

데이터처리프로그래밍

데이터분석도구(Pandas)



강원대학교 교육혁신원 송혜정
<hjsong@kangwon.ac.kr>



Pandas

✓ 학습목표

- 빅데이터 분석 도구인 파이썬 라이브러리 pandas를 이해한다.

✓ 학습내용

- Indexing / Selection / Viewing
- Setting / Addition / Deletion



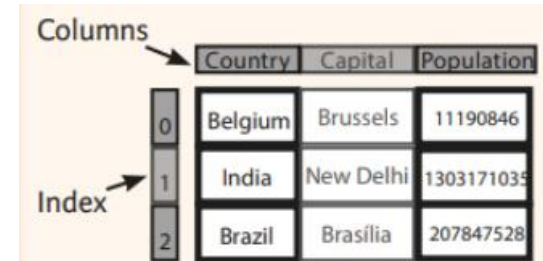
강의에 앞서서..

- 본 강의자료는 아래의 자료들을 참고하여 만들어 졌음을 알립니다

1. Pandas (<https://pandas.pydata.org/pandas-docs/stable/index.html>)

DataFrame

(3)DataFrame, Indexing / Selection



Columns		Country	Capital	Population
Index	0	Belgium	Brussels	11190846
	1	India	New Delhi	1303171035
	2	Brazil	Brasília	207847528

Operation	Syntax	Result
Select column	df[col]	Series
Select row by label	df.loc[label]	Series
Select row by integer location	df.iloc[loc]	Series
Slice rows	df[5:10]	DataFrame
Select rows by boolean vector	df[bool_vec]	DataFrame

```
#DataFrame, Indexing / Selection
d = {'one': pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
      'two': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),
      'three': [5,6,7,8]}
df1 = pd.DataFrame(d)
display("df1=", df1)
```

```
#DataFrame, row selection
display("df1[0:2]=", df1[0:2])
display("df1[['a':'c']]=", df1[['a':'c']])

display("df1.loc['a']=", df1.loc['a'])
display("df1.loc['b:']=", df1.loc['b:'])
display("df1.loc[['a','c']]=", df1.loc[['a','c']])

display("df1.iloc[1]=", df1.iloc[1])
display("df1.iloc[1:3]=", df1.iloc[1:3])
```

#위치값 슬라이싱으로 선택
#행 레이블 슬라이싱으로 선택

#행 레이블로 선택
#슬라이싱으로 여러행 선택
#인덱스 배열로 여러 행 선택

#행인덱스값으로 선택

```
'df1='
      one two three
a  1.0  1.0   5
b  2.0  2.0   6
c  3.0  3.0   7
d  NaN  4.0   8
```

```
'df1[0:2]='
      one two three
a  1.0  1.0   5
b  2.0  2.0   6
```

```
"df1[['a':'c']]="
      one two three
a  1.0  1.0   5
b  2.0  2.0   6
c  3.0  3.0   7
```

```
"df1.loc['a']="
one    1.0
two    1.0
three   5.0
Name: a, dtype: float64
```

```
"df1.loc['b:']="
      one two three
b  2.0  2.0   6
c  3.0  3.0   7
d  NaN  4.0   8
```

```
"df1.loc[['a','c']]="
      one two three
a  1.0  1.0   5
c  3.0  3.0   7
```

```
'df1.iloc[1]='
one    2.0
two    2.0
three   6.0
Name: b, dtype: float64
```

```
'df1.iloc[1:3]='
      one two three
b  2.0  2.0   6
c  3.0  3.0   7
```



DataFrame

(3)DataFrame, Indexing / Selection

```
#DataFrame , Column selection,
display("df1['one']= ", df1['one'] )           #열레이블로 선택
display("df1.one= ", df1.one )                 #열레이블로 선택
display("df1[['one','three']]= ", df1[['one','three']] ) #열레이블 여러개 인덱스 배열로 선택
```

```
"df1['one']= "
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
```

```
'df1.one= '
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
```

```
"df1[['one','three']]= "
```

	one	three
a	1.0	5
b	2.0	6
c	3.0	7
d	NaN	8

DataFrame

(3)DataFrame, Indexing / Selection

```
#DataFrame, row, column selection,
display("df1.loc['a', 'one']=", df1.loc['a', 'one'])           #'a'행, 'one' 열
display("df1.loc[['a','c'], 'one']=", df1.loc[['a','c'], 'one']) #'a','c'행, 'one' 열
display("df1.loc['a':'c', 'one':'two']=", df1.loc['a':'c', 'one':'two']) #'a':'c'행, 'one':'two'열
display("df1.loc[:, ['one','three']=", df1.loc[:, ['one','three']]) #['one','three']열의 모든행

display("df1['two']['a']=", df1['two']['a']) #열레이블, 행레이블로 선택
display("df1['two']['b:']=", df1['two']['b:']) #열레이블, 행레이블 슬라이싱
display("df1['two'][2:]=", df1['two'][2:]) #열레이블, 행슬라이싱으로 선택
```

```
"df1.loc['a', 'one']="
```

```
1.0
```

```
"df1.loc[['a','c'], 'one']="
```

```
a    1.0
```

```
c    3.0
```

```
Name: one, dtype: float64
```

```
"df1.loc['a':'c', 'one':'two']="
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0

