12. 데이터프레임과 시리즈(Pandas) #src: 8_Python/ch12_데이터프레임과시리즈(Pandas) 참고

1) Pandas 패키지; 1차원 구조의 시리즈(Series)와 2차원 구조의 데이터프레임(DataFrame) 제공

- 부분집합 조회, 열추가 및 제거, 병합, 데이터 구조 변경 등 데이터 전처리를 위한 기능 제공

#https://pypi.org/project/pandas/ #https://pandas.pydata.org/pandas-docs/stable/reference/index.html

2) 데이터프레임 만들기 #import pandas as pd

(1) 딕셔너리를 이용해서 데이터프레임 생성 pd.DataFrame(data='딕셔너리') #key가 열 이름

(2) 리스트를 이용해서 데이터프레임 생성 pd.DataFrame({'col1':리스트1, 'col2':리스트2})

pd.DataFrame(numpy.c_[리스트1, 리스트2], columns=['열이름1','열이름2'])

(3) read_csv('파일명', sep=',', encoding='인코딩', comment='#') #sep=구분기호, comment=주석행

3) 열, 행 이름 지정

(1) 열이름 지정: df.columns = ['열이름1', ...] (2) 행이름 지정: df.index = ['행이름1', ...]

(3) 레벨 이름 지정: df.columns = [['열분류1', '열분류2'], ['열이름1', '열이름2', ...]] #index도 동일

4) 부분 데이터 조회: 단일열, loc[columns, index], iloc, 조건

(1) iloc[from, to, by]; from부터 by씩 증가해서 to전(to 미포함)까지 #처음:0, 마지막:-1

5) 데이터 추가 및 삭제

(1) 데이터프레임 요소 삭제

① 단일행 삭제: df.drop(n, axis=0); n행 삭제된 데이터셋을 반환(데이터프레임에 반영X)

② 단일열 삭제: df.drop('열이름', axis=1); 해당 열을 삭제한 데이터셋을 반환(실제 삭제X)

③ 여러 행이나 열을 삭제: df.drop(labels=['행이름1','행이름2',...]) #열 axis=1 기재 必

(2) 데이터프레임 요소 추가

① 열 추가: df['추가할 열이름'] = 추가할 데이터 ┌#ignore_index=True 필요X, index조정 필요

② 행 추가: df.append(시리즈or딕셔너리or데이터프레임, ignore_index=True)

6) 정렬 #결과 반영하려면 inplace=True 기재 필요

(1) df.sort_index(); 행이름(index)으로 정렬 df.sort_index(axis=1); 열이름(columns)으로 정렬

(2) sort_values(by=["정렬기준"], inplace=True); 값으로 정렬 #inplace=True; 데이터프레임에 반영

7) 기초 통계분석

(1) describe; 요약 통계량(기본값은 숫자 열의 분석만 반환) #include='all' 모든 열 분석 반환

① 숫자 데이터: count, mean, std, min, median, max, quantile; 분위수 포함 #var; 분산(std*std)

② 객체 데이터(문자): count, unique, top, freq 포함 #top; 가장 일반적인 값, freq; top의 빈도

(2) count; 결측치(NaN)을 제외한 개수, cumprod; 누적합, corr; 상관관계(상관계수), cov; 공분산

8) 데이터 그룹화 및 집계

(1) groupby df.groupby(df['그룹화할 열']) #groupby[df['열이름1'], df.열이름2]; 다중열 그룹화可

① df.pivot_table(index="그룹화할 열", values=['열1', '열2'], aggfunc='mean')

② transpose(), T; 전치행렬, unstack() 등 이용해서 행과 열 위치 변경 → 가독성 높이기

(2) 그룹간 데이터 반복 처리 #df.groupby(df.열이름).take([추출할 행 번호])

```
for type, group in data_grouped:
    print(type, '₩n', group.head())
```

(3) 레이블(원핫 인코딩): LabelEncoder()이용 #from sklearn.preprocessing import LabelEncoder 필요

9) 데이터 구조 변경

(1) 와이드 포맷과 롱 포맷

와이드 포맷(wide format)	롱 포맷(long format)
가로로 긴 형식, 열 단위 데이터 구조 피벗테이블(pivot table)	세로로 긴 형식, 행 단위 데이터 구조 언피벗테이블(unpivot table)

(2) melt()를 이용한 언피벗팅; df.melt(id_vars=['열로 남겨둘 열', ...])

#melt: https://pandas.pydata.org/docs/user_guide/reshaping.html#reshaping-by-melt

#Tidy Data: https://vita.had.co.nz/papers/tidy-data.pdf (7페이지)

(3) pivot_table()을 이용한 피벗팅(롱 포맷→와이드 포맷)

```
df2 = df1.pivot_table(index=['열1',...], columns=['열2'], values=['멜팅된 열'], aggfunc='mean')
df3 = df2.reset_index(level=['열1',...], col_levle=1)
df3.columns = df3.columns.droplevel(level=0)
```

10) 데이터프레임에 함수 적용

(1) apply(); 데이터프레임이나 시리즈의 각 열(axis=0) 또는 각 행(axis=1)에 함수 적용 가능

(2) applymap(); 데이터프레임의 각 요소 하나하나 별로 함수 적용

(3) map(); 시리즈 타입의 벡터만 가능

11) 일괄 변경(결측치나 특정값) #inplace=True; 변경된 내용 데이터프레임에 적용(반환값 無)

(1) fillna(특정값); 결측치를 특정값으로 변경

① df.fillna(method=ffill or pad); 결측치가 아닌 이전값으로 채움

② df.fillna(method=bfill or backfill); 결측치가 아닌 다음값으로 채움

(2) replace(to_value, new_value, inplace=False); to_value를 new_value로 변경

(3) where; 조건이 만족하는 요소는 그대로 출력 ↔ mask ┌#axis='columns'; NaN있는 열 제거

(4) dropna; 결측치 있는 행 제거 #thresh=2; NaN 2이상 있는 행, how='all'; 전부 NaN인 행 제거

12) 시리즈; pd.Series([값], index=['행 이름']