363	2017-04-08	O	구름 조금	28514	24756	638.0	3120	220	137	12373	40887	2017	4	8	토	285140000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1081	656	2018-01-26	X	맑음	50	31	13.0	6	0	0	172	222	2018	1	26	금	500000	
1082	655	2018-01-25	X	맑음	23	23	0.0	0	1	0	139	162	2018	1	25	목	230000	
1083	654	2018-01-24	X	맑음	18	16	1.0	1	0	0	110	128	2018	1	24	수	180000	
1084	353	2017-03-29	X	구름 조금	0	0	0.0	0	0	0	7	7	2017	3	29	수	0	
1085	352	2017-03-28	X	구름 많음	0	0	0.0	0	0	0	2	2	2017	3	28	화	0	

index가 초기화되면서  
기존의 index를 저장한 "index"컬럼이 생김

df.reset_index(drop=True)																		
	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일	매출액		
0	2016-05-05	O	NaN	40219	38511	1708.0	0	476	35	18469	58688	2016	5	5	목	402190000		
1	2018-05-05	O	맑음	29521	28592	899.0	30	208	65	22789	52310	2018	5	5	토	295210000		
2	2017-05-05	O	구름 조금	27046	25824	1222.0	0	616	170	19218	46264	2017	5	5	금	270460000		
3	2016-06-05	O	구름 조금	33504	28943	1079.0	3482	729	190	8239	41743	2016	6	5	일	335040000		
4	2017-04-08	O	구름 조금	28514	24756	638.0	3120	220	137	12373	40887	2017	4	8	토	285140000		
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		
1081	2018-01-26	X	맑음	50	31	13.0	6	0	0	172	222	2018	1	26	금	500000		
1082	2018-01-25	X	맑음	23	23	0.0	0	1	0	139	162	2018	1	25	목	230000		
1083	2018-01-24	X	맑음	18	16	1.0	1	0	0	110	128	2018	1	24	수	180000		
1084	2017-03-29	X	구름 조금	0	0	0.0	0	0	0	7	7	2017	3	29	수	0		
1085	2017-03-28	X	구름 많음	0	0	0.0	0	0	0	2	2	2017	3	28	화	0		

## ✓ 데이터 삭제: drop()

- 데이터프레임에서 특정 행이나 열을 삭제할 때 사용

```
df.drop("행/열 이름", axis=0 또는 1)
```

- axis=0이면 행 삭제, axis=1이면 열 삭제

```
df=df.drop(["유료합계", "무료합계"], axis=1)
```

df에서 “유료합계”, “무료합계” 컬럼(axis=1이므로)을 삭제하고 df에 저장(덮어씌우기)

	날짜	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계	연	월	일	요일	매출액
0	2016-01-01	O	구름 조금	3,59	2799	141.0	419	47	0	10,3	4382	2016	1	1	금	33590000
1	2016-01-02	O	구름 많음	5,73	4370	203.0	600	100	111	20,2	7265	2016	1	2	토	51730000
2	2016-01-03	O	구름 많음	3,08	2571	128.0	309	91	0	15,9	4557	2016	1	3	일	30080000
3	2016-01-04	X	구름 많음	90	602	Nan	235	51	223	8,0	1690	2016	1	4	월	8900000
4	2016-01-05	X	구름 많음	16	319	35.0	62	43	47	8,0	1256	2016	1	5	화	4160000



	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총계	연	월	일	요일	매출액
0	2016-01-01	O	구름 조금	2799	141.0	419	47	0	4382	2016	1	1	금	33590000
1	2016-01-02	O	구름 많음	4370	203.0	600	100	111	7265	2016	1	2	토	51730000
2	2016-01-03	O	구름 많음	2571	128.0	309	91	0	4557	2016	1	3	일	30080000
3	2016-01-04	X	구름 많음	602	Nan	235	51	223	1690	2016	1	4	월	8900000
4	2016-01-05	X	구름 많음	319	35.0	62	43	47	1256	2016	1	5	화	4160000