```
"df1.loc[:, ['one','three']="
```

	one	three
a	1.0	5
b	2.0	6
c	3.0	7
d	NaN	8

```
"df1['two']['a']="
```

```
1.0
```

```
"df1['two']['b:']="
```

```
b    2.0
```

```
c    3.0
```

```
d    4.0
```

```
Name: two, dtype: float64
```

```
"df1['two'][2:]="
```

```
c    3.0
```

```
d    4.0
```

```
Name: two, dtype: float64
```

DataFrame

(3)DataFrame, Indexing / Selection

```
#DataFrame, bool indexing, 조건으로 인덱싱
#one 열의 값이 2이상인 경우
display("df1[df1.one >=2] = ", df1[df1.one >=2] )
#one 열의 값이 1 ~ 2 범위인 경우
display("df1[(df1.one >= 1) & (df1.one <= 2)] = ", df1[(df1.one >= 1) & (df1.one <=2)] )
#one 열의 값이 2보다 작거나 3보다 큰 경우
display("df1[(df1.one < 2) | (df1.one > 3)] = ", df1[(df1.one < 2) | (df1.one > 3)] )
# one이 NaN인 경우
display("pd.isna(df1.one) = ", pd.isna(df1.one) )
```

```
'df1[df1.one >=2] = '
```

	one	two	three
b	2.0	2.0	6
c	3.0	3.0	7

```
'df1[(df1.one >= 1) & (df1.one <= 2)] = '
```

	one	two	three
a	1.0	1.0	5
b	2.0	2.0	6

```
'df1[(df1.one < 2) | (df1.one > 3)] = '
```

	one	two	three
a	1.0	1.0	5

```
'pd.isna(df1.one) = '
```

```
a    False
b    False
c    False
d     True
Name: one, dtype: bool
```

DataFrame

예제 1. 데이터프레임 인덱싱 연습

```
#5명의 성적 데이터 생성 (성명, 입학년도, 학점(0~4.5), 성별(M,F))
data = {'Name' : ['kim', 'lee', 'min', 'jin', 'park'],
        'Adyear' : [2017, 2017, 2018, 2019, 2018],
        'Grade' : [3.5, 2.0, 4.5, 2.2, 4.0],
        'Sex' : ['M', 'M', 'F', 'F', 'M']}
df = pd.DataFrame(data)

#1)성명만 추출
r1 = df['Name']
display("성명만 추출", r1)

#2)입학년도와 성적만 추출
r2 = df[['Adyear', 'Grade']]
#r2 = df.loc[:, ['Adyear', 'Grade']]
display("입학년도와 성적만 추출", r2)

#3)성명이 'kim'인 데이터만 추출
r3 = df[df.Name == 'kim']
display("성명이 'kim'인 데이터만 추출", r3)

#4)입학년도가 2018년도 이상만 추출
r4 = df[df.Adyear >= 2018]
display("입학년도가 2018년도이상만 추출", r4)

#5)여성만 추출
r5 = df[df.Sex == 'F']
display("여성만 추출", r5)

#6)여성이면서 입학년도가 2019년도만 추출
r6 = df[(df.Sex == 'F') & (df.Adyear == 2019)]
display("여성이면서 입학년도가 2019만 추출", r6)

#7)성적이 3.5이상이거나 남성의 경우 추출
r7 = df[(df.Sex == 'M') | (df.Grade >= 3.5)]
display("성적이 3.5이상이거나 남성인 경우 추출", r7)
```


DataFrame

(4) DataFrame, setting

```
#DataFrame 생성
data = np.zeros((5,3))
idx= list('abcde')
col = list('XYZ')
df = pd.DataFrame(data, index=idx, columns=col)
display(df)

#행 슬라이싱 인덱스로 값설정
df[3:] = 1
df['a':'b'] = 2
display(df)
```

	X	Y	Z
a	0.0	0.0	0.0
b	0.0	0.0	0.0
c	0.0	0.0	0.0
d	0.0	0.0	0.0
e	0.0	0.0	0.0

	X	Y	Z
a	2.0	2.0	2.0
b	2.0	2.0	2.0
c	0.0	0.0	0.0
d	1.0	1.0	1.0
e	1.0	1.0	1.0

#열 레이블을 기준으로 값설정

```
df['X']['a'] = 10
df['X']['c':] = 20
df['Y']['a','c'] = 30
display(df)
```

#행 열 레이블로 값설정

```
df.at[['c','e'], 'X':'Y'] = 100
df.loc['d', :2] = -100
display(df)
```

	X	Y	Z
a	10.0	30.0	2.0
b	2.0	2.0	2.0
c	20.0	30.0	0.0
d	20.0	1.0	1.0
e	20.0	1.0	1.0

	X	Y	Z
a	10.0	30.0	2.0
b	2.0	2.0	2.0
c	100.0	100.0	0.0
d	-100.0	-100.0	1.0
e	100.0	100.0	1.0

DataFrame

(4) DataFrame, setting

#행열 위치로 값설정

```
df.iat[2, 1] = 1
df.iloc[3:, :1] = -1
display(df)
```

#열레이블로 행전체 값설정

```
df.loc[:, 'Z'] = np.arange(5)
display(df)
```

#조건으로 값설정, df 전체에서 음수값을 0으로 설정

```
df[df < 0] = 0
display(df)
```

#조건으로 값설정, Y열의 값이 10미만인 것을 9로 설정

```
df.loc[df.Y < 10, 'Y'] = 9
display(df)
```

	X	Y	Z
a	10.0	30.0	2.0
b	2.0	2.0	2.0
c	100.0	1.0	0.0
d	-1.0	-100.0	1.0
e	-1.0	100.0	1.0

	X	Y	Z
a	10.0	30.0	0
b	2.0	2.0	1
c	100.0	1.0	2
d	-1.0	-100.0	3
e	-1.0	100.0	4

	X	Y	Z
a	10.0	30.0	0
b	2.0	2.0	1
c	100.0	1.0	2
d	0.0	0.0	3
e	0.0	100.0	4

	X	Y	Z
a	10.0	30.0	0
b	2.0	9.0	1
c	100.0	9.0	2
d	0.0	9.0	3
e	0.0	100.0	4

DataFrame

(5)DataFrame, addition, deletion

```
d = {'one': pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
      'two': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
df1 = pd.DataFrame(d)
display(df1)

#DataFrame , Row addition
df1.loc['e'] = [10,10] #새로운 index에 값 설정으로 추가
df1.loc['h'] = [5,6]   #새로운 index에 값 설정으로 추가
df1.loc['sum'] = df1.sum(axis=0) #합을 구한 결과로 새로운 행 추가
display(df1)
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0
e	10.0	10.0
h	5.0	6.0
sum	21.0	26.0

DataFrame

(5)DataFrame, addition, deletion

deletion

- drop(): axis index (0 or 'index', default) or columns (1 or 'columns').
 - df = df.drop(['column'], axis=1)
 - df = df.drop(['row'])

```
#DataFrame, Row deletion
#drop : 삭제 결과를 반환하여 처리, 원본 유지
df1.drop(['e']) #원본 유지되므로 삭제처리 안됨
display(df1)
df1 = df1.drop(['sum']) #반환으로 원본 수정
display(df1)
df1.drop(['e'], inplace=True) #원본에서 삭제
display(df1)
df2= df1.drop(['a', 'h']) #df1에서 'a', 'h' 행 삭제하여 df2에 저장
display(df2)
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0
e	10.0	10.0
h	5.0	6.0
sum	21.0	26.0

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0
h	5.0	6.0

	one	two
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0
e	10.0	10.0
h	5.0	6.0

DataFrame

(5) DataFrame, addition, deletion

```
#DataFrame , Column addition
df1['three'] = df1['one'] + df1['two'] #열의 합으로 새로운 열 three 생성
df1['flag'] = df1['one'] > 3          #조건에 대한 논리값으로 새로운 열 flag 생성
df1['class'] = 'bar'                  #문자열로 class열 생성
df1['one2'] = df1['one'][:2]          #one 열의 2개행만으로 one2열 생성
df1['new'] = [1,2,3,4,5]              #new열에 [1,2,3,4,5]로 추가
display(df1)
```

	one	two	three	flag	class	one2	new
a	1.0	1.0	2.0	False	bar	1.0	1
b	2.0	2.0	4.0	False	bar	2.0	2
c	3.0	3.0	6.0	False	bar	NaN	3
d	NaN	4.0	NaN	False	bar	NaN	4
h	5.0	6.0	11.0	True	bar	NaN	5

DataFrame

(5) DataFrame, addition, deletion

#DataFrame, Column deletion

```
del df1['two']          #two 열 삭제
display(df1)

df3=df1.drop(['one2'], axis=1)  # one2 column (axis=1) 삭제
display(df3)

tree = df1.pop('three')      #three 열 삭제, 값을 반환
print("three=", tree)
display(df1)
```

	one	three	flag	class	one2
a	1.0	2.0	False	bar	1.0
b	2.0	4.0	False	bar	2.0
c	3.0	6.0	False	bar	NaN
d	NaN	NaN	False	bar	NaN
h	5.0	11.0	True	bar	NaN

```
three= a      2.0
b      4.0
c      6.0
d      NaN
h     11.0
Name: three, dtype: float64
```

	one	three	flag	class
a	1.0	2.0	False	bar
b	2.0	4.0	False	bar
c	3.0	6.0	False	bar
d	NaN	NaN	False	bar
h	5.0	11.0	True	bar

	one	flag	class	one2
a	1.0	False	bar	1.0
b	2.0	False	bar	2.0
c	3.0	False	bar	NaN
d	NaN	False	bar	NaN
h	5.0	True	bar	NaN

DataFrame

(6) DataFrame, missing value

```
#missing value 처리
df2 = df1.fillna(value=1)    #누락된 값을 채움
df3 = df1.dropna(how='any')  #누락된 값 있는 행 삭제
df4 = pd.isna(df1)          #마스크 생성
display(df1)
display(df2)
display(df3)
display(df4)
```

	one	flag	class	one2
a	1.0	False	bar	1.0
b	2.0	False	bar	2.0
c	3.0	False	bar	NaN
d	NaN	False	bar	NaN
h	5.0	True	bar	NaN

	one	flag	class	one2
a	1.0	False	bar	1.0
b	2.0	False	bar	2.0
c	3.0	False	bar	1.0
d	1.0	False	bar	1.0
h	5.0	True	bar	1.0

	one	flag	class	one2
a	1.0	False	bar	1.0
b	2.0	False	bar	2.0

	one	flag	class	one2
a	False	False	False	False
b	False	False	False	False
c	False	False	False	True
d	True	False	False	True
h	False	False	False	True

DataFrame

예제 2 데이터프레임 삽입, 삭제 연습

```
#5명의 성적 데이터 생성 (성명, 입학년도, 학점(0~4.5), 성별(M,F))
data = {'Name' : ['kim', 'lee', 'min', 'jin', 'park'],
        'Adyear' : [2017, 2017, 2018, 2019, 2018],
        'Grade' : [3.5, 2.0, 4.5, 2.2, 4.0],
        'Sex' : ['M', 'M', 'F', 'F', 'M']}
df = pd.DataFrame(data)
```

	Name	Adyear	Grade	Sex
0	kim	2017	3.5	M
1	lee	2017	2.0	M
2	min	2018	4.5	F
3	jin	2019	2.2	F
4	park	2018	4.0	M

```
#1)성별 "F"는 "여"로, "M"은 "남"으로 변경
df.loc[df.Sex == 'F', 'Sex'] = "여"
df.loc[df.Sex == 'M', 'Sex'] = "남"
display(df)
```

	Name	Adyear	Grade	Sex
0	kim	2017	3.5	남
1	lee	2017	2.0	남
2	min	2018	4.5	여
3	jin	2019	2.2	여
4	park	2018	4.0	남

```
#2)새로운 학생 1명을 추가
df.loc[5] = ['shin', 2019, 4.4, '여']
display(df)
```

	Name	Adyear	Grade	Sex
0	kim	2017	3.5	남
1	lee	2017	2.0	남
2	min	2018	4.5	여
3	jin	2019	2.2	여
4	park	2018	4.0	남
5	shin	2019	4.4	여

DataFrame

예제 2 데이터프레임 삽입, 삭제 연습

```
#3) lee 학생을 인덱스로 접근하여 삭제 (원본 유지)
df1 = df.drop([1])
display("행삭제 데이터 =", df1)

#4) age 열 추가
df['age']=[20,21,22,20,23, 22]
display("열추가 데이터 =", df)
```

'행삭제 데이터 ='

	Name	Adyear	Grade	Sex
0	kim	2017	3.5	남
2	min	2018	4.5	여
3	jin	2019	2.2	여
4	park	2018	4.0	남
5	shin	2019	4.4	여

'열추가 데이터 ='

	Name	Adyear	Grade	Sex	age
0	kim	2017	3.5	남	20
1	lee	2017	2.0	남	21
2	min	2018	4.5	여	22
3	jin	2019	2.2	여	20
4	park	2018	4.0	남	23
5	shin	2019	4.4	여	22