
Introduction To Data Analysis
데이터분석입문
Lecture 28. 패키지 Pandas 활용
인공지능소프트웨어학과 강환수 교수

학습개요

- 패키지 pandas 개요
  - 시리즈와 데이터프레임
- 10 minutes to pandas
  - 150 여개의 pandas 훈련 문장으로 구성

학습목표

- 시리즈의 속성 name, index, values를 이해하고 활용할 수 있다.
- 데이터프레임의 index, columns를 이해하고 활용할 수 있다.
- 데이터프레임의 메소드 sort_index() sort_values() 등 다양한 메소드를 활용할 수 있다.
- 이미 생성된 데이터프레임에 새로운 열이나 행을 추가할 수 있다.

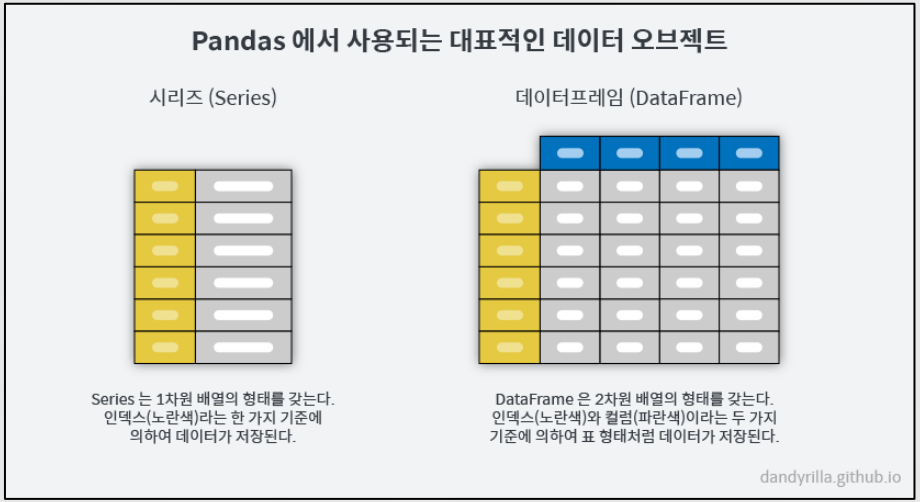
LESSON 01

패키지 pandas 개요

패키지 판다스

개요

- 표 형식의 데이터나 다양한 형태의 테이블을 처리하기 위한 라이브러리
  - 2010년부터, 약 800여명의 기여자가 활동
- 주 자료 구조
  - 시리즈와 데이터프레임



Series 속성

- name, values
- 기본 index
  - 0, 1, 2, 3, …

```
import pandas as pd

a = pd.Series([3, 5.6, 7, 9], name='basic')
```

✓ 0.0s

```
print(a.name)
print(a.values)
print(a.index)
```

✓ 0.0s

```
basic
[3.  5.6 7.  9. ]
RangeIndex(start=0, stop=4, step=1)
```

속성 index 지정

index = list

```
import pandas as pd

a = pd.Series([3, -5.6, 7, 9], index=list('abcd'), name='basic')
a
```

✓ 0.0s

```
a    3.0
b   -5.6
c    7.0
d    9.0
Name: basic, dtype: float64
```

```
print(a.name)
print(a.values)
print(a.index)
```

✓ 0.0s

```
basic
[ 3.  -5.6  7.   9. ]
Index(['a', 'b', 'c', 'd'], dtype='object')
```