## ✓ 열 이름 바꾸기: rename()

- 데이터프레임의 특정 열 이름을 바꾸는데 사용할 때 사용

```
df.rename(columns = {"바꿀 컬럼이름" : "새컬럼이름"})
```

```
df=df.rename(columns = {"총계" : "총입장객수"})
```

df에서 “총계” 컬럼의 이름을 “총입장객수”로 변환한 뒤 df에 저장(덮어씌우기)

	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총계	연	월	일	요일
0	2016-01-01	O	구름 조금	2799	141.0	419	47	0	4382	2016	1	1	금
1	2016-01-02	O	구름 많음	4370	203.0	600	100	111	7265	2016	1	2	토
2	2016-01-03	O	구름 많음	2571	128.0	309	91	0	4557	2016	1	3	일
3	2016-01-04	X	구름 많음	602	Nan	235	51	223	1690	2016	1	4	월
4	2016-01-05	X	구름 많음	319	35.0	62	43	47	1256	2016	1	5	화



	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총입장객수	연	월	일	요일	매출액
0	2016-01-01	O	구름 조금	2799	141.0	419	47	0	4382	2016	1	1	금	33590000
1	2016-01-02	O	구름 많음	4370	203.0	600	100	111	7265	2016	1	2	토	51730000
2	2016-01-03	O	구름 많음	2571	128.0	309	91	0	4557	2016	1	3	일	30080000
3	2016-01-04	X	구름 많음	602	Nan	235	51	223	1690	2016	1	4	월	8900000
4	2016-01-05	X	구름 많음	319	35.0	62	43	47	1256	2016	1	5	화	4160000

## ✓ 결측치 탐색: isnull()

- 데이터프레임의 각 원소가 결측치인지 여부를 True/False로 반환할 때 사용
- sum()과 함께 사용하여 각 컬럼별로 결측치가 몇 개 존재하는지 확인하는 용도로 사용

	날짜	공휴일	날씨	유료함께	어른	청소년	어린이	외국인	단체	무료함께	총계	연	월	일	요일	매출액
0	False															
1	False															
2	False															
3	False	False	False	False	False	True	False									
4	False															
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1081	False															
1082	False															
1083	False															
1084	False															
1085	False															

df.isnull().sum()

데이터가 커서 결측치가 있는지 일일이 찾기 어려움

날짜	0
공휴일	0
날씨	140
어른	5
청소년	0
어린이	0
외국인	0
단체	0
총입장객수	0
연	0
월	0
일	0
요일	0
매출액	0
	dtype: int64

## ✓ 결측치 채우기: fillna()

- 데이터프레임 내의 결측치를 다른 값으로 대체하기 위해 사용

```
df.fillna(대체값)
```

```
df["청소년"] = df["청소년"].fillna(int(df["청소년"].mean()))
```

1. fillna를 사용하여 "청소년"컬럼에 존재하는 결측치를 채워 넣음
2. df의 "청소년"컬럼의 평균을 구한 뒤 정수로 변환
3. 결측치가 다 채워진 컬럼을 df의 "청소년"컬럼에 저장(덮어씌우기)

## ⑤ 결측치 데이터 삭제하기: dropna()

- 데이터프레임 내의 결측치가 포함된 행을 삭제하는데 사용

```
df.dropna(subset="컬럼이름", ignore_index=True 또는 False)
```

- subset: 특정 컬럼에 결측치가 존재하는지를 검사 (생략 시 전체 컬럼)
- ignore\_index: True면 결측치가 존재하는 행이 삭제된 데이터프레임의 인덱스 초기화,  
False면 그대로(기본값은 False)

```
df=df.dropna(subset=["청소년"], ignore_index=True)
```

df에서 "청소년" 컬럼에 결측치가 있는 행을 삭제한 뒤 인덱스를 초기화하여  
df에 저장(덮어씌우기)

