



8. 판다스 (Pandas) 기초2

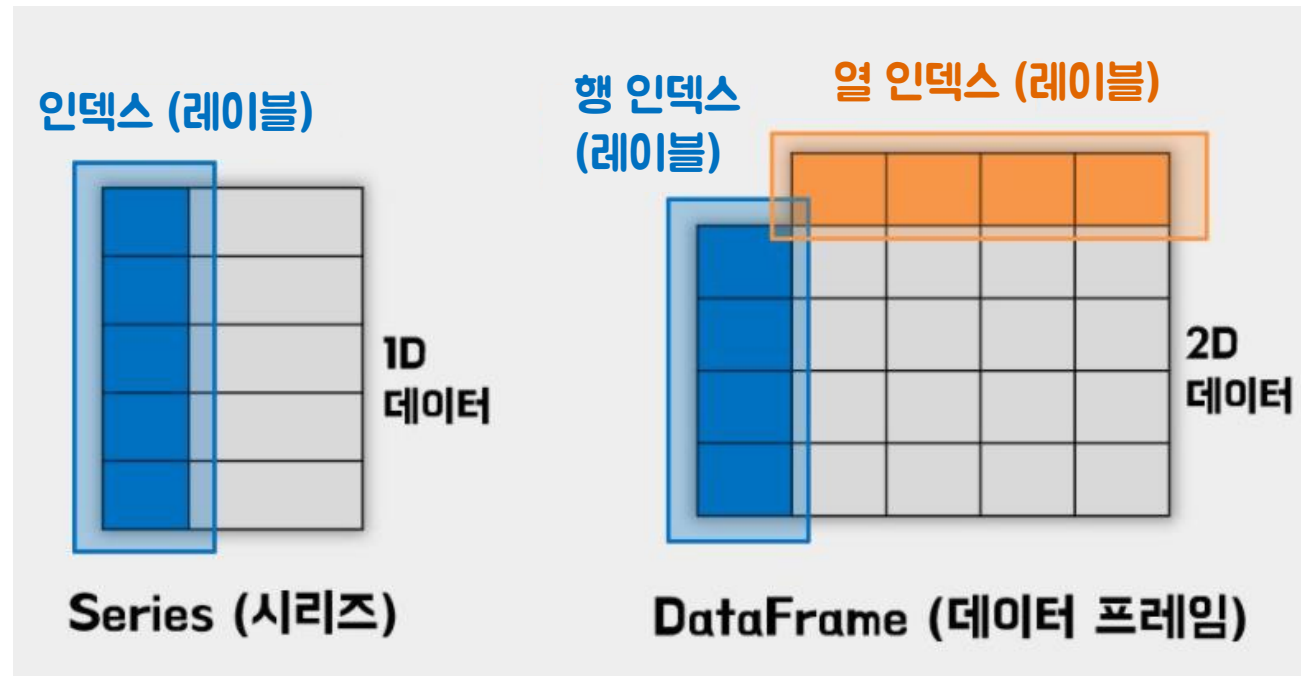
판다스 (Pandas)

- 넘파이를 기반으로 처리속도가 빠르며, 행과 열을 잘 관리할 수 있는 데이터프레임을 제공
- 이종 자료형의 열을 가진 테이블 데이터에 적합
- 데이터 처리와 분석에 필요한 기능들을 제공
 - ⊙ 결측 데이터 처리 및 필터링
 - ⊙ 열 데이터 조작 및 정렬 용이
 - ⊙ matplotlib 와 결합하여 데이터 시각화 용이

설치: `pip install pandas`

판다스 기본 자료구조

- 시리즈 (Series) : 1차원 동일 자료형 배열
- 데이터프레임 (DataFrame) : 복수의 (다른 자료형) 열로 구성된 2차원 배열



데이터프레임 예시

○ 데이터프레임 행 & 열 인덱스 레이블링

```
import pandas as pd
```

```
names = ['춘향', '몽룡', '향단', '방자']
```

```
korean = [100, 95, 90, 85]
```

```
english = [85, 90, 95, 100]
```

```
math = [90, 85, 100, 95]
```

```
df = pd.DataFrame( {'국어':korean, '영어':english, '수학':math }, index=names)
```

```
print( type(df) )
```

```
print( df )
```

```
<class 'pandas.core.frame.DataFrame'>
```

