/* elice */

파이썬으로 시작하는 데이터 분석

Pandas 심화 알아보기

임원균 선생님

목차

조건으로 검색하기

조건으로 검색하기

numpy array와 마찬가지로 masking 연산이 가능하다

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randint(5, 2), columns=["A", "B"])
df[“A”] < 0.5
```

	A	B
0	0.416760	0.417993
1	0.417333	0.010951
2	0.490884	0.335433
3	0.942838	0.114225
4	0.909844	0.214219

```
0     True
1     True
2     True
3    False
4    False
Name: A, dtype: bool
```

조건으로 검색하기

조건에 맞는 DataFrame row를 추출 가능하다

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randint(5, 2), columns=["A", "B"])
df[(df["A"] < 0.5) & (df["B"] > 0.3)]
df.query("A < 0.5 and B > 0.3")
```

```
0     True
1     True
2     True
3    False
4    False
Name: A, dtype: bool
```

```
0     True
1    False
2     True
3    False
4    False
Name: B, dtype: bool
```

	A	B
0	0.416760	0.417993
2	0.490884	0.335433

조건으로 검색하기

문자열이라면 다른 방식으로도 조건 검색이 가능하다

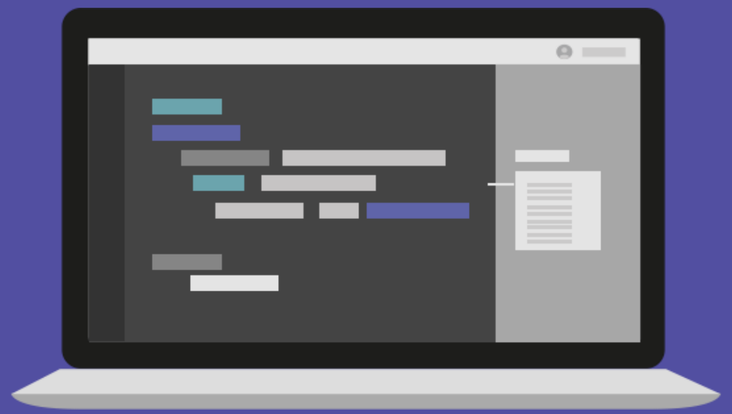
```
df["Animal"].str.contains("Cat")

df.Animal.str.match("Cat")
```

	Animal	Name
0	Dog	Happy
1	Cat	Sam
2	Cat	Toby
3	Pig	Mini
4	Cat	Rocky

```
0    False
1     True
2     True
3    False
4     True
Name: Animal, dtype: bool
```

[실습1]

조건으로 검색하기



함수로 데이터 처리하기

함수로 데이터 처리하기

apply를 통해서 함수로 데이터를 다룰 수 있다

```
df = pd.DataFrame(np.arange(5), columns=["Num"])
def square(x):
    return x**2
df["Num"].apply(square)
df["Square"] = df.Num.apply(lambda x: x ** 2)
```

	Num	Square
0	0	0
1	1	1
2	2	4
3	3	9
4	4	16

```
0     0
1     1
2     4
3     9
4    16
Name: Num, dtype: int64
```

함수로 데이터 처리하기

apply를 통해서 함수로 데이터를 다룰 수 있다

```
df = pd.DataFrame(columns=["phone"])
df.loc[0] = "010-1234-1235"
df.loc[1] = "공일공-일이삼사-1235"
df.loc[2] = "010.1234.일이삼오"
df.loc[3] = "공1공-1234.1이3오"
df["preprocess_phone"] = ''
```

	phone	preprocess_phone
0	010-1234-1235	
1	공일공-일이삼사-1235	
2	010.1234.일이삼오	
3	공1공-1234.1이3오	

함수로 데이터 처리하기

```
def get_preprocess_phone(phone):
    mapping_dict = {
        "공": "0",
        "일": "1",
        "이": "2",
        "삼": "3",
        "사": "4",
        "오": "5",
         "-": "",
        ".": "",
    }
    for key, value in mapping_dict.items():
        phone = phone.replace(key, value)
    return phone
df["preprocess_phone"] = df["phone"].apply(
      get_preprocessed_phonenumber)
```

	phone	preprocess_phone
0	010-1234-1235	01012341235
1	공일공-일이삼사-1235	01012341235
2	010.1234.일이삼오	01012341235
3	공1공-1234.1이3오	01012341235