08

## 데이터 병합하기

## ✓ 데이터 병합

- 데이터가 여러 개로 분산되어 있거나, 추가 데이터를 합치고 싶은 경우
- 데이터 병합을 통해 데이터의 일관성을 유지하고 분석 결과의 신뢰도를 높일 수 있음

index	국어	수학	영어
민수	95	88	96
도윤	72	72	68
서아	83	93	85
연우	89	80	94
하준	100	95	93



index	국어	수학	영어
민수	95	88	96
도윤	72	72	68
서아	83	93	85
연우	89	80	94
하준	100	95	93
선우	95	92	97

index	국어	수학	영어	과학
민수	95	88	96	94
도윤	72	72	68	85
서아	83	93	85	90
연우	89	80	94	90
하준	100	95	93	88

## ⑤ 데이터 합치기: concat()

- 공통의 컬럼을 가지는 데이터프레임을 합치는데 사용

```
pd.concat([데이터프레임1, 데이터프레임2], axis = 0 또는 1,  
          join = 'inner' 또는 'outer', ignore_index = True 또는 False)
```

- axis가 0이면 위아래로, 1이면 좌우로 합침 (기본값은 0)
- join이 'outer'이면 컬럼을 합집합으로, 'inner'이면 교집합으로 처리 (기본값은 outer)
- ignore\_index가 False면 원본 데이터프레임의 인덱스를 유지, True면 기존의 인덱스를 무시하고 새로운 인덱스를 부여 (기본값은 False)

## ✓ 데이터 합치기: concat()

	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총입장객수	연	월	일	요일	매출액
0	2016-01-01	O	구름 조금	2799	141.0	419	47	0	4382	2016	1	1	금	33590000
1	2016-01-02	O	구름 많음	4370	203.0	600	100	111	7265	2016	1	2	토	51730000
2	2016-01-03	O	구름 많음	2571	128.0	309	91	0	4557	2016	1	3	일	30080000
3	2016-01-04	X	구름 많음	602	463.0	235	51	223	1690	2016	1	4	월	8900000
4	2016-01-05	X	구름 많음	319	35.0	62	43	47	1256	2016	1	5	화	4160000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1081	2019-03-27	X	구름 많음	464	10.0	30	21	0	1117	2019	3	27	수	5040000
1082	2019-03-28	X	구름 많음	687	46.0	28	35	108	1665	2019	3	28	목	7610000
1083	2019-03-29	X	구름 조금	1447	120.0	77	14	188	2870	2019	3	29	금	16440000
1084	2019-03-30	O	흐림	1326	44.0	169	29	115	2452	2019	3	30	토	15390000
1085	2019-03-31	O	구름 조금	2563	111.0	387	53	0	4418	2019	3	31	일	30610000

df: 2016년 1월 1일부터 2019년 3월 31일까지의 데이터  
(폐쇄기간 제외)

	날짜	요일	공휴일	날씨	유료합계	어른	청소년	어린이	외국인	단체	무료합계	총계
0	2019-04-01	월	X	구름 조금	1,064	953	12	72	42	0	812	1,818
1	2019-04-02	화	X	구름 조금	1,020	938	4	34	36	55	1,061	1,958
2	2019-04-03	수	X	맑음	2,749	2,313	57	337	31	1651	1,208	3,813
3	2019-04-04	목	X	맑음	1,630	1,472	25	74	50	123	1,430	2755
4	2019-04-05	금	X	맑음	2,991	1,507	1306	97	21	1368	1,109	3872

df2: 2019년 4월 1일부터 2019년 4월 30일까지의 데이터

## ✓ 데이터 합치기: concat()

```
df_concat=pd.concat([df, df2], axis=0, join='inner', ignore_index=True)
```

df와 df2를 병합한 데이터프레임을 df\_concat에 저장 (axis=0이므로 세로 방향으로,  
join이 'inner'이므로 컬럼은 두 데이터프레임 모두에 포함된 컬럼만, ignore\_index는 True이므로 인덱스를 다시 부여)

