

인공지능 개론

L06.1 Pandas

국민대학교
소프트웨어융합대학원
박하명

Pandas

- Python용 데이터분석 라이브러리
- MS Office의 Excel과 같이, **행과 열로 구성된 데이터 객체**를 다룸

100% \$ % .0 .00 123 Arial				
fx	5			
	A	B	C	D
1				
2	#rounds			
3		PACC-opt4	PACC-opt3	PACC-opt2
4	livejournal	5	9	5
5	patent	6	10	6
6	friendster	6	11	6
7	skitter	5	9	5
8	subdomain	6	10	6
9	twitter	5	8	5
10	yahooweb	7	16	16
11	clueweb09	8	13	13



Pandas 사용하기

- Pandas와 Numpy 패키지 import

```
import numpy as np
import pandas as pd
```

Pandas 자료구조: Series

- **Series:** 1차원 자료구조
 - label 지정 가능
 - 다양한 종류의 데이터 타입 지원 (숫자, 문자, 파이썬 객체 등)

```
s = pd.Series([1, 3, 5, np.nan, 6, 8])
```

```
s
```

```
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

Pandas 자료구조: Series

- Series의 값만 확인하기
- 인덱스만 확인하기
- 자료형 확인하기

```
print(s.values)
print(s.index)
print(s.dtypes)
```

```
[ 1.  3.  5. nan  6.  8.]
RangeIndex(start=0, stop=6, step=1)
float64
```

Pandas 자료구조: Series

- Series의 인덱스 지정하기

```
s2 = pd.Series([2, 4, 9, -3], index=['d', 'b', 'a', 'c'])  
s2
```

```
d      2  
b      4  
a      9  
c     -3  
dtype: int64
```

Pandas 자료구조: Series

- Series의 인덱스 변경하기

```
s2.index = ['D', 'B', 'A', 'C']  
s2
```

```
D      2  
B      4  
A      9  
C     -3  
dtype: int64
```

Pandas 자료구조: DataFrame

- **DataFrame**: 2차원 자료구조
 - label 지정 가능
 - 열(Column)마다 다른 데이터 타입 사용 가능 (숫자, 문자, 파이썬 객체 등)

DataFrame

A 6x5 grid of squares. The top row consists of five dark gray squares. The remaining five rows (rows 2 to 6) each consist of one dark gray square in the first column and four light gray squares in the other columns. A horizontal line and a vertical line intersect at the square in the third row and fourth column. The label 'row' is positioned to the right of the horizontal line, and the label 'column' is positioned below the vertical line.

Pandas 자료구조: DataFrame

- DataFrame 생성하기

```
data = { '영화': ['명량', '국제시장', '베테랑', '아바타', '도둑들'],  
         '개봉년도': [2014, 2014, 2015, 2009, 2012],  
         '관객수': [17615039, 14261672, 13414136, 13302637, 12983841]}  
df = pd.DataFrame(data)  
df
```

	영화	개봉년도	관객수
0	명량	2014	17615039
1	국제시장	2014	14261672
2	베테랑	2015	13414136
3	아바타	2009	13302637
4	도둑들	2012	12983841

Pandas 자료구조: DataFrame

- 행, 열의 인덱스 확인하기
- 값만 확인하기

```
print(df.index)
print(df.columns)
print(df.values)
```

```
RangeIndex(start=0, stop=5, step=1)
Index(['영화', '개봉년도', '관객수'], dtype='object')
[['명량' 2014 17615039]
 ['국제시장' 2014 14261672]
 ['베테랑' 2015 13414136]
 ['아바타' 2009 13302637]
 ['도둑들' 2012 12983841]]
```

Pandas 자료구조: DataFrame

- DataFrame 생성시 columns과 index 지정하기

```
df2 = pd.DataFrame(data, columns=['영화', '관객수', '스크린수'],  
                    index=['A', 'B', 'C', 'D', 'E'])
```

df2

	영화	관객수	스크린수
A	명량	17615039	NaN
B	국제시장	14261672	NaN
C	베테랑	13414136	NaN
D	아바타	13302637	NaN
E	도둑들	12983841	NaN

Pandas 자료구조: DataFrame

- DataFrame 생성하기 version 2

```
data2 = [['명량', 2014, 17615039],  
         ['국제시장', 2014, 14261672],  
         ['베테랑', 2015, 13414136],  
         ['아바타', 2009, 13302637],  
         ['도둑들', 2012, 12983841]]  
  
df3 = pd.DataFrame(data2, columns=['영화', '개봉년도', '관객수'])  
df3
```

	영화	개봉년도	관객수
0	명량	2014	17615039
1	국제시장	2014	14261672
2	베테랑	2015	13414136
3	아바타	2009	13302637
4	도둑들	2012	12983841

Pandas 자료구조: DataFrame

- DataFrame 간단 통계치 확인하기
 - 계산 가능한 열에 대해서만 통계 계산

```
df2.describe()
```

관객수

count	5.000000e+00
mean	1.431546e+07
std	1.904043e+06
min	1.298384e+07
25%	1.330264e+07
50%	1.341414e+07
75%	1.426167e+07
max	1.761504e+07

DataFrame 다루기

- 열 선택하기
 - 해당 열이 Series 타입으로 선택됨

```
df['관객수']
```

```
df.관객수
```

```
0    17615039
1    14261672
2    13414136
3    13302637
4    12983841
Name: 관객수, dtype: int64
```

DataFrame 다루기

- 열 여러개 한꺼번에 선택하기

```
df[['year','points']]
```

	영화	개봉년도
0	명량	2014
1	국제시장	2014
2	베테랑	2015
3	아바타	2009
4	도둑들	2012

DataFrame 다루기

- 새로운 열 추가하고 값 대입하기
 - Python List, Pandas Series, Numpy array, 값 등을 대입 가능

```
df['평점'] = [4.5, 4.0, 4.0, 3.0, 2.5]
```

```
df['전문가평점'] = 3.5
```

```
df
```

	영화	개봉년도	관객수	평점	전문가평점
0	명량	2014	17615039	4.5	3.5
1	국제시장	2014	14261672	4.0	3.5
2	베테랑	2015	13414136	4.0	3.5
3	아바타	2009	13302637	3.0	3.5
4	도둑들	2012	12983841	2.5	3.5

DataFrame 다루기

- 열 연산하기
 - 연산 결과는 Series 임

```
df['평점차이'] = df['평점'] - df['전문가평점']  
df['높은평점'] = df['평점'] > 3.5  
df
```

	영화	개봉년도	관객수	평점	전문가평점	평점차이	높은평점
0	명량	2014	17615039	4.5	3.5	1.0	True
1	국제시장	2014	14261672	4.0	3.5	0.5	True
2	베테랑	2015	13414136	4.0	3.5	0.5	True
3	아바타	2009	13302637	3.0	3.5	-0.5	False
4	도둑들	2012	12983841	2.5	3.5	-1.0	False

DataFrame 다루기

- 열 삭제하기

```
del df['평점차이']  
del df['높은평점']  
df
```

	영화	개봉년도	관객수	평점	전문가평점
0	명량	2014	17615039	4.5	3.5
1	국제시장	2014	14261672	4.0	3.5
2	베테랑	2015	13414136	4.0	3.5
3	아바타	2009	13302637	3.0	3.5
4	도둑들	2012	12983841	2.5	3.5

DataFrame 다루기

- 행 번호로 행 선택하기

```
df[1:3]
```

```
df.iloc[1:3]
```

	영화	개봉년도	관객수	평점	전문가평점
1	국제시장	2014	14261672	4.0	3.5
2	베테랑	2015	13414136	4.0	3.5

DataFrame 다루기

- 행 이름으로 행 선택하기

```
df.index = ['A', 'X', 'T', 'B', 'C']  
df['X':'B']
```

```
df.loc['X':'B']
```

	영화	개봉년도	관객수	평점	전문가평점
X	국제시장	2014	14261672	4.0	3.5
T	베테랑	2015	13414136	4.0	3.5
B	아바타	2009	13302637	3.0	3.5

DataFrame 다루기

- 행, 열 이름으로 행, 열 같이 선택하기

```
df.loc['X':'T', '영화':'개봉년도']
```

영화		개봉년도
X	국제시장	2014
T	베테랑	2015

DataFrame 다루기

- 행, 열 이름으로 행, 열 같이 선택하기

```
df.loc[:, ['영화', '평점']]
```

	영화	평점
A	명량	4.5
X	국제시장	4.0
T	베테랑	4.0
B	아바타	3.0
C	도둑들	2.5

DataFrame 다루기

- 행 추가하기

```
df.loc['W', :] = ['7번방의 선물', 2013, 12811213, 4.4, 4.0]  
df
```

	영화	개봉년도	관객수	평점	전문가평점
A	명량	2014.0	17615039.0	4.5	3.5
X	국제시장	2014.0	14261672.0	4.0	3.5
T	베테랑	2015.0	13414136.0	4.0	3.5
B	아바타	2009.0	13302637.0	3.0	3.5
C	도둑들	2012.0	12983841.0	2.5	3.5
W	7번방의 선물	2013.0	12811213.0	4.4	4.0

DataFrame 다루기

- 행 지우기

```
df.drop('W')
```

	영화	개봉년도	관객수	평점	전문가평점
A	명량	2014.0	17615039.0	4.5	3.5
X	국제시장	2014.0	14261672.0	4.0	3.5
T	베테랑	2015.0	13414136.0	4.0	3.5
B	아바타	2009.0	13302637.0	3.0	3.5
C	도둑들	2012.0	12983841.0	2.5	3.5

DataFrame 다루기

- 행, 열 번호로 행, 열 선택하기

```
df.iloc[3:5, 0:2]
```

	영화	개봉년도
B	아바타	2009.0
C	도둑들	2012.0

```
df.iloc[[0,1,3], [1,2]]
```

	개봉년도	관객수
A	2014.0	17615039.0
X	2014.0	14261672.0
B	2009.0	13302637.0

DataFrame 다루기

- 조건에 맞는 행, 열 찾기
 - 예) 평점이 3.5점 이상인 영화 모두 찾기

```
df.loc[df['평점']>3.5, :]
```

	영화	개봉년도	관객수	평점	전문가평점
A	명량	2014.0	17615039.0	4.5	3.5
X	국제시장	2014.0	14261672.0	4.0	3.5
T	베테랑	2015.0	13414136.0	4.0	3.5
W	7번방의 선물	2013.0	12811213.0	4.4	4.0

DataFrame 다루기

- 조건에 맞는 행, 열 찾기
 - 예) 영화 베테랑의 관객수 찾기

```
df.loc[df['영화'] == '베테랑', ['영화', '관객수']]
```

	영화	관객수
T	베테랑	13414136.0

DataFrame 다루기

- 조건에 맞는 행, 열 찾기
 - 예) 2014년 이전에 개봉한 영화중 평점이 3.5점을 넘는 영화 찾기

```
df.loc[(df['개봉년도'] < 2014) & (df['평점'] > 3.5), :]
```

	영화	개봉년도	관객수	평점	전문가평점
W	7번방의 선물	2013.0	12811213.0	4.4	4.0

- 2014년 이후 영화이거나 평점이 3.5 이상인 영화를 찾으려면?

DataFrame 연산하기

- 데이터 새로 만들기

```
data = [[np.nan, 2.3],  
        [-2.3, 4.5],  
        [np.nan, np.nan],  
        [1.00, -6.2]]  
df = pd.DataFrame(data, columns=["x", "y"], index=["a", "b", "c", "d"])  
df
```

	x	y
a	NaN	2.3
b	-2.3	4.5
c	NaN	NaN
d	1.0	-6.2

DataFrame 연산하기

- 행, 열 합계 구하기

```
df.sum(axis=0)
```

```
x  -1.3  
y   0.6  
dtype: float64
```

```
df.sum(axis=1)
```

```
a   2.3  
b   2.2  
c   0.0  
d  -5.2  
dtype: float64
```

DataFrame 연산하기

- 특정 행, 열 합계 구하기

```
df['x'].sum()
```

-1.29999999999999998

```
df.loc['b'].sum()
```

2.2

- sum(): 합계
- mean(): 평균
- min(): 최소값
- max(): 최대값
- count(): NaN이 아닌 값의 개수

DataFrame 연산하기

- NaN 다루기 - NaN을 포함하는 행, 열 지우기

```
df.dropna(how='any')
```

```
df.dropna(how='all')
```

	x	y
a	NaN	2.3
b	-2.3	4.5
c	NaN	NaN
d	1.0	-6.2

	x	y
b	-2.3	4.5
d	1.0	-6.2

	x	y
a	NaN	2.3
b	-2.3	4.5
d	1.0	-6.2

DataFrame 연산하기

- NaN 다루기 - NaN 자리에 다른 값 채우기, NaN인지 확인하기

```
df.fillna(0)
```

```
df.isnull()
```

	x	y
a	NaN	2.3
b	-2.3	4.5
c	NaN	NaN
d	1.0	-6.2

	x	y
a	0.0	2.3
b	-2.3	4.5
c	0.0	0.0
d	1.0	-6.2

	x	y
a	True	False
b	False	False
c	True	True
d	False	False

DataFrame 연산하기

- NaN 다루기 - 특정 열에 NaN을 포함하는 행 선택하기

```
df.loc[df.isnull()['x'], :]
```

	x	y
a	NaN	2.3
b	-2.3	4.5
c	NaN	NaN
d	1.0	-6.2

	x	y
a	NaN	2.3
c	NaN	NaN

- 특정 행에 NaN을 포함하는 열을 선택하려면?

DataFrame 정렬하기

- 새로 데이터 생성

```
df2 = pd.DataFrame(np.random.randn(6, 5),  
                    columns=["A", "B", "C", "D", "E"],  
                    index=pd.date_range("20200425", periods=6))
```

df2

	A	B	C	D	E
2020-04-25	-0.194166	-0.496491	-0.213415	0.464056	1.110668
2020-04-26	1.613261	0.041508	0.313409	0.035532	2.210315
2020-04-27	0.470258	-1.071416	-0.127556	0.095329	0.476436
2020-04-28	0.557931	-2.004822	-1.494962	1.842903	-0.288499
2020-04-29	0.498283	1.008972	-0.619039	1.038682	0.255232
2020-04-30	-0.353168	-1.118425	-0.311908	-0.063403	0.180001

DataFrame 정렬하기

- Index와 Column 순서 섞기

```
df2.columns = np.random.permutation(df2.columns)
df2.index = np.random.permutation(df2.index)
df2
```

	B	D	C	E	A
2020-04-25	-0.194166	-0.496491	-0.213415	0.464056	1.110668
2020-04-27	1.613261	0.041508	0.313409	0.035532	2.210315
2020-04-28	0.470258	-1.071416	-0.127556	0.095329	0.476436
2020-04-29	0.557931	-2.004822	-1.494962	1.842903	-0.288499
2020-04-26	0.498283	1.008972	-0.619039	1.038682	0.255232
2020-04-30	-0.353168	-1.118425	-0.311908	-0.063403	0.180001

DataFrame 정렬하기

- Index 정렬하기

```
df2.sort_index(axis=0)
```

	B	D	C	E	A
2020-04-25	-0.194166	-0.496491	-0.213415	0.464056	1.110668
2020-04-26	0.498283	1.008972	-0.619039	1.038682	0.255232
2020-04-27	1.613261	0.041508	0.313409	0.035532	2.210315
2020-04-28	0.470258	-1.071416	-0.127556	0.095329	0.476436
2020-04-29	0.557931	-2.004822	-1.494962	1.842903	-0.288499
2020-04-30	-0.353168	-1.118425	-0.311908	-0.063403	0.180001

DataFrame 정렬하기

- Column 정렬하기

```
df2.sort_index(axis=1)
```

	A	B	C	D	E
2020-04-25	1.110668	-0.194166	-0.213415	-0.496491	0.464056
2020-04-27	2.210315	1.613261	0.313409	0.041508	0.035532
2020-04-28	0.476436	0.470258	-0.127556	-1.071416	0.095329
2020-04-29	-0.288499	0.557931	-1.494962	-2.004822	1.842903
2020-04-26	0.255232	0.498283	-0.619039	1.008972	1.038682
2020-04-30	0.180001	-0.353168	-0.311908	-1.118425	-0.063403

DataFrame 정렬하기

- 내림차순 정렬하기

```
df2.sort_index(axis=0)
```

	B	D	C	E	A
2020-04-30	-0.353168	-1.118425	-0.311908	-0.063403	0.180001
2020-04-29	0.557931	-2.004822	-1.494962	1.842903	-0.288499
2020-04-28	0.470258	-1.071416	-0.127556	0.095329	0.476436
2020-04-27	1.613261	0.041508	0.313409	0.035532	2.210315
2020-04-26	0.498283	1.008972	-0.619039	1.038682	0.255232
2020-04-25	-0.194166	-0.496491	-0.213415	0.464056	1.110668

DataFrame 정렬하기

- 값을 기준으로 정렬하기

```
df2.sort_values(by='B')
```

	B	D	C	E	A
2020-04-30	-0.353168	-1.118425	-0.311908	-0.063403	0.180001
2020-04-25	-0.194166	-0.496491	-0.213415	0.464056	1.110668
2020-04-28	0.470258	-1.071416	-0.127556	0.095329	0.476436
2020-04-26	0.498283	1.008972	-0.619039	1.038682	0.255232
2020-04-29	0.557931	-2.004822	-1.494962	1.842903	-0.288499
2020-04-27	1.613261	0.041508	0.313409	0.035532	2.210315

Question?