함수로 데이터 처리하기

replace: apply 기능에서 데이터 값만 대체 하고 싶을때

```
df.Sex.replace({"Male": 0, "Female": 1})
df.Sex.replace({"Male": 0, "Female": 1}, inplace=True)
```

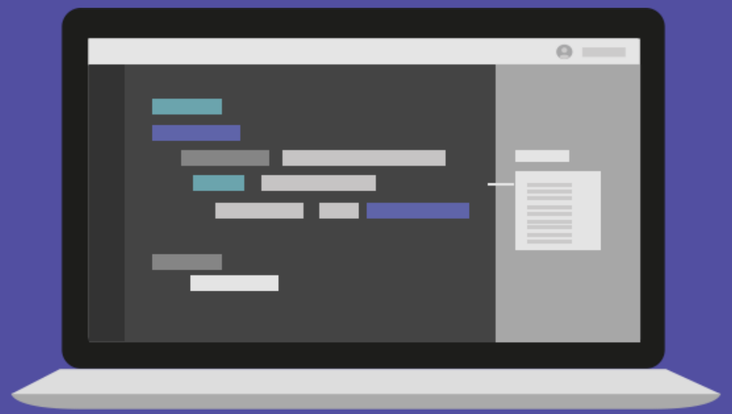
	Sex
0	Male
1	Male
2	Female
3	Female
4	Male

```
0    0
1    0
2    1
3    1
4    0
Name: Sex, dtype: int64
```

	Sex
0	0
1	0
2	1
3	1
4	0

[실습2]

함수로 데이터 처리하기



그룹으로 묶기

그룹으로 묶기

간단한 집계를 넘어서서 조건부로 집계하고 싶은 경우

```
df = pd.DataFrame({'key': ['A', 'B', 'C', 'A', 'B', 'C'],
  'data1':[1, 2, 3, 1, 2, 3], 'data2' : np.random.randint(0, 6, 6)})
df.groupby('key')
# <pandas.core.groupby.groupby.DataFrameGroupBy object at 0x10e3588>
df.groupby('key').sum()
df.groupby(['key','data1']).sum()
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

	data
key	
A	3
B	5
C	7

		data2
key	**data1**	
A	**0**	4
	3	0
B	**1**	4
	4	6
C	**2**	6
	5	1

aggregate

groupby를 통해서 집계를 한번에 계산하는 방법

```
df.groupby('key').aggregate(['min', np.median, max])
df.groupby('key').aggregate({'data1': 'min', 'data2': np.sum})
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

	data1			data2		
	min	median	max	min	median	max
key						
A	0	1.5	3	0	2.0	4
B	1	2.5	4	4	5.0	6
C	2	3.5	5	1	3.5	6

	data1	data2
key		
A	0	4
B	1	10
C	2	7

filter

groupby를 통해서 그룹 속성을 기준으로 데이터 필터링

```
def filter_by_mean(x):
    return x['data2'].mean() > 3
df.groupby('key').mean()
df.groupby('key').filter(filter_by_mean)
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

key	data1	data2
A	1.5	2.0
B	2.5	5.0
C	3.5	3.5

	data1	data2	key
1	1	4	B
2	2	6	C
4	4	6	B
5	5	1	C

apply

groupby를 통해서 묶인 데이터에 함수 적용