	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	요일
0	2016-01-01 00:00:00	O	구름 조금	2799	141.0	419	47	0	금
1	2016-01-02 00:00:00	O	구름 많음	4370	203.0	600	100	111	토
2	2016-01-03 00:00:00	O	구름 많음	2571	128.0	309	91	0	일
3	2016-01-04 00:00:00	X	구름 많음	602	463.0	235	51	223	월
4	2016-01-05 00:00:00	X	구름 많음	319	35.0	62	43	47	화
...	...	...	...	...	...	...	...	...	...
1111	2019-04-26	X	구름 많음	1719	270.0	291	17	1911	금
1112	2019-04-27	O	구름 조금	15218	207.0	2173	125	447	토
1113	2019-04-28	O	구름 많음	10947	178.0	1780	88	455	일
1114	2019-04-29	X	흐림	1,169	47.0	130	27	370	월
1115	2019-04-30	X	NaN	1,910	378.0	381	63	2070	화

df\_concat: 2016년 1월 1일부터 2019년 4월 30일까지의 데이터  
(폐쇄기간 제외)

## ⑤ 데이터 합치기: merge()

- 특정 컬럼을 기준으로 합치는데 사용

```
pd.merge(데이터프레임1, 데이터프레임2, on="컬럼이름",
          how='inner' 또는 'outer' 또는 'left' 또는 'right')
```

- on은 데이터프레임을 합칠 때 기준(key)이 되는 컬럼의 이름
- how는 데이터프레임의 데이터(행)를 합치는 방법으로 4가지 방법이 존재
  - inner: 두 데이터프레임에 모두 존재하는 데이터만 (교집합)
  - outer: 두 데이터프레임의 모든 데이터 사용 (합집합)
  - left: 데이터프레임1에 있는 모든 데이터
  - right: 데이터프레임2에 있는 모든 데이터

 예시로 보는 `merge()`

index	월	평균 입장객수
0	3	2904
1	4	4035
2	5	5871
3	6	3661

index	월	미세먼지
0	5	85
1	6	73
2	7	54
3	8	60

✔ 예시로 보는 `merge()`

index	월	평균 입장객수
0	3	2904
1	4	4035
2	5	5871
3	6	3661

index	월	미세먼지
0	5	85
1	6	73
2	7	54
3	8	60

```
inner_df = pd.merge(df1, df2, on='월', how='inner')
```

index	월	평균 입장객수	미세먼지
0	5	5871	85
1	6	3661	73

```
outer_df = pd.merge(df1, df2, on='월', how='outer')
```

index	월	평균 입장객수	미세먼지
0	3	2904	NaN
1	4	4035	NaN
2	5	5871	85
3	6	3661	73
4	7	NaN	54
5	8	NaN	60

✔ 예시로 보는 `merge()`

index	월	평균 입장객수
0	3	2904
1	4	4035
2	5	5871
3	6	3661

index	월	미세먼지
0	5	85
1	6	73
2	7	54
3	8	60

```
left_df = pd.merge(df1, df2, on='월', how='left')
```

```
right_df = pd.merge(df1, df2, on='월', how='right')
```

index	월	평균 입장객수	미세먼지
0	3	2904	NaN
1	4	4035	NaN
2	5	5871	85
3	6	3661	73

index	월	평균 입장객수	미세먼지
0	5	5871	85
1	6	3661	73
2	7	NaN	54
3	8	NaN	60

## ✓ 데이터 합치기: merge()

	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총입장객수	연	월	일	요일
0	2016-01-01	O	구름 조금	2799	141.0	419	47	0	4382	2016	1	1	금
1	2016-01-02	O	구름 많음	4370	203.0	600	100	111	7265	2016	1	2	토
2	2016-01-03	O	구름 많음	2571	128.0	309	91	0	4557	2016	1	3	일
3	2016-01-04	X	구름 많음	602	463.0	235	51	223	1690	2016	1	4	월
4	2016-01-05	X	구름 많음	319	35.0	62	43	47	1256	2016	1	5	화
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1081	2019-03-27	X	구름 많음	464	10.0	30	21	0	1117	2019	3	27	수
1082	2019-03-28	X	구름 많음	687	46.0	28	35	108	1665	2019	3	28	목
1083	2019-03-29	X	구름 조금	1447	120.0	77	14	188	2870	2019	3	29	금
1084	2019-03-30	O	흐림	1326	44.0	169	29	115	2452	2019	3	30	토
1085	2019-03-31	O	구름 조금	2563	111.0	387	53	0	4418	2019	3	31	일

df: 2016년 1월 1일부터 2019년 3월 31일까지의  
서울대공원 입장객 데이터 (폐쇄기간 제외)