	국어	영어	수학
춘향	100	85	90
몽룡	95	90	85
향단	90	95	100
방자	85	100	95

데이터프레임 인덱싱 요약

행 인덱싱

	가능 여부	출력 자료형
레이블	X	
레이블 리스트	X	
(정수) 인덱스	X	
(정수) 슬라이스	O	데이터프레임
불리언 배열	O	데이터프레임

열 인덱싱

	가능 여부	출력 자료형
레이블	O	시리즈
레이블 리스트	O	데이터프레임
(정수) 인덱스	X	
(정수) 슬라이스	X	
불리언 배열	X	

행 추가 및 삭제

```
import pandas as pd
names = ['춘향', '몽룡', '향단', '방자']
korean = [100, 95, 90, 85]
english = [85, 90, 95, 100]
math = [90, 85, 100, 95]
df = pd.DataFrame( {'국어':korean, '영어':english, '수학':math }, index=names)
print( df )
print()
df.loc[ '학도' ] = { '국어':50, '영어':50, '수학':5 } # 행 추가
df.drop( '학도', axis=0, inplace=True )             # 행 삭제, axis=0 행 방향을 의미
# inplace=True 는 삭제를 현재 데이터프레임 내에 반영 하라는 의미
```

	국어	영어	수학
춘향	100	85	90
몽룡	95	90	85
향단	90	95	100
방자	85	100	95
학도	50	50	5

	국어	영어	수학
춘향	100	85	90
몽룡	95	90	85
향단	90	95	100
방자	85	100	95

열 추가 및 삭제

```
import pandas as pd

names = ['춘향', '몽룡', '향단', '방자']

korean = [100, 95, 90, 85]
english = [85, 90, 95, 100]
math = [90, 85, 100, 95]

df = pd.DataFrame( {'국어':korean, '영어':english, '수학':math }, index=names)
print( df )
print()

df[ '체육' ] = [ 100,100,100,100 ]          # 열 추가

df.drop( '체육', axis=1, inplace=True )      # 열 삭제, axis=1 열 방향을 의미
# inplace=True 는 삭제를 현재 데이터프레임 내에 반영 하라는 의미
```

	국어	영어	수학	체육
춘향	100	85	90	100
몽룡	95	90	85	100
향단	90	95	100	100
방자	85	100	95	100

	국어	영어	수학
춘향	100	85	90
몽룡	95	90	85
향단	90	95	100
방자	85	100	95

판다스: 데이터 개수 세기

○ count() 메서드를 사용하여 데이터 개수를 확인

⊙ 열 별로 데이터 누락 여부를 확인할 때 유용함

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
# print( df.head() )
# print()
# print(df['Species'])
print(df['Species'].count()) # 한 열의 데이터 개수
print()
print(df.count())          # 각 열마다 데이터 개수
```

```
150

caseno      150
SepalLength 150
SepalWidth  150
PetalLength 150
PetalWidth  150
Species     150
dtype: int64
```


판다스: 데이터 개수 세기

- 시리즈의 경우 `value_counts` 메서드로 각 값이 나온 횟수를 셀 수 있음
 - ⊙ 데이터프레임 은 제공하지 않음

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
print(df['Species'].value_counts())
# print(df['caseno'].value_counts())
# print(df['SepalLength'].value_counts())
```

```
virginica      50
setosa         50
versicolor     50
Name: Species, dtype: int64
```

판다스: 데이터 정렬

- `sort_index()` : 인덱스 값을 기준으로 정렬
- `sort_values()` : 데이터 값을 기준으로 정렬

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
# print(df['SepalLength'].sort_index ())
print(df['SepalLength'].sort_values())
# 열 선택 후 정렬 → 시리즈 반환
```

```
13      4.3
42      4.4
38      4.4
8       4.4
41      4.5
...
122     7.7
118     7.7
117     7.7
135     7.7
131     7.9
Name: SepalLength, Length:
150, dtype: float64
```

판다스: 데이터 정렬

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
print( df.sort_values('SepalLength') )
# 데이터프레임에서 sort_values( )
# 결과 타입도 데이터프레임
```

	caseno	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
13	14	4.3	3.0	1.1	0.1	setosa
42	43	4.4	3.2	1.3	0.2	setosa
38	39	4.4	3.0	1.3	0.2	setosa
8	9	4.4	2.9	1.4	0.2	setosa
41	42	4.5	2.3	1.3	0.3	setosa
..
122	123	7.7	2.8	6.7	2.0	virginica
118	119	7.7	2.6	6.9	2.3	virginica
117	118	7.7	3.8	6.7	2.2	virginica
135	136	7.7	3.0	6.1	2.3	virginica
131	132	7.9	3.8	6.4	2.0	virginica

판다스: 데이터 정렬

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
print( df.sort_values(
    ['SepalLength', 'SepalWidth']
))
# 데이터프레임 sort_values( [ A ,B ] )
# A 열 기준으로 정렬한 후 동일한
# 값이 나오면 B 열로 정렬
```

	caseno	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
13	14	4.3	3.0	1.1	0.1	setosa
8	9	4.4	2.9	1.4	0.2	setosa
38	39	4.4	3.0	1.3	0.2	setosa
42	43	4.4	3.2	1.3	0.2	setosa
41	42	4.5	2.3	1.3	0.3	setosa
..
118	119	7.7	2.6	6.9	2.3	virginica
122	123	7.7	2.8	6.7	2.0	virginica
135	136	7.7	3.0	6.1	2.3	virginica
117	118	7.7	3.8	6.7	2.2	virginica
131	132	7.9	3.8	6.4	2.0	virginica

[150 rows x 6 columns]

판다스: 데이터 합계

- `sum(axis)` 메소드를 사용하며 방향축 (0=행방향, 1=열방향) 을 지정할 수 있음

```
import numpy as np
import pandas as pd

df = pd.DataFrame( np.random.randint( 10, size=(3, 6) ) )
print(df)
print()
print( df.sum() ) # 방향축 지정 안하면 axis = 0 가 기본
print()
print( df.sum(axis=1) )
```

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

0	11
1	17
2	19
3	16
4	6
5	11

dtype: int64

0	27
1	29
2	24

dtype: int64

판다스: 데이터 합계

```
import numpy as np
import pandas as pd

df = pd.DataFrame( np.random.randint( 10, size=(3, 6) ) )
print(df)
print()
df['RowSum'] = df.sum(axis=1)
df.loc['ColSum'] = df.sum(axis=0)
print(df)
```

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

	0	1	2	3	4	5	RowSum
0	5.0	8.0	9.0	5.0	0.0	0.0	27.0
1	1.0	7.0	6.0	9.0	2.0	4.0	29.0
2	5.0	2.0	4.0	2.0	4.0	7.0	24.0
ColSum	11.0	17.0	19.0	16.0	6.0	11.0	80.0

행과 열에 각각 합계 결과를 추가하기

판다스: 데이터 평균

- `mean()` 은 평균을 구하며 `sum()` 과 사용법 유사함

```
import numpy as np
import pandas as pd
df=pd.read_csv( "iris.csv" )
print( df.mean() )
print()
print( df[ 'SepalLength' ].mean() )
```

```
caseno          75.500000
SepalLength      5.843333
SepalWidth       3.057333
PetalLength      3.758000
PetalWidth       1.199333
dtype: float64

5.8433333333333334
```

판다스 : 평균, 불리언 배열 함께 사용 예시

○ 각 열에 대해 평균보다 큰 값들의 불리언 배열 적용

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
# 평균보다 큰 값들에 대한 불리언 배열
df2 = df[ df > df.mean() ]
print(df2)
```

	caseno	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	NaN	NaN	3.5	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	3.2	NaN	NaN	NaN
3	NaN	NaN	3.1	NaN	NaN	NaN
4	NaN	NaN	3.6	NaN	NaN	NaN
..
145	146.0	6.7	NaN	5.2	2.3	NaN
146	147.0	6.3	NaN	5.0	1.9	NaN
147	148.0	6.5	NaN	5.2	2.0	NaN
148	149.0	6.2	3.4	5.4	2.3	NaN
149	150.0	5.9	NaN	5.1	1.8	NaN