```
df.groupby('key').apply(lambda x: x.max() - x.min())
```

	data1	data2	key
0	0	4	A
1	1	4	B
2	2	6	C
3	3	0	A
4	4	6	B
5	5	1	C

key	data1	data2
A	3	4
B	3	2
C	3	5

get_group

groupby로 묶인 데이터에서 key값으로 데이터를 가져올 수 있다

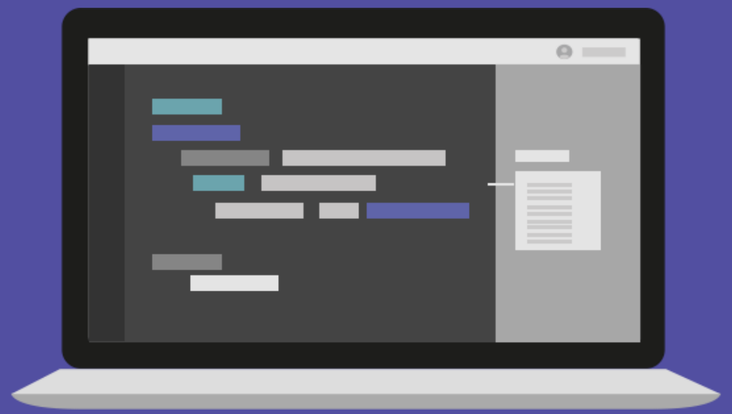
```
df = pd.read_csv("./univ.csv")
df.head()
df.groupby("시도").get_group("충남")
len(df.groupby("시도").get_group("충남"))
# 94
```

	시도	학교명
0	충남	충남도립청양대학
1	경기	한국복지대학교
2	경북	가톨릭상지대학교
3	전북	군산간호대학교
4	경남	거제대학교

	시도	학교명
0	충남	충남도립청양대학
44	충남	신성대학교
60	충남	백석문화대학교
67	충남	혜전대학교
92	충남	아주자동차대학
112	충남	천안연암대학

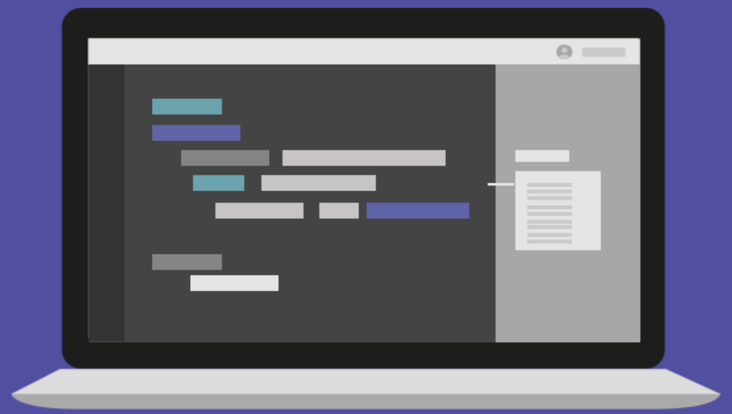
[실습3]

그룹으로 묶기 (1)

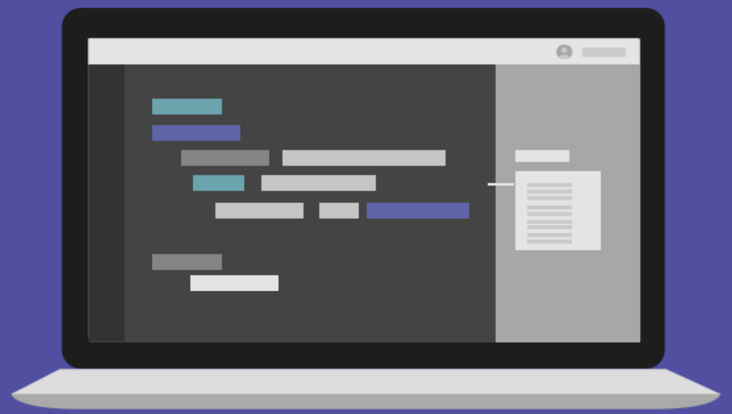


[실습3]

그룹으로 묶기 (2)



[실습3]
그룹으로 묶기 (3)



MultiIndex & pivot_table

MultiIndex

인덱스를 계층적으로 만들 수 있다

```
df = pd.DataFrame(
    np.random.randn(4, 2),
    index=[['A', 'A', 'B', 'B'], [1, 2, 1, 2]],
    columns=['data1', 'data2']
)
```

		data1	data2
A	**1**	1.374474	0.883503
	2	1.471934	-0.004420
B	**1**	0.749019	1.263473
	2	-1.302791	-0.969855

MultiIndex

열 인덱스도 계층적으로 만들 수 있다

```
df = pd.DataFrame(
    np.random.randn(4, 4),
    columns=[["A", "A", "B", "B"], ["1", "2", "1", "2"]]
)
```

	A		B	
	1	2	1	2
0	0.779620	0.089044	1.620036	-0.619421
1	-1.135626	0.219745	1.092230	-0.981231
2	0.181091	-1.089722	-0.323383	-0.586702
3	-0.171893	1.688616	-0.861637	0.156536

MultiIndex

다중 인덱스 컬럼의 경우 인덱싱은 계층적으로 한다

인덱스 탐색의 경우에는 loc, iloc를 사용가능하다