	날짜	미세먼지	초미세먼지	오존	이산화질소	일산화탄소	아황산가스
0	2016-01-01	68	45	0.006	0.047	1.0	0.006
1	2016-01-02	51	31	0.009	0.044	0.7	0.006
2	2016-01-03	96	58	0.013	0.049	0.9	0.007
3	2016-01-04	74	47	0.025	0.025	0.6	0.006
4	2016-01-05	32	16	0.016	0.025	0.4	0.006
...	...	...	...	...	...	...	...
1181	2019-03-27	94	60	0.048	0.027	0.6	0.005
1182	2019-03-28	81	49	0.034	0.027	0.5	0.004
1183	2019-03-29	39	19	0.037	0.023	0.4	0.004
1184	2019-03-30	35	24	0.037	0.018	0.4	0.003
1185	2019-03-31	42	26	0.037	0.013	0.4	0.003

mm: 2016년 1월 1일부터 2019년 3월 31일까지의  
미세먼지 데이터

## ✓ 데이터 합치기: merge()

```
df_merge=pd.merge(df, mm, on = "날짜", how = 'left')
```

df와 mm을 병합한 데이터프레임을 df\_merge에 저장  
(on이 “날짜” 이므로 “날짜” 컬럼을 기준으로, how가 ‘left’이므로 왼쪽에 위치한 df의 날짜 컬럼을 기준으로 병합)

	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총입장객수	연	월	일	요일
347	2016-12-13	X	NaN	301	40.0	8	30	0	1312	2016	12	13	화
348	2016-12-14	X	NaN	208	15.0	28	13	137	930	2016	12	14	수
349	2016-12-15	X	NaN	101	18.0	4	13	0	698	2016	12	15	목
350	2016-12-16	X	구름 조금	129	14.0	12	9	0	565	2016	12	16	금
351	2016-12-17	O	구름 많음	1087	74.0	140	24	80	2403	2016	12	17	토
352	2017-03-28	X	구름 많음	0	0.0	0	0	0	2	2017	3	28	화
353	2017-03-29	X	구름 조금	0	0.0	0	0	0	7	2017	3	29	수
354	2017-03-30	X	구름 조금	1187	13.0	102	96	115	2369	2017	3	30	목
355	2017-03-31	X	눈/비	386	328.0	13	41	270	1299	2017	3	31	금
356	2017-04-01	O	눈/비	11155	204.0	1455	100	30	17070	2017	4	1	토

- 중요한 것은 미세먼지가 아닌 입장객 수
- 서울대공원이 폐쇄된 기간의 미세먼지 데이터는 의미가 없기에 how를 left로 설정하여 입장객 데이터가 존재하는 날짜만을 남김

## ✓ 데이터 합치기: merge()

	날짜	공휴일	날씨	어른	청소년	어린이	외국인	단체	총입장객수	연	월	일	요일	미세먼지	초미세먼지	오존	이산화질소	일산화탄소	아황산가스
0	2016-01-01	O	구름 조금	2799	141.0	419	47	0	4382	2016	1	1	금	68	45	0.006	0.047	1.0	0.006
1	2016-01-02	O	구름 많음	4370	203.0	600	100	111	7265	2016	1	2	토	51	31	0.009	0.044	0.7	0.006
2	2016-01-03	O	구름 많음	2571	128.0	309	91	0	4557	2016	1	3	일	96	58	0.013	0.049	0.9	0.007
3	2016-01-04	X	구름 많음	602	463.0	235	51	223	1690	2016	1	4	월	74	47	0.025	0.025	0.6	0.006
4	2016-01-05	X	구름 많음	319	35.0	62	43	47	1256	2016	1	5	화	32	16	0.016	0.025	0.4	0.006
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1081	2019-03-27	X	구름 많음	464	10.0	30	21	0	1117	2019	3	27	수	94	60	0.048	0.027	0.6	0.005
1082	2019-03-28	X	구름 많음	687	46.0	28	35	108	1665	2019	3	28	목	81	49	0.034	0.027	0.5	0.004
1083	2019-03-29	X	구름 조금	1447	120.0	77	14	188	2870	2019	3	29	금	39	19	0.037	0.023	0.4	0.004
1084	2019-03-30	O	흐림	1326	44.0	169	29	115	2452	2019	3	30	토	35	24	0.037	0.018	0.4	0.003
1085	2019-03-31	O	구름 조금	2563	111.0	387	53	0	4418	2019	3	31	일	42	26	0.037	0.013	0.4	0.003

df\_merge: 2016년 1월 1일부터 2019년 3월 31일까지의 서울대공원 입장객 수와 미세먼지 데이터  
(폐쇄기간 제외)

## ✓ 참고: 판다스 docs 확인

### pandas.concat

```
pandas.concat(objs, *, axis=0, join='outer', ignore_index=False,
keys=None, levels=None, names=None, verify_integrity=False, sort=False,
copy=None)
```

[\[source\]](#)

Concatenate pandas objects along a particular axis.

Allows optional set logic along the other axes.

Can also add a layer of hierarchical indexing on the concatenation axis, which may be useful if the labels are the same (or overlapping) on the passed axis number.

**Parameters:** `objs : a sequence or mapping of Series or DataFrame objects`

If a mapping is passed, the sorted keys will be used as the `keys` argument, unless it is passed, in which case the values will be selected (see below). Any `None` objects will be dropped silently unless they are all `None` in which case a `ValueError` will be raised.

**axis : {0/'index', 1/'columns'}, default 0**

The axis to concatenate along.

**join : {'inner', 'outer'}, default 'outer'**

How to handle indexes on other axis (or axes).

**ignore\_index : bool, default False**

If `True`, do not use the index values along the concatenation axis. The resulting axis will be labeled 0, ..., n - 1. This is useful if you are concatenating objects where the concatenation axis does not have meaningful indexing information. Note the index values on the other axes are still respected in the join.

**keys : sequence, default None**

If multiple levels passed, should contain tuples. Construct hierarchical index using the passed keys as the outermost level.

**levels : list of sequences, default None**

Specific levels (unique values) to use for constructing a MultiIndex.

Otherwise they will be inferred from the keys.

**names : list, default None**

Names for the levels in the resulting hierarchical index.

**verify\_integrity : bool, default False**

Check whether the new concatenated axis contains duplicates. This can be very expensive relative to the actual data concatenation.

**sort : bool, default False**

Sort non-concatenation axis if it is not already aligned.

**copy : bool, default True**

If `False`, do not copy data unnecessarily.

**Returns:** `object, type of objs`

When concatenating all `Series` along the index (`axis=0`), a `Series` is returned. When `objs` contains at least one `DataFrame`, a `DataFrame` is returned. When concatenating along the columns (`axis=1`), a `DataFrame` is returned.

#### Examples

Combine two `Series`.

```
>>> s1 = pd.Series(['a', 'b'])
>>> s2 = pd.Series(['c', 'd'])
>>> pd.concat([s1, s2])
0    a
1    b
0    c
1    d
dtype: object
```

Clear the existing index and reset it in the result by setting the `ignore_index` option to `True`.

```
>>> pd.concat([s1, s2], ignore_index=True)
0    a
1    b
2    c
3    d
dtype: object
```

Add a hierarchical index at the outermost level of the data with the `keys` option.

```
>>> pd.concat([s1, s2], keys=['s1', 's2'])
s1    0    a
      1    b
s2    0    c
      1    d
dtype: object
```

Label the index keys you create with the `names` option.