4차시 수업

겨울방학 파이썬 스터디

Pandas 기초

학습 목표

WINTER VACATION PYTHON STUDY

1. DataFrame, Series 이해하기
2. Indexing, Slicing 방법 이해하기
3. Merge, Concat 이해하기
4. 데이터 불러오기, 내보내기

전처리(processing)

WINTER VACATION PYTHON STUDY

전처리(processing) 란?

- 데이터 분석을 하는데 있어서 가장 기초가 되는 작업으로 주어진 데이터를 원하는 대로 수정 및 가공하는 작업을 말한다.
- 오늘 수업의 내용은 데이터 전처리의 가장 기초가 되는 부분.

Pandas란?

- Pandas는 python에서 사용하는 데이터분석 라이브러리로,
행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있게 되며
보다 안정적으로 대용량의 데이터들을 처리하는데 매우 편리한 도구

```
import pandas as pd
```

```
from pandas import Series, DataFrame
```

DataFrame , Series

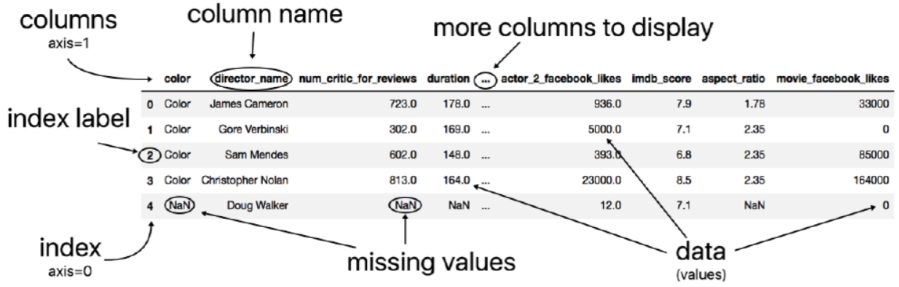
WINTER VACATION PYTHON STUDY

Series

- 하나의 열로 이루어진 데이터 (Series + Series + … = DataFrame)

DataFrame

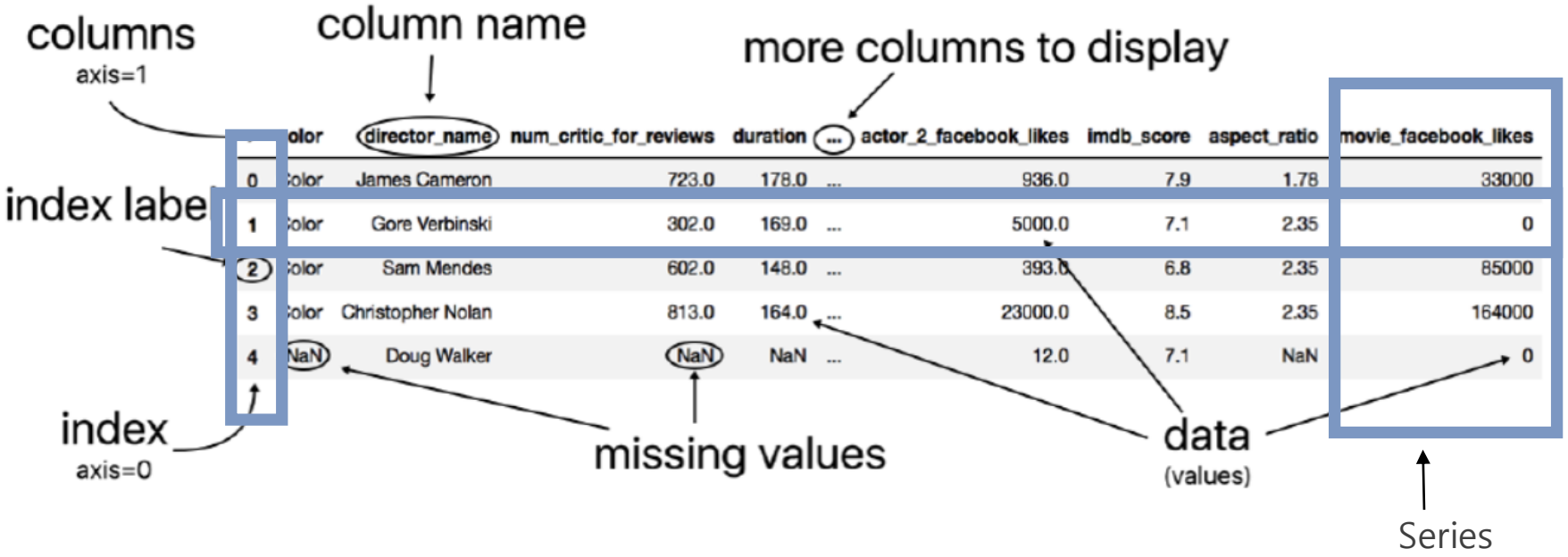
- DataFrame은 테이블 형식의 데이터 (tabular, rectangular grid 등으로 불림)를 다룰 때 사용.
- 3요소 : 컬럼(column, 열), 로우(row, 행), 인덱스(index)



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

DataFrame , Series

WINTER VACATION PYTHON STUDY



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

- Series 정의하기

```
obj = pd.Series([4,7,-5,3]) ; obj
```

```
0    4
1    7
2   -5
3    3
dtype: int64
```

function	뜻
obj.values	값만 따로 확인
obj.index	Index 범위값 확인
obj.dtypes	데이터 타입 확인
obj.name	시리즈 이름

DataFrame

WINTER VACATION PYTHON STUDY

- DataFrame 정의하기

```
data = {
    'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Ohio'],
    'year': [2000, 2001, 2002, 2001, 2002, 2000],
    'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]
}
df = pd.DataFrame(data)  ; df
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Ohio	2000	3.2

DataFrame

WINTER VACATION PYTHON STUDY

function	뜻
df.info()	대략적인 정보
df.columns	열의 이름
df.values()	값
df.size	행의 개수 x 열의 개수
df.shape	행, 열의 개수
len(df)	행 개수
df.describe()	통계 정보

function	뜻
df.head()	상위 5개 항목
df.tail()	하위 5개 항목
df.unique()	고유값 리스트
df.nunique()	고유값 개수
df.value_counts()	고유값과 빈도수
df.sort_values()	정렬
df.iloc() , df.loc()	Indexing, slicing
df.query()	질의

Column handling

WINTER VACATION PYTHON STUDY

새로운 열 추가하기

- data[new_column_name] = [values]
- Series 이용

열 삭제하기

- **del** data[column_name]
- data.**drop**(column_name, **axis = 1**, **inplace = True**)

열 이름 변경하기

- data.**rename**(columns = {before_name : after_name})
- data.columns = [column1, column2 ...]

```
df['num'] = [0,1,0,2,3,5] ; df
```

	state	year	pop	num
0	Ohio	2000	1.5	0
1	Ohio	2001	1.7	1
2	Ohio	2002	3.6	0
3	Nevada	2001	2.4	2
4	Nevada	2002	2.9	3
5	Ohio	2000	3.2	5

```
# inplace = True 인자를 적지 않으면 원본 데이터를 보존.
df.drop( 'pop', axis = 1) ; df
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Ohio	2000	3.2

Indexing, Slicing

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Ohio	2000	3.2

열 indexing

Series로 반환

- df.column name
- df[column name]

DataFrame으로 반환

- df[[column1, column2 ...]]
- df.filter([column1, column2 ...])
- df.filter(like = 's')

```
df['state'] # 시리즈
```

```
0      Ohio
1      Ohio
2      Ohio
3    Nevada
4    Nevada
5      Ohio
Name: state, dtype: object
```

```
## 여러개의 열을 동시에 indexing [[]] 대괄호를 두개!!
df[['state', 'pop']]  # 데이터 프레임
```

	state	pop
0	Ohio	1.5
1	Ohio	1.7
2	Ohio	3.6
3	Nevada	2.4
4	Nevada	2.9
5	Ohio	3.2

Indexing, Slicing

WINTER VACATION PYTHON STUDY

행 indexing

- .iloc[,] : 행 번호 (**index**) 기준으로 행 데이터 읽기
- .loc[,] : 인덱스 **이름**을 기준으로 행 데이터 읽기

	name	age	year	points
0	지은	20	2016	1.5
1	병하	20	2017	1.7
2	지환	20	2018	2.0
3	종은	40	2019	3.8
4	민지	22	2019	1.9
5	채원	25	2017	2.3

.iloc[ ]

```
## iloc - 행 번호 (0 부터) 기준
df2.iloc[1:3]  # 인덱싱번호가 1부터 2까지인 행 추출
```

	name	age	year	points
1	병하	20	2017	1.7
2	지환	20	2018	2.0

.loc[ ]

```
df2.loc[1:3]  # 인덱싱번호가 1부터 3까지인 행 추출
```

	name	age	year	points
1	병하	20	2017	1.7
2	지환	20	2018	2.0
3	종은	40	2019	3.8

행 indexing

- Boolean indexing : 값이 참 (True) 과 거짓 (False)으로 나타나는 데이터 형태

```
df2['year'] > 2017
```

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

True인 값만 추출

```
df2.loc[df2['year'] > 2017, :]  # year가 2017 초과인 행
```

	name	age	year	points
2	지환	20	2018	2.0
3	종은	40	2019	3.8
4	민지	22	2019	1.9

행 indexing

- Query : 질의기능을 수행.

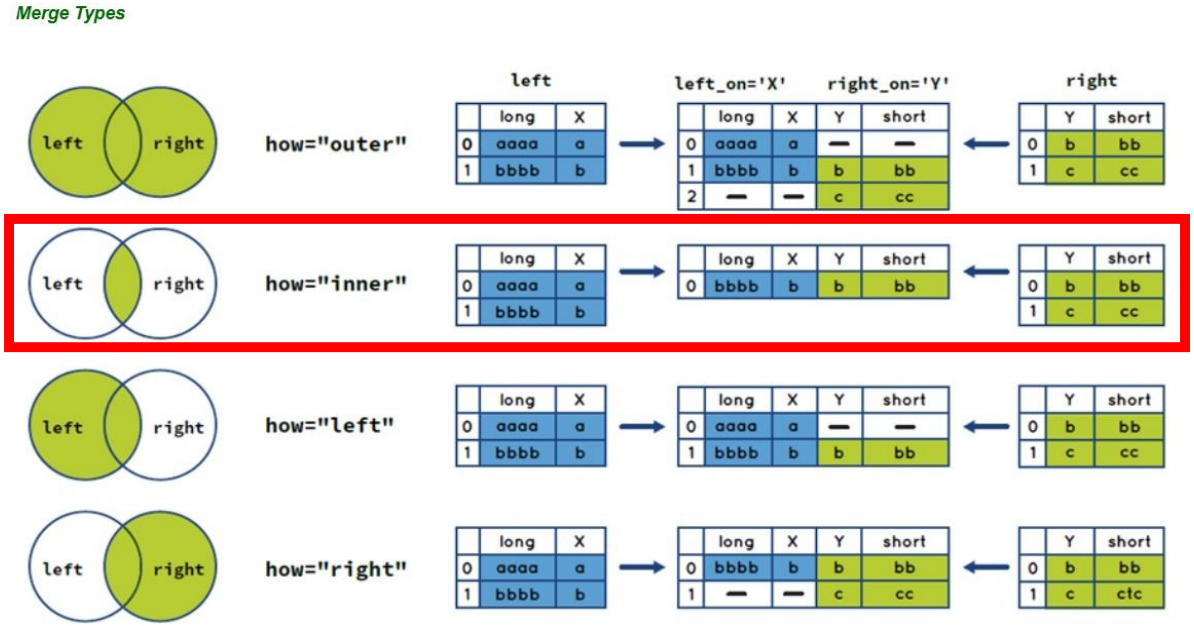
```
df2.query('year > 2017')
```

	name	age	year	points
2	지환	20	2018	2.0
3	종은	40	2019	3.8
4	민지	22	2019	1.9

Merge

WINTER VACATION PYTHON STUDY

- 데이터 병합하기



Merge

WINTER VACATION PYTHON STUDY

- **Inner** : 양쪽 DataFrame 모두에 있는 행을 추출

```
display(df1,df2)
```

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

	key	data2
0	a	0
1	b	1
2	d	2

```
pd.merge(df1, df2, on='key')
```

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

3x1 조합

3x1 조합

Merge

WINTER VACATION PYTHON STUDY

- **Outer** : 한쪽 DataFrame에만 있는 행들도 추가

```
display(df1,df2)
```

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

	key	data2
0	a	0
1	b	1
2	d	2

```
pd.merge(df1, df2, on='key', how='outer')
```

	key	data1	data2	
0	b	0.0	1.0	
1	b	1.0	1.0	3x1 조합
2	b	6.0	1.0	
3	a	2.0	0.0	
4	a	4.0	0.0	3x1 조합
5	a	5.0	0.0	
6	c	3.0	NaN	
7	d	NaN	2.0	

- **right** : 오른쪽의 df2가 기준이 되어 고정.