```
df[“A”]
df[“A”][“1”]
```

	A		B	
	1	2	1	2
0	0.779620	0.089044	1.620036	-0.619421
1	-1.135626	0.219745	1.092230	-0.981231
2	0.181091	-1.089722	-0.323383	-0.586702
3	-0.171893	1.688616	-0.861637	0.156536

	1	2
0	1.424545	0.810491
1	0.463098	-0.304480
2	-0.561584	-0.902295
3	-0.219135	0.235004

pivot_table

데이터에서 필요한 자료만 뽑아서 새롭게 요약,

분석 할 수 있는 기능 엑셀에서의 피봇 테이블과 같다

- Index : 행 인덱스로 들어갈 key
- Column : 열 인덱스로 라벨링될 값
- Value : 분석할 데이터

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman
3	1	1	female	35.0	1	0	53.1000	S	First	woman
4	0	3	male	35.0	0	0	8.0500	S	Third	man

pivot_table

타이타닉 데이터에서 성별과 좌석별 생존률 구하기

```
df.pivot_table(
    index='sex', columns='class', values='survived’,
    aggfunc=np.mean
)
```

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

pivot_table

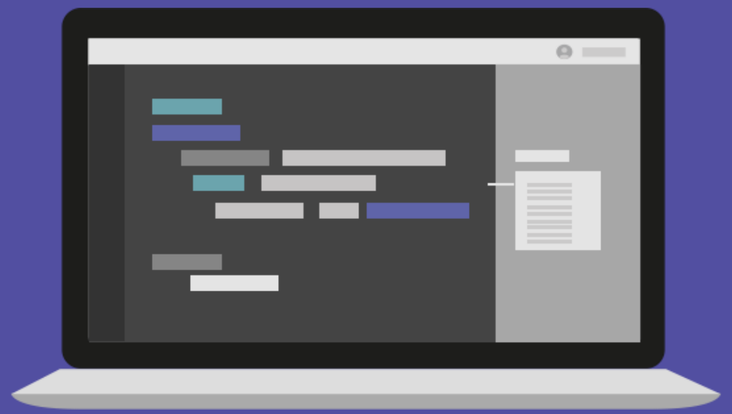
```
df.pivot_table(
    index="월별", columns='내역', values=["수입", '지출'])
```

	월별	내역	지출	수입
0	201805	관리비	200000	0
1	201805	교통비	50000	0
2	201805	월급	0	400000
3	201806	관리비	300000	0
4	201806	교통비	100000	0
5	201806	월급	0	500000
6	201807	관리비	250000	0
7	201807	교통비	150000	0
8	201807	월급	0	600000

	수입			지출		
내역	관리비	교통비	월급	관리비	교통비	월급
월별						
201805	0	0	400000	200000	50000	0
201806	0	0	500000	300000	100000	0
201807	0	0	600000	250000	150000	0

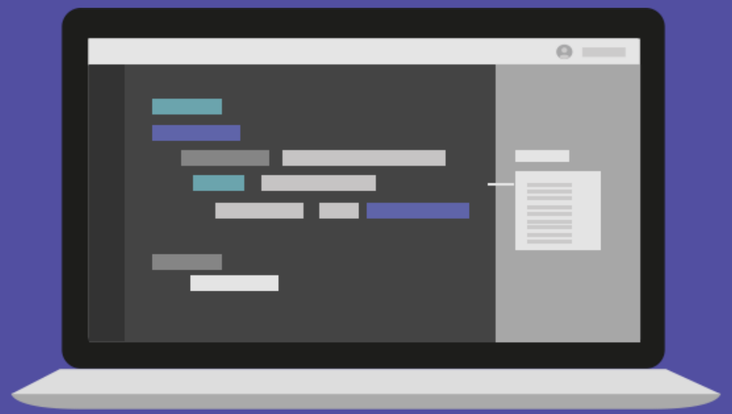
[실습4]

MultiIndex & pivot_table



[실습5]

피리 부는 사나이를 따라가는 아이들



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice