# Pandas Basic

1. **Series**

dictionary -> series
list -> series / 인덱스 별도정의
tuple -> series / 정수형 위치 인덱스

- **값나열, index 설정**

pd.Series(value, [index])

- **값 나열**

df.arange(1:10)

- **값 제거**

df.drop() - 복사본에 연산을 진행, 원본은 그대로
df.drop(inplace=true) - df 원본에 바로 연산

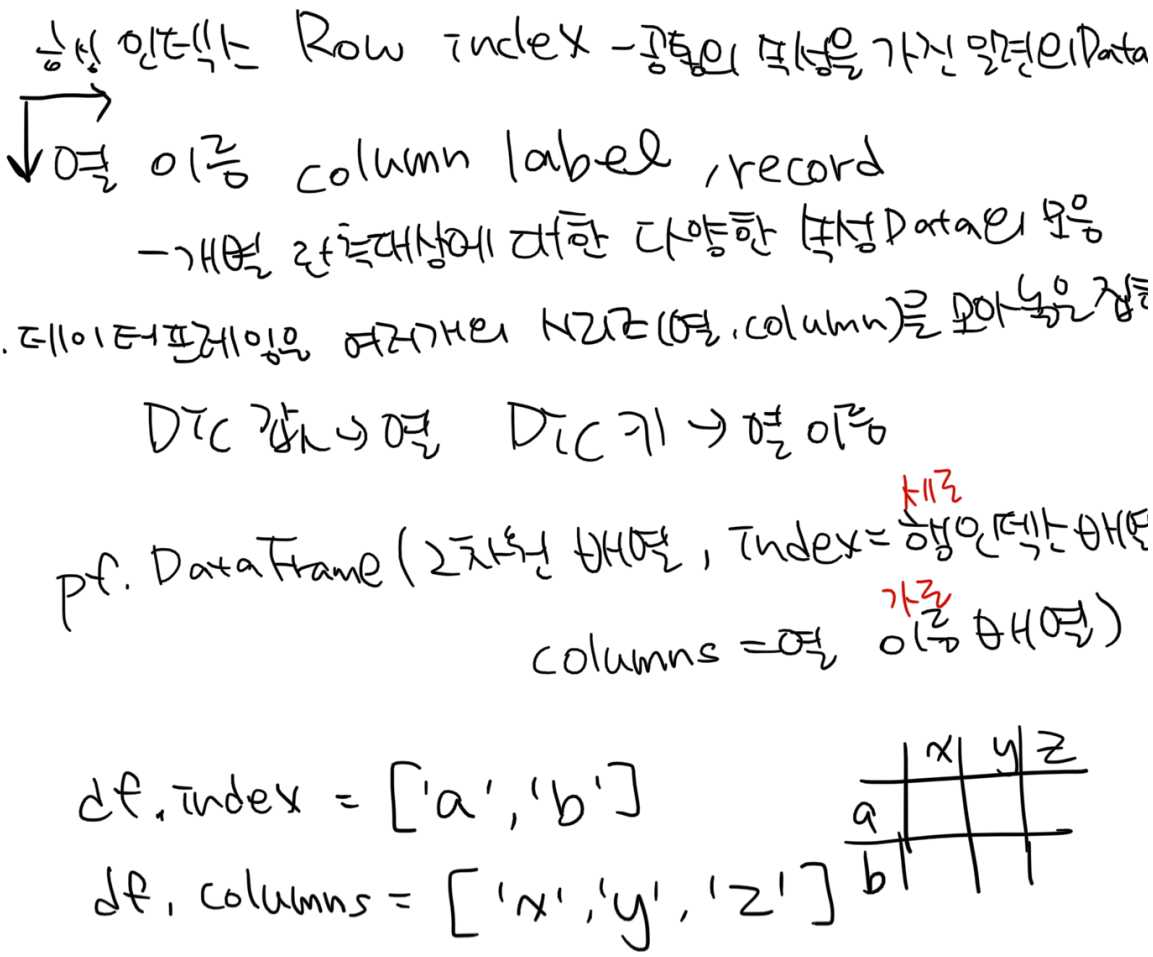
*문자열로 이루어진 인덱스는 문자열로 호출할때 마지막 인덱스도 출력
df.shape