  왼쪽의 df1는 동일한 key값을 가질 때마다 달라 붙음, 없는 경우 NaN

```
display(df1,df2)
```

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

	key	data2
0	a	0
1	b	1
2	d	2

```
# 오른쪽의 df2가 기준이 되어 고정,
# 왼쪽의 df1는 동일한 key값을 가질 때마다 달라 붙음, 없는 경우 NaN
pd.merge(df1, df2, on='key', how='right')
```

	key	data1	data2	
0	b	0.0	1	
1	b	1.0	1	3x1 조합
2	b	6.0	1	
3	a	2.0	0	
4	a	4.0	0	3x1 조합
5	a	5.0	0	
6	d	NaN	2	

Merge

WINTER VACATION PYTHON STUDY

- **Column명이 다른 경우** : 각 DataFrame에서 기준 열을 인자로 지정

	lkey	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

	rkey	data2
0	a	0
1	b	1
2	d	2

```
# lkey 와 rkey가 기준열
pd.merge(df3, df4, left_on='lkey', right_on='rkey')
```

	lkey	data1	rkey	data2
0	b	0	b	1
1	b	1	b	1
2	b	6	b	1
3	a	2	a	0
4	a	4	a	0
5	a	5	a	0

Concat

WINTER VACATION PYTHON STUDY

- **Concat** : 데이터 단순 연결

	one	two
a	0	1
b	2	3
c	4	5

axis = 0

	three	four
a	5	6
c	7	8

```
pd.concat([df5,df6]) # 행으로 추가
```

	one	two	three	four
a	0.0	1.0	NaN	NaN
b	2.0	3.0	NaN	NaN
c	4.0	5.0	NaN	NaN
a	NaN	NaN	5.0	6.0
c	NaN	NaN	7.0	8.0

axis = 1

```
pd.concat([df5,df6], axis=1) # 열로 추가 : axis = 1 필요
```

	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

Handling Missing Data

WINTER VACATION PYTHON STUDY

결측값 확인하기

- df.isnull() : 결측값이 True로 반환
- df.notnull() : 결측값이 False로 반환

결측값 채우기

- df.fillna(값)

결측값 삭제

- df.dropna()

	lkey	data1	rkey	data2
0	b	0.0	b	1.0
1	b	1.0	b	1.0
2	b	6.0	b	1.0
3	a	2.0	a	0.0
4	a	4.0	a	0.0
5	a	5.0	a	0.0
6	c	3.0	NaN	NaN
7	NaN	NaN	d	2.0

Duplicate Rows

WINTER VACATION PYTHON STUDY

중복되는 행 확인

- df.duplicated() : 중복이면 True 반환

중복되는 행 제거

- df.drop_duplicates()

	key1	key2	num
0	a	v	1
1	b	w	2
2	b	w	2
3	c	x	4
4	c	y	5

Reading and Writing Data

WINTER VACATION PYTHON STUDY

데이터 불러오기

(1) pd.read_csv : csv 파일을 불러올 때 구분자를 써주지 않아도 된다

```
titanic = pd.read_csv('titanic.csv')
```

```
titanic = pd.read_csv('titanic.csv', encoding='cp949')
```

한글이 깨지는 경우가 종종 발생
-> encoding으로 해결

(2) pd.read_table

```
titanic2 = pd.read_table('titanic.csv', sep=',')
```

구분자 추가
Ex) '₩t' (Tab으로 구분), '|' , ',' 등

```
titanic2 = pd.read_table('titanic.csv', sep=',', encoding='cp949')
```

(3) pd.read_excel

```
excel = pd.read_excel('example2.xlsx')
```

Reading and Writing Data

WINTER VACATION PYTHON STUDY

데이터 내보내기

```
df
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9

```
df.to_csv('df.csv')
```

Index를 같이 내보냄

	A	B	C	D
1		tate	year	pop
2	0	Ohio	2000	1.5
3	1	Ohio	2001	1.7
4	2	Ohio	2002	3.6
5	3	Nevada	2001	2.4
6	4	Nevada	2002	2.9

```
df.to_csv('df.csv', sep=',', index=False
```

	A	B	C
1	state	year	pop
2	Ohio	2000	1.5
3	Ohio	2001	1.7
4	Ohio	2002	3.6
5	Nevada	2001	2.4
6	Nevada	2002	2.9

Index를 같이 내보내지 않음

감사합니다.