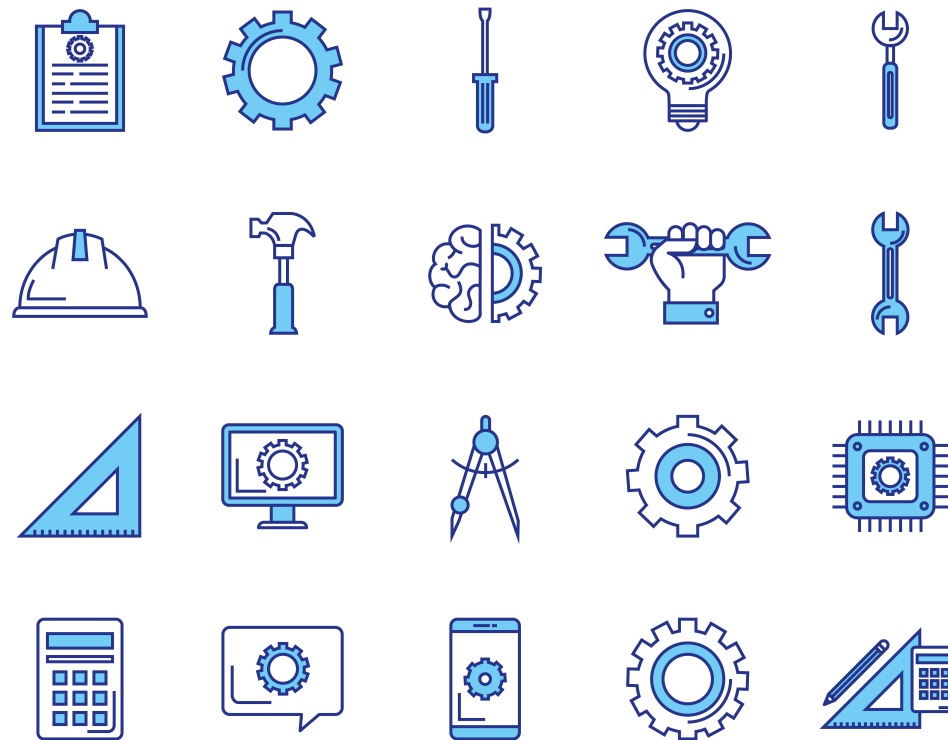


인슈어테크

파이썬 데이터 분석



이태일 강사

pandas

- 구조화된 데이터를 처리하고 저장하는 라이브러리
- Numpy n-d array를 기반으로 제작
- n-d array가 보강된 형태

pandas

Series

```
import pandas as pd
import numpy as np

print(pd.Series([1,2,3,4]))
print(pd.Series(np.array([1.,2.,3.,4.])))
```

0	1
1	2
2	3
3	4
dtype: int64	

0	1.0
1	2.0
2	3.0
3	4.0
dtype: float64	

pandas

Series

```
import pandas as pd
import numpy as np

print(pd.Series([1,2,3,4], index=["A", "B", "C", "D"]))
```

A	1
B	2
C	3
D	4
dtype: int 64	

pandas

Series

```
import pandas as pd
import numpy as np

ser = pd.Series([1,2,3,4], index=["A", "B", "C", "D"])
print(ser["A"]) # 1
```

A	1
B	2
C	3
D	4
dtype: int 64	

pandas

Series

```
population_dict = {  
    'KOR': 4000,  
    'JAP': 12700,  
    'CHN': 140000,  
    'USA': 30000  
}  
population = pd.Series(population_dict)
```

KOR	4000
JAP	12700
CHN	140000
USA	30000
dtype: int 64	

pandas

Dataframe

```
population_dict = {
    'KOR': 4000,
    'JAP': 12700,
    'CHN': 140000,
    'USA': 30000
}
gdp_dict = {
    'KOR': 16900,
    'JAP': 51600,
    'CHN': 140900,
    'USA': 204100,
}
population = pd.Series(population_dict)
gdp = pd.Series(gdp_dict)
country = pd.DataFrame({
    'population': population,
    'gdp': gdp })
```

	population	gdp
KOR	4000	16900
JAP	12700	51600
CHN	140000	140900
USA	30000	204100
population: int 64		
gdp: int 64		

pandas

Dataframe

	population	gdp
KOR	4000	16900
JAP	12700	51600
CHN	140000	140900
USA	30000	204100



	population	gdp	gni
KOR	4000	16900	?
JAP	12700	51600	?
CHN	140000	140900	?
USA	30000	204100	?

pandas

Dataframe

```
gni = pd.DataFrame({"gni":gdp/population})  
print(pd.concat([country,gni],axis=1))
```

	population	gdp	gni
KOR	4000	16900	4.225
JAP	12700	51600	4.062
CHN	140000	140900	1.006
USA	30000	204100	6.803

pandas

Dataframe

```
print(country["population"])  
print(country[["population", "gdp"]])
```

	population
KOR	4000
JAP	12700
CHN	140000
USA	30000

	population	gdp
KOR	4000	16900
JAP	12700	51600
CHN	140000	140900
USA	30000	204100

pandas

Dataframe

```
country = pd.concat([country,gni],axis=1)
country.to_csv("./country.csv")
country.to_excel("country.xlsx")
country = pd.read_csv("./country.csv")
country = pd.read_excel("country.xlsx")
```

	population	gdp	gni
KOR	4000	16900	4.225
JAP	12700	51600	4.062992
CHN	140000	140900	1.006429
USA	30000	204100	6.803333

pandas

indexing/slicing(인덱스 기반)

```
print(country.loc["KOR","gni"])  
print(country.loc["KOR":"JAP",:"gdp"])
```

	population	gdp	gni
KOR	4000	16900	4.225
JAP	12700	51600	4.062
CHN	140000	140900	1.006
USA	30000	204100	6.803

● country.loc["KOR","gni"]

● country.loc["KOR":"JAP",:"gdp"]

pandas

indexing/slicing(python 스타일)

```
print(country.iloc[2,2])  
print(country.iloc[:2,:2])
```

	population	gdp	gni
KOR	4000	16900	4.225
JAP	12700	51600	4.062
CHN	140000	140900	1.006
USA	30000	204100	6.803

● country.iloc[2,2]

● country.iloc[:2,:2]

pandas

Dataframe 데이터 추가

```
df = pd.DataFrame(columns=['이름', '나이', '직업'])
df.loc[0] = ['태일', '28', '강사']
print(df)
df.loc[1] = {'이름': '철수', '나이': '30', '직업': '개발자'}
print(df)
df.loc[1, '이름'] = '미래'
print(df)
```

	이름	나이	직업
0	태일	28	강사
1	미래	30	개발자

pandas

Dataframe 데이터 추가

```
df = pd.DataFrame(columns=['이름', '나이', '직업'])
df.loc[0] = ['태일', '28', '강사']
df.loc[1] = ['철수', '30', '개발자']
df['전화'] = np.nan
df.loc[0, '전화'] = "010"
print(df)
```

	이름	나이	직업	전화
0	태일	28	강사	010
1	미래	30	개발자	NaN

pandas

누락 데이터 체크

```
df.isnull()  
df.notnull()
```

	이름	나이	직업	전화
0	태일	28	강사	010
1	철수	30	개발자	NaN

	이름	나이	직업	전화
0	False	False	False	False
1	False	False	False	True

isnull()

	이름	나이	직업	전화
0	True	True	True	True
1	True	True	True	False

notnull()

pandas

누락 데이터 체크

```
df.dropna()  
df['전화번호'] = df['전화번호'].fillna('없음')
```

	이름	나이	직업	전화
0	태일	28	강사	010
1	철수	30	개발자	NaN

	이름	나이	직업	전화
0	태일	28	강사	010

dropna()

	이름	나이	직업	전화
0	태일	28	강사	010
1	철수	30	개발자	없음

fillna()

pandas

pandas 연산: Series

```
s1 = pd.Series([2,3,5], index=[0,1,2])
s2 = pd.Series([1,3,2], index=[1,2,3])

print(s1+s2)
print(s1.add(s2, fill_value=0))
```

* 같은 인덱스끼리 계산

** 정수+na 계산 불가로 NaN 리턴

0	NaN
1	4
2	8
3	NaN
dtype: int64	

0	2
1	4
2	8
3	2
dtype: int64	

pandas

pandas 연산: DataFrame

* 같은 인덱스끼리 계산

** 정수+na 계산 불가로 NaN 리턴

```
df1=pd.DataFrame(np.arange(4).reshape(2,2),columns=['A','B'])
df2=pd.DataFrame(np.arange(9).reshape(3,3),columns=['A','B','C'])
print(df1+df2)
print(df1.add(df2,fill_value=0)) # add,sub,mul,div
```

	A	B
0	0	1
1	2	3

	A	B	C
0	0	1	2
1	3	4	5
2	6	7	8

	A	B	C
0	0	2	NaN
1	5	7	NaN
2	NaN	NaN	NaN

	A	B	C
0	0.0	2.0	2.0
1	5.0	7.0	5.0
2	6.0	7.0	8.0

pandas

집계함수

```
df=pd.DataFrame(np.arange(9).reshape(3,3),columns=['A','B','C'])
print(df.sum())
print(df['A'].sum())
print(df.iloc[0,:].sum()) # mean
```

	A	B	C
0	0	1	2
1	3	4	5
2	6	7	8

df.sum()

A	B	C
9	12	15

df['A'].sum()

9

df.iloc[0,:].sum()

A	B	C
0	1	2

* column을 기준으로 집계

pandas

정렬

```
df = pd.DataFrame({  
    '인구': [68.6, 43.9, 54.8, 43.6, 52, 36.9, 42.9, 45.2, 46.5],  
    '지역': ['송파구', '구로구', '강남구', '서초구', '관악구', '광진구',  
            '강동구', '성북구', '양천구']  
})
```

	인구	지역
0	68.6	송파구
1	43.9	구로구
2	54.8	강남구
...		
6	42.9	강동구
7	45.2	성북구
8	46.5	양천구

pandas

정렬

```
print(df.sort_values(['지역'], ascending=True))  
df = df.sort_values(['인구'], ascending=False)
```

* ascending은 기본 True(오름차순)

** False시 역순정렬

	인구	지역
2	54.8	강남구
6	42.9	강동구
4	52	관악구

...

7	45.2	성북구
0	68.6	송파구
8	46.5	양천구

	인구	지역
0	68.6	송파구
2	54.8	강남구
4	52	관악구

...

3	43.6	서초구
6	42.9	강동구
5	36.9	광진구

pandas

정렬

```
df = df.sort_values(['인구'], ascending=False).reset_index(drop=True)
```

* 정렬시 인덱스는 변하지 않음
** reset_index로 인덱스 재 정렬

	인구	지역
0	68.6	송파구
1	54.8	강남구
2	52	관악구

...

6	43.6	서초구
7	42.9	강동구
8	36.9	광진구

pandas

	인구	지역	순위
0	68.6	송파구	
1	54.8	강남구	
2	52	관악구	

...

6	43.6	서초구	
7	42.9	강동구	
8	36.9	광진구	

?

pandas

```
rank = pd.DataFrame(np.arange(1,df.shape[0]+1),columns=['순위'])  
print(pd.concat([df,rank],axis=1))
```

concat 메소드로 이어 붙이기(axis 1기준)

	인구	지역	순위
0	68.6	송파구	1
1	54.8	강남구	2
2	52	관악구	3

...

6	43.6	서초구	7
7	42.9	강동구	8
8	36.9	광진구	9

pandas

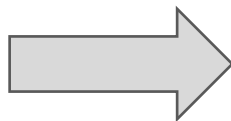
마스킹 연산: 필터링

```
df = pd.DataFrame(np.random.randint(1,5,(3,3)),columns=['A','B','C'])  
print(df)  
mask = df['A'] > 2 # 마스크 생성  
print(df[mask])
```

	A	B	C
0	3	1	1
1	3	4	3
2	1	1	3

mask

	A
0	True
1	True
2	False



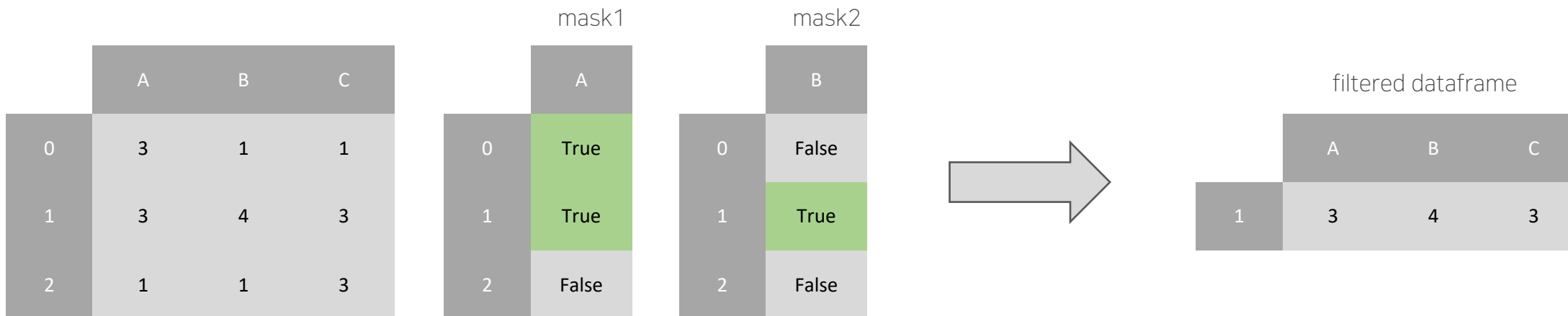
filtered dataframe

	A	B	C
0	3	1	1
1	3	4	3

pandas

마스킹 연산: 필터링

```
df = pd.DataFrame(np.random.randint(1,5,(3,3)),columns=['A','B','C'])  
print(df)  
mask1 = df['A'] > 2 # 마스크 생성  
mask2 = df['B'] > 2 # 마스크 생성  
new_mask = mask1 & mask2  
print(df[new_mask])
```



pandas

쿼리

```
df = pd.DataFrame(np.random.randint(1,5,(3,3)),columns=['A', 'B', 'C'])
print(df)
print(df.query("A>2 and B<4"))
print(df.query("A>2 or B<4"))
```

AND	OR
False	True
True	True
True	True

	A	B	C
0	2	1	4
1	3	1	2
2	3	2	4

	A	B	C
0	2	1	4
1	3	1	2
2	3	2	4

AND			
	A	B	C
1	3	1	2
2	3	2	4

OR			
	A	B	C
0	2	1	4
1	3	1	2
2	3	2	4

pandas

데이터 필터링

```
df = pd.DataFrame({  
    "이름": ['철수', '미희', '영희', '철수'],  
    "보험명": ['차', '암', '암', '실비'],  
    "수납액": ['143', '110', '125', '100']  
})
```

```
filter = df["보험명"].str.contains("암")  
print(filter)  
print(df[filter])
```

AND	OR
False	True
True	True
True	True

	이름	보험명	수납액
0	철수	차	143
1	미희	암	110
2	영희	암	125
3	철수	실비	100

	A
0	False
1	True
2	True
3	False




	이름	보험명	수납액
1	미희	암	110
2	영희	암	125

pandas

데이터 핸들링

```
def add(x):  
    return x+1  
  
df = pd.DataFrame(np.arange(5), columns=["Num"])  
print(df["Num"].apply(add))  
df["Add"] = df["Num"].apply(add)  
print(df)
```



	Num
0	0
1	1
2	2
3	3
4	4

	Num	Add
0	0	1
1	1	2
2	2	3
3	3	4
4	4	5

pandas

데이터 핸들링

```
def length(x):  
    return len(x)  
  
df = pd.DataFrame({  
    "Words": ["apple", "insurance", "talk", "key"]  
})  
print(df["Words"].apply(length))  
df["Len"] = df["Words"].apply(length)  
print(df)
```

	Words	Len
0	apple	5
1	insurance	9
2	talk	4
3	key	3

pandas

데이터 핸들링

```
DataSet = ["Doctor", "Programmer", "Lawyer", "Janitor", "Teacher"]
JobMap = {
    "Doctor": 1,
    "Programmer": 2,
    "Lawyer": 3,
    "Janitor": 4,
    "Teacher": 5,
}

df = pd.DataFrame({
    "Jobs": [DataSet[np.random.randint(0,5)] for _ in range(30)]
})
print(df["Jobs"].replace(JobMap))
```

	Jobs
0	Janitor
1	Janitor
2	Programmer

...

27	Lawyer
28	Doctor
29	Lawyer

	Jobs
0	4
1	4
2	2

...

27	3
28	1
29	3

pandas

그룹핑

```
df = pd.DataFrame({
    "Blood": ["A", "AB", "B", "A", "A", "O", "AB"],
    "Reserved": [200, 100, 150, 200, 200, 100, 200],
    "Type": ["RH+", "RH+", "RH+", "RH+", "RH-", "RH+", "RH-"],
    "Age": [13, 22, 9, 37, 40, 71, 54]
})
```

Q. A 혈액형을 가진 사람들의 나이 중간값은 ?

Q. RH+ 혈액의 총 저장 용량은 ?

	Blood	Reserved	Type	Age
0	A	200	RH+	13
1	AB	100	RH+	22
2	B	150	RH+	9
3	A	200	RH+	37
4	A	200	RH-	40
5	O	100	RH+	71
6	AB	200	RH-	54

pandas

```
df.groupby("Blood").sum()  
df.groupby(["Blood", "Type"]).median()  
df.groupby(["Blood", "Type"]).sum()
```

#1

Blood	Reserved	Age
A	600	90
AB	300	76
B	150	9
O	100	71

#2

Blood	Type	Reserved	Age
A	RH+	200	25
	RH-	200	40
AB	RH+	100	22
	RH-	200	54
B	RH+	150	9
O	RH+	100	71

#3

Blood	Type	Reserved	Age
A	RH+	400	50
	RH-	200	40
AB	RH+	100	22
	RH-	200	54
B	RH+	150	9
O	RH+	100	71

pandas

그룹 필터링

```
def filter_mean(x):  
    print(x['Age'].mean())  
    return x['Age'].mean() > 20  
  
# mean()  
# A:30, AB:38, B:9, O:71  
# 혈액형 B를 제외하는 필터  
df.groupby('Blood').filter(filter_mean)
```

	mean
A	30
AB	38
B	9
O	71

pandas

그룹 필터링

```
df = pd.read_excel("kor_univ.xlsx")
df.groupby("지역").get_group("서울")
```

지역	대학명
부산/울산	가야대
인천/경기	가천대(글로벌)
인천/경기	가천대(메디컬)
강원	가톨릭관동대
인천/경기	가톨릭대(성심)
서울	가톨릭대(성의/성신)
서울	감리교신대
인천/경기	강남대
강원	강릉원주대(강릉)
강원	강릉원주대(원주)
강원	강원대(삼척)

	지역	대학명
5	서울	가톨릭대(성의/성신)
6	서울	감리교신대
12	서울	건국대
16	서울	경기대(서울)
34	서울	경희대
37	서울	고려대
45	서울	광운대
50	서울	국민대

Q&A