- **숫자로 이루어진 값의 summary**

df.describe()

- 데이터 형태 변경

df.astype(str)




- **이름 변경**

df.rename(columns={a:'연령', b:'남녀'}

- 인덱스 활용

set_index() -

- 원소 선택 : [행,열] 2차원 좌표를 입력하여 원소 위치를 지정

df.loc["수영", 수학"]
df.iloc[row#, column#]
*2개 이상의 행과 2개 이상의 열을 선택하면 데이터프레임 객체를 반환

- 열/행 추가

df.["과학"] = 80 -

df.loc["은서"] = 80 - 열은 loc인덱서를 사용
df.loc[4] = ["동규", 90,70,61,57]
df.loc[5] = df.loc[4] - 기존 행 복사

- 원소 값 변경 : 인덱싱과 슬라이싱 기법 사용
*새로운 인덱스를 지정하고 객체에 변경사항 반영 df.set_index("이름", inplace=True)
df.iloc[0][3] = 80
df.loc["서준"]["체육"]
df.loc["서준", "체육"]
df.loc["서준", ["음악", "체육"]]

- 행열 위치 바꾸기
df.transpose() - 메소드 활용
df = df.T - 클래스속성 활용??

df = pd.read_csv('PATH', sep=' , ' , header=None, index_col='', usecols =[' ', ' '])

- index_col = '해당column' 으로 지정 가능

df[7:10] column행값을 선택
df.loc - 존재하는 인덱스에서 원하는 row선택하기
df.iloc - 어떤 index상관없이 인덱스 순서대로 가져옴, 문자대신 인덱스 숫자만

-

- 중복 요소 수 세기
value_counts() - 동일한 요소 갯수 확인
df.value_counts()

ex) values = df['Invoice'].value_counts().index.sort_value()
동일 인보이스별로 묶고 그걸 인덱스로 만들고 그 인덱스로 나열

- **데이터 합치기**
*concat은 그냥합치기. merge/join은 인덱스 결합

df = pd.concat([df1, df2], axis=0, join='outer') - axis: 0 행합치기 1 열합치기/ join=
outer 합치기 inner교집합

df = pd.

merge() - SQL의 join처럼 특정한 column을 기준으로 병합
PM: inner- 기본값, 일치하는 값이 있는 경우
left-left outer join / right-right outer join / outer: full outer join

- **Boolean Selection - row가로 선택**

30대 & 1등석

class_ = df[‘Class’] ==1 - true or false
age_ = (df[‘Age’] >= 30 ) & (df[‘Age’] < 40) - true or false

df[class_ & age_]

- **Column세로 추가와 삭제**

df[‘Age_db’] = df[‘Age’] * 2 - 나이두배값 변수컬럼 추가, 마지막에

df.insert(#행위치, ‘Fare10’,  df[‘Fare’]/10) - 원하는 위치에 변수컬럼 추가

df.drop(‘Age_tripple’, axis=1) - 원하는 변수 전체 삭제

*연산은 원본데이터를 건들지 않음, 변하고 싶으면 inplace=True / 기본값은 False

- **변수사이의 상관계수(correlation)**

df.corr() - 대각선값이 1인 변수들간의 상관계수 구함, 절대값이 크면 영향을 크게 미침
plt.matahow(df.corr()) - 상관계수를 plot으로 표현, 색이 깊을수록 관계가 깊음