DataFrame 이해

여러 시리즈의 모임

Series + Series = DataFrame

	apples
0	3
1	2
2	0
3	1

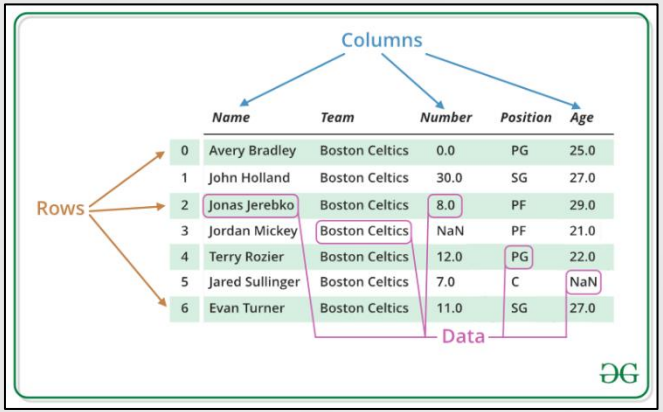
+

	oranges
0	0
1	3
2	7
3	2

=

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

인덱스와 칼럼



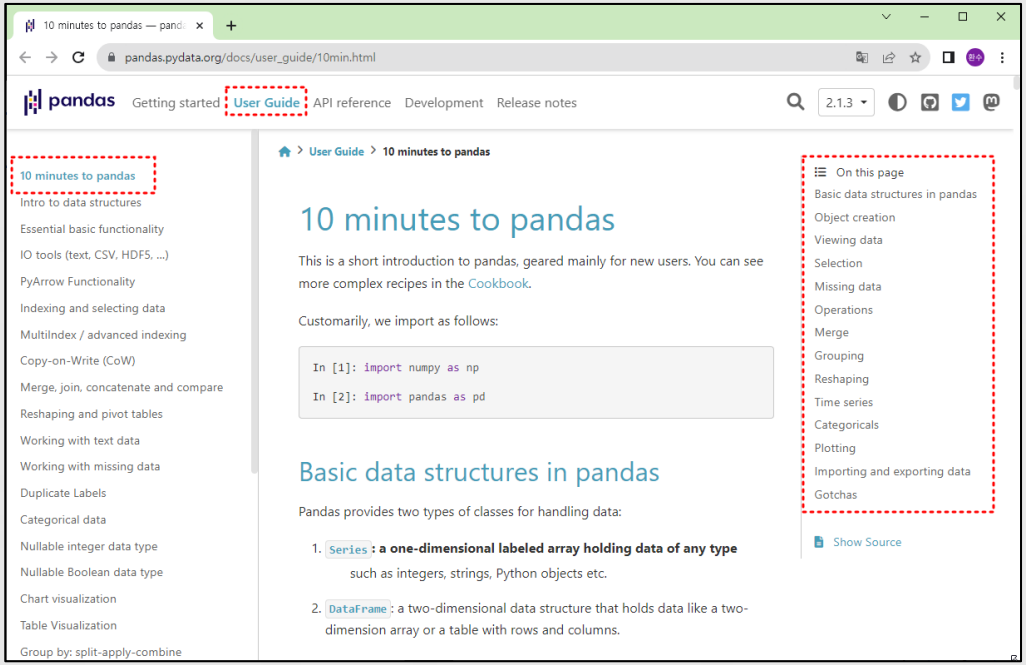
LESSON 02

10 minutes to pandas

10 minutes to pandas

실제로는 약 3시간 정도 소요

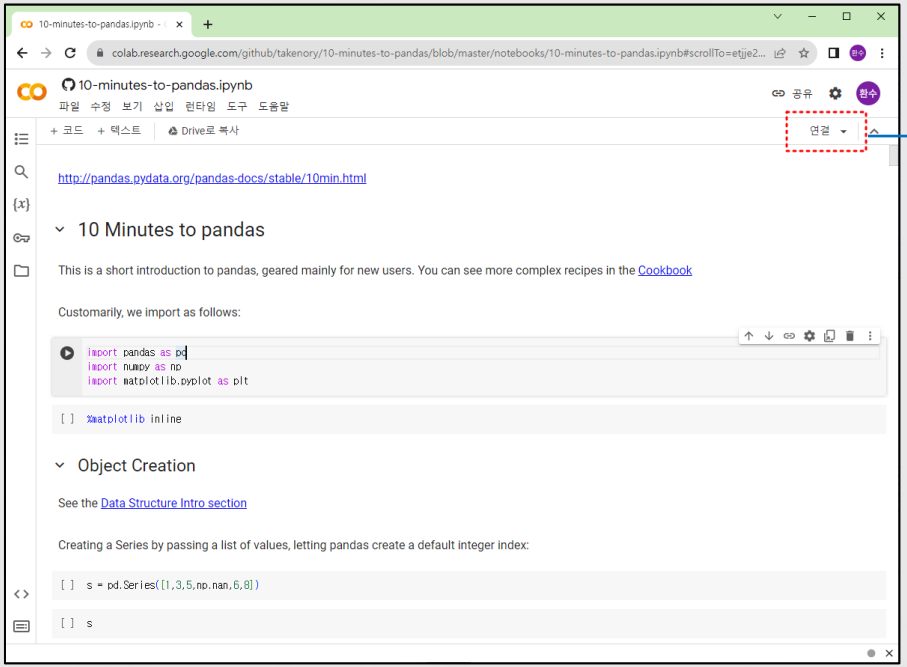
- https://pandas.pydata.org/docs/user_guide/10min.html
  - 약 150 여개의 문장