[150 rows x 6 columns]

판다스 : 평균, 불리언 배열 함께 사용 예시

○ 각 열에 대해 평균보다 큰 값들의 개수 세기

⊙ NaN (Not a Number) 값들을 제외하고 카운트함

```
import pandas as pd
df=pd.read_csv( "iris.csv" )
print( df.count() )
print()
df2 = df[ df > df.mean() ]
# 평균보다 큰 값들에 대한 불리언 인덱싱

print( df2.count() )
# 평균보다 큰 값들의 개수
```

```
caseno      150
SepalLength 150
SepalWidth  150
PetalLength 150
PetalWidth  150
Species     150
dtype: int64
```

```
caseno      75
SepalLength 70
SepalWidth  67
PetalLength 93
PetalWidth  90
Species      0
dtype: int64
```

데이터프레임 합치기

○ 두 데이터프레임을 하나로 합치는 concat() 함수

```
import pandas as pd

df1 = pd.DataFrame(
    np.zeros(6).reshape(3, 2),
    index=['a', 'b', 'c'],
    columns=['v1', 'v2'])

df2 = pd.DataFrame(
    np.ones(4).reshape(2, 2),
    index=['a', 'c'],
    columns=['v1', 'v2'])
```

```
# print(df1)
# print(df2)
# print()
pd.concat( [df1, df2], axis=0 )
# 행 방향으로 합치기
print()
pd.concat( [df1, df2], axis=1 )
# 열 방향을 합치기
```

	v1	v2
a	0.0	0.0
b	0.0	0.0
c	0.0	0.0
a	1.0	1.0
c	1.0	1.0

	v1	v2	v1	v2
a	0.0	0.0	1.0	1.0
b	0.0	0.0	NaN	NaN
c	0.0	0.0	1.0	1.0

concat 함수의 특징

- concat 함수는 단순히 데이터프레임을 연결함 (concatenate)
- 이러한 단순 합치기는 인덱스 값이 중복될 수 있음
- 공통 행, 열 인덱스를 고려하여 합칠 수 있을까?

데이터프레임 병합

○ merge() 함수는 두 데이터프레임의 공통 열 기준으로 병합함

```
df1 = pd.DataFrame({
    '고객번호': [1001, 1002, 1003, 1004, 1005, 1006, 1007],
    '이름': ['둘리', '도우너', '또치', '길동', '희동', '마이콜', '영희']
}, columns=['고객번호', '이름'])
print(df1)

df2 = pd.DataFrame({
    '고객번호': [1001, 1001, 1005, 1006, 1008, 1001],
    '금액': [10000, 20000, 15000, 5000, 100000, 30000]
}, columns=['고객번호', '금액'])
print(df2)

df3 = pd.merge(df1, df2)
print(df3)
```

	고객번호	이름
0	1001	둘리
1	1002	도우너
2	1003	또치
3	1004	길동
4	1005	희동
5	1006	마이콜
6	1007	영희

	고객번호	금액
0	1001	10000
1	1001	20000
2	1005	15000
3	1006	5000
4	1008	100000
5	1001	30000

	고객번호	이름	금액
0	1001	둘리	10000
1	1001	둘리	20000
2	1001	둘리	30000
3	1005	희동	15000
4	1006	마이콜	5000

데이터프레임 병합

- "outer"병합 방식은 열 값이 한쪽만 있어도 합쳐서 보여줌

```
# 위에 이어서)  
df4 = pd.merge( df1, df2, how='outer' )  
print(df4)
```

	고객번호	이름	금액
0	1001	둘리	10000.0
1	1001	둘리	20000.0
2	1001	둘리	30000.0
3	1002	도우너	NaN
4	1003	또치	NaN
5	1004	길동	NaN
6	1005	희동	15000.0
7	1006	마이콜	5000.0
8	1007	영희	NaN
9	1008	NaN	100000.0


Pivot Table

- Pivot Table 이란 데이터 열 중에서 세 개의 열을 선택하여, 각각 행 인덱스, 열 인덱스, 데이터로 사용하여 데이터를 재구성하는 것

df

```
df.pivot(index='foo', columns='bar', values='baz')
```

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t



bar	A	B	C
foo			
one	1	2	3
two	4	5	6

```
data = {
    "도시": ["서울", "서울", "서울", "부산", "부산", "부산", "인천", "인천"],
    "연도": ["2015", "2010", "2005", "2015", "2010", "2005", "2015", "2010"],
    "인구": [9904312, 9631482, 9762546, 3448737, 3393191, 3512547, 2886172, 2660610],
    "지역": ["수도권", "수도권", "수도권", "경상권", "경상권", "경상권", "수도권", "수도권"]
}

columns = ["도시", "연도", "인구", "지역"]
df1 = pd.DataFrame(data, columns=columns)
print(df1)

df2=df1.pivot( "도시", "연도", "인구")
print(df2)
```

Pivot Table 예시

	도시	연도	인구	지역
0	서울	2015	9904312	수도권
1	서울	2010	9631482	수도권
2	서울	2005	9762546	수도권
3	부산	2015	3448737	경상권
4	부산	2010	3393191	경상권
5	부산	2005	3512547	경상권
6	인천	2015	2886172	수도권
7	인천	2010	2660610	수도권

연도	2005	2010	2015
도시			
부산	3512547.0	3393191.0	3448737.0
서울	9762546.0	9631482.0	9904312.0
인천	NaN	2660610.0	2886172.0

Pivot Table 예시

```
# 앞 슬라이드에 이어서
df2=df1.pivot( "도시", "연도", "인구")
print(df2)
print()
print( df2.columns )
print( df2.index )
print()
print( df2[ "2015" ] )
print()
print( df2.loc[ "서울" ] )
```

연도	2005	2010	2015
도시			
부산	3512547.0	3393191.0	3448737.0
서울	9762546.0	9631482.0	9904312.0
인천	NaN	2660610.0	2886172.0

```
Index(['2005', '2010', '2015'], dtype='object', name='연도')
```

```
Index(['부산', '서울', '인천'], dtype='object', name='도시')
```

도시	
부산	3448737.0
서울	9904312.0
인천	2886172.0

```
Name: 2015, dtype: float64
```

연도	
2005	9762546.0
2010	9631482.0
2015	9904312.0

```
Name: 서울, dtype: float64
```


Groupby

- groupby() 메소드는 특정 값에 기반하여 데이터를 그룹으로 묶을 수 있음
 - ⊙ groupby 결과에 대해 count(), mean(), sum(), describe() 등을 사용할 수 있음.

```
# 앞 슬라이드에 이어서
gb = df1.groupby('도시')
print()
print( gb.mean() )
print()
print( gb.sum() )
print()
print( gb.count() )
```

인구	
도시	
부산	3.451492e+06
서울	9.766113e+06
인천	2.773391e+06

	인구
도시	
부산	10354475
서울	29298340
인천	5546782

	연도	인구	지역
도시			
부산	3	3	3
서울	3	3	3
인천	2	2	2

Groupby

```
# 앞 슬라이드에 이어서
gb = df1.groupby('도시')
print()
des = gb.describe()
print( des )
print( des.columns )
print()
print( des[ ('인구', 'mean') ] )
```

인구	count	mean	std	...	50%	75%	max
도시				...			
부산	3.0	3.451492e+06	59725.663038	...	3448737.0	3480642.0	3512547.0
서울	3.0	9.766113e+06	136449.978473	...	9762546.0	9833429.0	9904312.0
인천	2.0	2.773391e+06	159496.419778	...	2773391.0	2829781.5	2886172.0

```
[3 rows x 8 columns]
MultiIndex([('인구', 'count'),
            ('인구', 'mean'),
            ('인구', 'std'),
            ('인구', 'min'),
            ('인구', '25%'),
            ('인구', '50%'),
            ('인구', '75%'),
            ('인구', 'max')],
           )
```

도시	
부산	3.451492e+06
서울	9.766113e+06
인천	2.773391e+06

Name: (인구, mean), dtype: float64

