Chapter 14

Pandas 데이터 분석

Pandas란?

- 데이터 분석에 가장 인기있는 파이썬 패키지
- 열과 행으로 구성된 테이블 형태의 데이터를 다루는 데 주로 사용
- Series 클래스와 DataFrame 클래스로 구성

예제 14-1. Series 객체 생성

ex14.ipynb

```
import pandas as pd


obj = pd.Series([5, -4, 7, 0, 12])

print(obj)
```

: : 실행 결과

```
0 5
1 -4
2 7
3 0
4 12
dtype: int64
```

예제 14-2. Series 객체의 값과 인덱스

ex14.ipynb

```
import pandas as pd


obj = pd.Series([8, -20, -3, 13, 2])

print(obj.values)

print(obj.index)


print(obj[2])
```

: : 실행 결과

```
[ 8 -20 -3 13 2]
RangeIndex(start=0, stop=5, step=1)
-3
```

예제 14-3. Series 인덱스 설정

ex14.ipynb

```
obj = pd.Series([10, 20, 30, 40, 50], index=['a',
'b', 'c', 'd', 'e'])
print(obj)


print(obj['c'])
print(obj[['d', 'a']])
print(obj[1:4])
```

: : 실행 결과

```
a 10
b 20
c 30
d 40
e 50
dtype: int64
30
d 40
a 10
dtype: int64
b 20
c 30
d 40
dtype: int64
```

예제 14-4. Series 산술연산과 조건식

ex14.ipynb

```
import pandas as pd


obj = pd.Series([10, 20, 30, 40, 50])

print(obj * 10)


print(obj[obj > 25])
```

: : 실행 결과

```
0 100
1 200
2 300
3 400
4 500
dtype: int64
2 30
3 40
4 50
dtype: int64
```

예제 14-5. Series에서 for문 활용

ex14.ipynb

```
import pandas as pd


pop = pd.Series([9765623, 3441453,
2461769], index=['서울', '부산', '대구'])


for i, v in pop.items() :
    print('%s : %d명' % (i, v))
```

: : 실행 결과

```
서울 : 9765623명
부산 : 3441453명
대구 : 2461769명
```

예제 14-6. Series에서 딕셔너리 사용

ex14.ipynb

```
import pandas as pd
pop = pd.Series({'서울':9765623, '부산':3441453, '대구
':2461769}, index = ['서울', '부산', '대구', '광주', '대전'])
print(pop)
pop['광주'] = 149336


print('광주시 인구 : %.0f명' % pop['광주'])
```

: : 실행 결과

```
서울 9765623.0
부산 3441453.0
대구 2461769.0
광주 NaN
대전 NaN
dtype: float64
광주시 인구 : 149336명
```

예제 14-7. DataFrame 객체 생성

ex14.ipynb

```
import pandas as pd
data = {'이름':['홍지수', '안지영', '김성수', '최예린'],
        '아이디' : ['jshong', 'jyahn', 'sukim', 'yrchoi'],
        '비밀번호' : ['1234', '1234', '1234', '1234']}


frame = pd.DataFrame(data)
print(frame)
```

: : 실행 결과

```
  이름 아이디 비밀번호
0 홍지수 jshong 1234
1 안지영 jyahn 1234
2 김성수 sukim 1234
3 최예린 yrchoi 1234
```

예제 14-8. DataFrame 열과 행 인덱스 설정

ex14.ipynb

```
import pandas as pd
member = {'이름':['김영준','한지원'],  '나이':[20, 23],
        '전화번호':['010-3535-4576', '010-1295-7899']}

frame = pd.DataFrame(member, columns=['이름', '전화번
호', '나이', '주소'],  index=['01', '02'])
print(frame)
```

: : 실행 결과

```
이름 전화번호 나이 주소
01 김영준 010-3535-4576 20 NaN
02 한지원 010-1295-7899 23 NaN
```

예제 14-9. loc을 이용한 요소 추출

ex14.ipynb

```
import pandas as pd


data = {'학교명':['가나고', '다라고', '마바고',
'사아고', '자차고'],
        '학급수' : [25, 23, 15, 19, 10],
        '학생수' : [620, 600, 550, 580, 400],
        '교사수' : [80, 95, 70, 90, 65]}
```

```
frame = pd.DataFrame(data,index=['01', '02',
'03', '04', '05'])
print(frame)


print(frame.loc['02', '학생수'])
print(frame.loc['04', ['학교명', '학급수', '교
사수']])
```

예제 14-9. loc을 이용한 요소 추출(계속)

: : 실행 결과

```
학교명 학급수 학생수 교사수
01 가나고 25 620 80
02 다라고 23 600 95
03 마바고 15 550 70
04 사아고 19 580 90
05 자차고 10 400 65
600
학교명 사아고
학급수 19
교사수 90
Name: 04, dtype: object
```