Colab of 10 minutes to pandas

구글 계정 로그인 후, 다음 사이트 접속으로 Colab에서 바로 실행 가능

- https://colab.research.google.com/github/takenory/10-minutes-to-pandas/blob/master/notebooks/10-minutes-to-pandas.ipynb



DataFrame 생성

사전으로 DataFrame의 데이터 생성

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A": 1.0,
        "B": pd.Timestamp("20260102"),
        "C": np.arange(0, 4, .5),
        "D": np.array([3] * 8, dtype="int32"),
        "E": pd.Categorical(["test", "train", "test", "train"]*2),
        "F": "foo",
    }, index=pd.date_range("20250101", periods=8)
)
df
```

✓ 0.0s

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

DataFrame의 속성 출력

속성 index, columns, dtypes

```
df.index
```

✓ 0.0s

```
DatetimeIndex(['2025-01-01', '2025-01-02', '2025-01-03', '2025-01-04',
               '2025-01-05', '2025-01-06', '2025-01-07', '2025-01-08'],
              dtype='datetime64[ns]', freq='D')
```

```
df.columns
```

✓ 0.0s

```
Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')
```

```
df.dtypes
```

✓ 0.0s

```
A           float64
B    datetime64[ns]
C           float64
D             int32
E          category
F            object
dtype: object
```

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

메소드 df.to_numpy()

인덱스 또는 열 레이블 없이 기본 데이터의 NumPy 표현을 반환

```
df.to_numpy()
```

✓ 0.0s Python

```
array([[1.0, Timestamp('2026-01-02 00:00:00'), 0.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 0.5, 3, 'train', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 1.5, 3, 'train', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 2.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 2.5, 3, 'train', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 3.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2026-01-02 00:00:00'), 3.5, 3, 'train', 'foo']],
      dtype=object)
```

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

메소드 describe()

속성이 int 또는 float인 열에 대한 기본 통계 값 표시

```
df.describe()
```

✓ 0.0s

	A	C	D
count	8.0	8.000000	8.0
mean	1.0	1.750000	3.0
std	0.0	1.224745	0.0
min	1.0	0.000000	3.0
25%	1.0	0.875000	3.0
50%	1.0	1.750000	3.0
75%	1.0	2.625000	3.0
max	1.0	3.500000	3.0

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

속성 T

데이터프레임의 전치(transpose)

```
df.T
```

✓ 0.0s Python

	2025-01-01	2025-01-02	2025-01-03	2025-01-04	2025-01-05	2025-01-06	2025-01-07	2025-01-08
A	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B	2026-01-02 00:00:00	2026-01-02 00:00:00	2026-01-02 00:00:00	2026-01-02 00:00:00	2026-01-02 00:00:00	2026-01-02 00:00:00	2026-01-02 00:00:00	2026-01-02 00:00:00
C	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
D	3	3	3	3	3	3	3	3
E	test	train	test	train	test	train	test	train
F	foo	foo	foo	foo	foo	foo	foo	foo

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

메소드 sort_index

- 키워드인자 axis=0: index 이름 순서로 정렬
- 키워드인자 axis=1: columns 이름 순서로 정렬

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

✓ 0.0s

	F	E	D	C	B	A
2025-01-01	foo	test	3	0.0	2026-01-02	1.0
2025-01-02	foo	train	3	0.5	2026-01-02	1.0
2025-01-03	foo	test	3	1.0	2026-01-02	1.0
2025-01-04	foo	train	3	1.5	2026-01-02	1.0
2025-01-05	foo	test	3	2.0	2026-01-02	1.0
2025-01-06	foo	train	3	2.5	2026-01-02	1.0
2025-01-07	foo	test	3	3.0	2026-01-02	1.0
2025-01-08	foo	train	3	3.5	2026-01-02	1.0

```
df.sort_index(axis=0, ascending=False)
```

✓ 0.0s

	A	B	C	D	E	F
2025-01-08	1.0	2026-01-02	3.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-01	1.0	2026-01-02	0.0	3	test	foo

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

메소드 sort_values

열의 내용 순서로 정렬

```
df.sort_values(by="C", ascending=False)
```

✓ 0.0s

	A	B	C	D	E	F
2025-01-08	1.0	2026-01-02	3.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-01	1.0	2026-01-02	0.0	3	test	foo

```
df.sort_values(by="E")
```

✓ 0.0s

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

	A	B	C	D	E	F
2025-01-01	1.0	2026-01-02	0.0	3	test	foo
2025-01-02	1.0	2026-01-02	0.5	3	train	foo
2025-01-03	1.0	2026-01-02	1.0	3	test	foo
2025-01-04	1.0	2026-01-02	1.5	3	train	foo
2025-01-05	1.0	2026-01-02	2.0	3	test	foo
2025-01-06	1.0	2026-01-02	2.5	3	train	foo
2025-01-07	1.0	2026-01-02	3.0	3	test	foo
2025-01-08	1.0	2026-01-02	3.5	3	train	foo

새로운 DataFrame 생성

속성 index 지정

- [x] 지정 수가 행수와 일치하도록 유의

```
dates = pd.date_range("20250101", periods=6)
dates
```

✓ 0.0s

```
DatetimeIndex(['2025-01-01', '2025-01-02', '2025-01-03', '2025-01-04',
               '2025-01-05', '2025-01-06'],
              dtype='datetime64[ns]', freq='D')
```

```
df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list("ABCD"))
df
```

✓ 0.0s

	A	B	C	D
2025-01-01	-0.031224	0.785531	0.192394	-0.452696
2025-01-02	-0.970433	1.683700	0.649518	0.502665
2025-01-03	0.836719	0.975550	1.461757	-0.789831
2025-01-04	-1.525344	0.023947	-0.157183	2.280636
2025-01-05	0.176289	1.980561	0.293979	-1.628462
2025-01-06	0.356121	1.089422	-0.705456	-1.467717

시리즈를 기존 데이터프레임에 추가

시리즈의 index를 맞게 설정

```
s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date_range("20250101", periods=6))
s1
```

✓ 0.0s

```
2025-01-01    1
2025-01-02    2
2025-01-03    3
2025-01-04    4
2025-01-05    5
2025-01-06    6
Freq: D, dtype: int64
```

```
df["F"] = s1
df
```

✓ 0.0s

	A	B	C	D	F
2025-01-01	-0.031224	0.785531	0.192394	-0.452696	1
2025-01-02	-0.970433	1.683700	0.649518	0.502665	2
2025-01-03	0.836719	0.975550	1.461757	-0.789831	3
2025-01-04	-1.525344	0.023947	-0.157183	2.280636	4
2025-01-05	0.176289	1.980561	0.293979	-1.628462	5
2025-01-06	0.356121	1.089422	-0.705456	-1.467717	6

조건과 복사

조건에 맞는 행 추출과 df.copy()

```
df[df.A > 0]
```

✓ 0.2s

	A	B	C	D	F
2025-01-03	0.836719	0.975550	1.461757	-0.789831	3
2025-01-05	0.176289	1.980561	0.293979	-1.628462	5
2025-01-06	0.356121	1.089422	-0.705456	-1.467717	6

```
df2 = df.copy()
df2
```

✓ 0.0s

	A	B	C	D	F
2025-01-01	-0.031224	0.785531	0.192394	-0.452696	1
2025-01-02	-0.970433	1.683700	0.649518	0.502665	2
2025-01-03	0.836719	0.975550	1.461757	-0.789831	3
2025-01-04	-1.525344	0.023947	-0.157183	2.280636	4
2025-01-05	0.176289	1.980561	0.293979	-1.628462	5
2025-01-06	0.356121	1.089422	-0.705456	-1.467717	6

일반 리스트로 열 추가

열.isin(list)

Series.isin([item1, item2, ···])
시리즈의 각 요소가 전달된 값 시퀀스의 요소와 정확히 일치하는지 여부를 보여주는 부울 시리즈를 반환

```
df2['E'] = ['one','one', 'two','three','four','three']
df2
```

✓ 0.0s

	A	B	C	D	F	E
2025-01-01	-0.031224	0.785531	0.192394	-0.452696	1	one
2025-01-02	-0.970433	1.683700	0.649518	0.502665	2	one
2025-01-03	0.836719	0.975550	1.461757	-0.789831	3	two
2025-01-04	-1.525344	0.023947	-0.157183	2.280636	4	three
2025-01-05	0.176289	1.980561	0.293979	-1.628462	5	four
2025-01-06	0.356121	1.089422	-0.705456	-1.467717	6	three

```
df2[df2['E'].isin(['two','four'])]
```

✓ 0.0s

	A	B	C	D	F	E
2025-01-03	0.836719	0.975550	1.461757	-0.789831	3	two
2025-01-05	0.176289	1.980561	0.293979	-1.628462	5	four

```
df2
```

✓ 0.0s

	A	B	C	D	F	E
2025-01-01	-0.031224	0.785531	0.192394	-0.452696	1	one
2025-01-02	-0.970433	1.683700	0.649518	0.502665	2	one
2025-01-03	0.836719	0.975550	1.461757	-0.789831	3	two
2025-01-04	-1.525344	0.023947	-0.157183	2.280636	4	three
2025-01-05	0.176289	1.980561	0.293979	-1.628462	5	four
2025-01-06	0.356121	1.089422	-0.705456	-1.467717	6	three

행 추가

df.append()

- 추가된 데이터프레임을 단지 반환
  - 반영을 하려면 대입문으로 처리 해야 함

```
C:\Users\PC\AppData\Local\Temp\ipykernel_22984\2047427324.py:1: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead. df.append({0:10, 1:20, 2:30, 3:40}, ignore_index=True)
```

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.randint(1, 11, (8, 4)))
df
```

✓ 0.0s

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

```
df.append({0:10, 1:20, 2:30, 3:40}, ignore_index=True)
```

✓ 0.0s

```
C:\Users\PC\AppData\Local\Temp\ipykernel_22984\2047427324.py:1: FutureWarning:
  df.append({0:10, 1:20, 2:30, 3:40}, ignore_index=True)
```

	0	1	2	3
0	2	6	2	2
1	7	5	2	2
2	7	7	2	9
3	10	9	9	3
4	7	3	4	7
5	7	8	9	1
6	5	7	9	9
7	9	6	4	3
8	10	20	30	40

pd.concat()

여러 데이터프레임을 연결해 합치는 메소드

- 합쳐진 데이터프레임을 단지 반환
  - 반영을 하려면 대입문으로 처리 해야 함

```
new_df = pd.DataFrame(np.arange(10, 41, 10).reshape(1, 4))
pd.concat([df, new_df], ignore_index=True)
```

✓ 0.0s

	0	1	2	3
0	1	6	3	7
1	8	10	4	5
2	4	5	4	1
3	6	9	5	3
4	4	8	8	6
5	7	6	5	4
6	3	6	10	8
7	4	10	9	10
8	10	20	30	40

```
df
```

✓ 0.0s

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

행 추가를 바로 반영

df.loc[len(df)] = [10, 20, 30, 40]

```
df
```

✓ 0.0s

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

```
df.loc[len(df)] = [10, 20, 30, 40]
df
```

✓ 0.0s

	0	1	2	3
0	2	3	2	3
1	7	6	9	5
2	4	4	8	2
3	6	4	4	10
4	9	4	2	7
5	1	2	5	4
6	4	3	2	8
7	9	10	9	1
8	10	20	30	40

SUMMARY

학습정리

Series 속성

- 속성 name, values, index

DataFrame 속성

- 여러 배열 생성 가능

DataFrame 주요 메소드

- df.to_numpy()
- df.describe()
- df.sort_index(axis=0): index 명으로 정렬
  - ascending=True, ascending=False
- df.sort_index(axis=1): columns 명으로 정렬
- df.sort_values(by='열명'): columns 명의 값으로 정렬
- df.copy()

DataFrame 열 추가

- df['열명'] = 시리즈 또는 리스트

DataFrame 행 추가

- df.loc[len(df)] = [10, 20, 30, 40]