- **NA 결측값**

df.isna() - na값 있는지 확인
df.[‘age’].isna() - 특정 컬럼에 대해 na값 확인

df.isnull() - 누락데이터면 True/유효한 데이터면 False
df.notnull() - 유효한 데이터면 True/누락 데이터면 False

df.dropna() - 어떤 데이터에 NA가 있으면 전체 row가로 삭제
df.dropna(subset=[‘list’]) - 해당변수에 na가 있으면 row 전체 삭제

df.dropna(axix=1) - 컬럼기준으로 nar가 있으면 column세로 전체가 삭제

df[‘Age’].fillna(df.[‘Age’].mean()) - na를 평균값으로 대체

*inplace=true - 데이터 자체가 바뀜

df[df[‘Survived’] == 1][‘Age’].mean() - 생존자의 평균 나이
df[df[’Survived’] == 0][‘Age’].mean() - 사망자의 평균 나이

df[df[‘Survived’] == 1][‘Age’].fillna(mean1) - 생존자 평균 나이로 결측치 채우기
df[df[‘Survived’] == 0][‘Age’].fillna(mean2) - 사망자 평균 나이로 결측치 채우기

- **중복 데이터 처리**

df.duplicated() - 중복데이터 확인, 중복이면 True/아니면 False

- 행선택

df.loc[df[‘Survived’] == 1, ‘Age’] = df[df[Survived] == 1]
[‘Age’].fillna(mean1) - 생존자 평균 나이로 결측치 채우고 원본 데이터 바꾸기

- **데이터 형태에 따른 조치 apply**
  df.info() - 각 데이터 변수의 타입 확인 (int64정수, float64실수, object)

*범주형 데이터는 연산이 불가하기 때문에 버리거나 숫자형으로 변경해야함

숫자형 -> 범주형 age변수 변환하기, 변환로직을 함수로 만든 후 apply함수로 적용

```
import math

def age_categorize(age):
    if math.isnan(age):  #나이가 없으면 모두 음수1로 변환


        return -1
    return math.floor(age / 10) *10
```

df[‘Age’].apply(age_gategorize) - 범주형 변수로 변환

- **범주형 변수를 전처리 one-hot encoding**

*범주형 변수는 연산이 불가하므로 숫자형으로 변경해야함

pd.get.dummies(df) - 모든 object를 컬럼레벨로 올림, 전체 범주가 컬럼으로

올라가고 해당하면 1 아니면 0으로 변경
pm: columns=['Pclass', 'Sex', 'embarked'], drop_first=Ture) drop_first: 구분이 되는 변수는 줄임

- **group by**

df_group = df.groupby('Pclass') -
df_group.groups - 그룹스 속성 확인, dic
*생성된 그룹을 가지고 연산
df_group.count() - class에 따른 단순한 카운팅
df_group.sum()
df_group.mean/std/var - 평균, 표준편차, 분산
df_group.min/max()

성별에 따른 생존률 구하기
df.groupby('Sex').mean()['Survived']

복수개의 변수로 그루핑
df.groupby(['Pclass', 'Sex']).mean()

df.groupby(['Pclass', 'Sex']).mean().loc(2, 'female') -두번째 행의 female 항목만 가져옴

*데이터프레임에서 해당 변수를 인덱스로 바꿈, 멀티인덱스도 가능
df.set_index('Pclass').reset_index()

df.set_index('Age').groupby(level=0).mean()
인덱스가 있으면 레벨이 0, 0번 변수로 그루핑을 해라

나이대별로 생존률구하기
```
def age_categorize(age):
    if math.isnan(age):
        return -1
    return math.floor(age/10)*10
```

df.set_index('Age').groupby(age_categorize).mena()['Survived']

multi-index를 이용한 grouping

df.set_index(['Pclass', 'Sex']).groupby(level=[0,1]).mean()['Age']

집계함수 - 그룹별로 데이터를 한번에 볼수 있음, aggregate 호출해서 리스트에 함수 적용
df.set_index(['Pclass', 'Sex']).groupby(level=[0,1]).aggregate([np.mean,

np.sum, np.max])

- **transform함수**

원본데이터프레임에 인덱스를 유지한

- 데이터 쌓기. stack-unstack

df.unstack(인덱스#)
df.unstack(인덱스#).stack(인덱스#)

- 데이터프레임 정렬

df = df.sort_values()  - 해당변수에 따라 정렬
PM: by = [‘column’] axix=0세로컬럼이름/1가로row번호 ascending = False 내림차순

user_propensity_vector = []

for movie_list = tqdm_notebook(u