예제 14-10. iloc을 이용한 요소 추출

ex14.ipynb

```
import pandas as pd


data = {'아이디':['kim', 'song', 'han', 'choi'],
        '구매상품' : ['상품A', '상품B', '상품C', '상품D'],
        '가격' : [15000, 23000, 33000, 50000],
        '개수' : [3, 5, 1, 10],
        '구매일' : ['0303', '0810', '0120', '0601']}
```

```
frame = pd.DataFrame(data)

print(frame)


print(frame.iloc[2, 0])

print(frame.iloc[3, :2])

print(frame.iloc[:, [0, 4]])
```

예제 14-10. iloc을 이용한 요소 추출(계속)

: : 실행 결과

```
   아이디 구매상품 가격 개수 구매일
0 kim   상품A 15000 3   0303
1 song 상품B 23000 5   0810
2 han   상품C 33000 1   0120
3 choi  상품D 50000 10 0601
Han
아이디    choi
구매상품  상품D
Name: 3, dtype: object
```

```
   아이디 구매일
0 kim   0303
1 song 0810
2 han   0120
3 choi  0601
```

예제 14-11. DataFrame의 sum() 메소드

ex14.ipynb

```
import pandas as pd


scores = {'이름': ['김지영', '안지수', '최성수', '황예
린', '김소정'],
        '국어' : [95, 97, 90, 94, 87],
        '영어' : [90, 86, 93, 85, 93],
        '수학' : [85, 88, 89, 88, 99]}
```

```
frame = pd.DataFrame(scores)
print(frame)


frame2 = frame.iloc[:, [1, 2, 3]]
print(frame2)


total = frame2.sum(axis = 1)
print(total)
```

예제 14-11. DataFrame의 sum() 메소드(계속)

: : 실행 결과

```
   이름 국어 영어 수학
0 김지영 95 90 85
1 안지수 97 86 88 2
최성수 90 93 89
3 황예린 94 85 88
4 김소정 87 93 99
   국어 영어 수학
0 95 90 85
1 97 86 88
2 90 93 89
3 94 85 88
4 87 93 99
```

```
0 270
1 271
2 272
3 267
4 279
dtype: int64
```

예제 14-12. 성적 합계와 평균

ex14.ipynb

```
import pandas as pd
scores = {'이름': ['김지영', '안지수', '최성수',
'황예린', '김소정'],
        '국어' : [95, 97, 90, 94, 87],
        '영어' : [90, 86, 93, 85, 93],
        '수학' : [85, 88, 89, 88, 99]}
frame = pd.DataFrame(scores)
frame2 = frame.iloc[:, [1, 2, 3]]
```

```
total = frame2.sum(axis = 1)
avg = frame2.mean(axis = 1)
print('-' * 50)
print('이름    합계  평균')
print('-' * 50)
for i in range(5) :
    print('%s  %d   %.2f' % (frame.iloc[i, 0],
total.iloc[i], avg.iloc[i]))
print('-' * 50)
```

예제 14-12. 성적 합계와 평균(계속)

: : 실행 결과

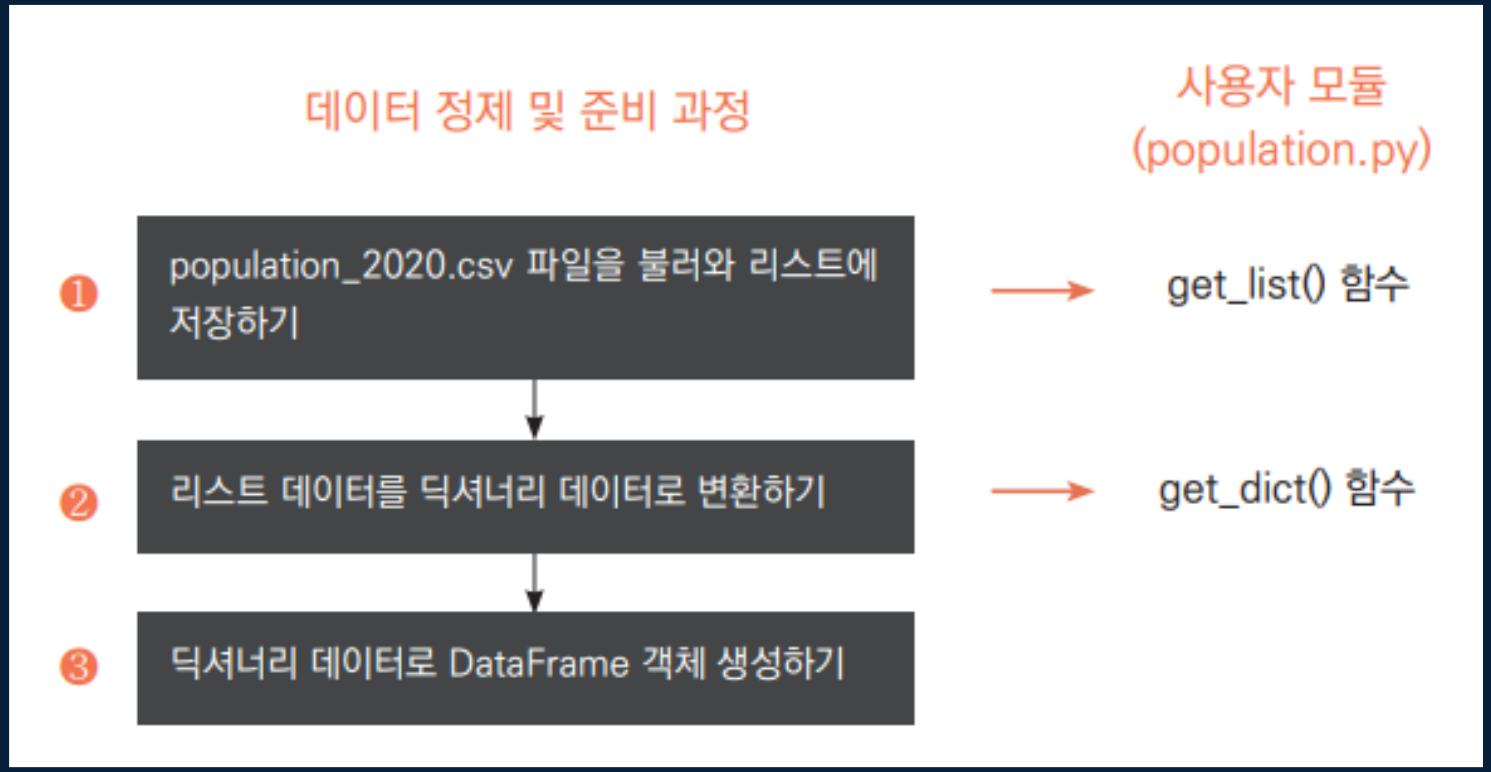
```
---------------------------------------------
이름 합계 평균
---------------------------------------------
김지영 270 90.00
안지수 271 90.33
최성수 272 90.67
황예린 267 89.00
김소정 279 93.00
---------------------------------------------
```

국내 인구 통계 데이터 분석

population_2020.csv

[['서울특별시 (1100000000)', '부산광역시 (2600000000)', '대구광역시 (2700000000)', '인천광역시 (2800000000)', '광주광역시 (2900000000)', '대전광역시 (3000000000)', '울산광역시 (3100000000)', '세종특별자치시 (3600000000)', '경기도 (4100000000)', '강원도 (4200000000)', '충청북도 (4300000000)', '충청남도 (4400000000)', '전라북도 (4500000000)', '전라남도 (4600000000)', '경상북도 (4700000000)', '경상남도 (4800000000)', '제주특별자치도 (5000000000)'], ['9,736,962', '3,410,925', '2,432,883', '2,954,955', '1,456,121', '1,473,125', '1,145,710', '343,788', '13,265,377', '1,539,521', '1,598,599', '2,120,995', '1,815,112', '1,861,894', '2,658,956', '3,358,828', '670,876'], …

데이터 정제 및 준비 과정


데이터 정제 및 준비 과정
사용자 모듈
(population.py)
❶
population_2020.csv 파일을 불러와 리스트에 저장하기
get_list() 함수
❷
리스트 데이터를 딕셔너리 데이터로 변환하기
get_dict() 함수
❸
딕셔너리 데이터로 DataFrame 객체 생성하기

예제 14-13. CSV 파일 => 리스트

ex14.ipynb

```
import population as pop


list_data = []
# CSV 파일을 읽어 들여 리스트 list_data에 저장
pop.get_list(list_data)
print(list_data)
```

예제 14-13. CSV 파일 => 리스트(계속)

: : 실행 결과

[['서울특별시 (1100000000)', '부산광역시 (2600000000)', '대구광역시 (2700000000)', '인천광역시 (2800000000)', '광주광역시 (2900000000)', '대전광역시 (3000000000)', '울산광역시 (3100000000)', '세종특별자치시 (3600000000)', '경기도 (4100000000)', '강원도 (4200000000)', '충청북도 (4300000000)', '충청남도 (4400000000)', '전라북도 (4500000000)', '전라남도 (4600000000)', '경상북도 (4700000000)', '경상남도 (4800000000)', '제주특별자치도 (5000000000)'], ['9,736,962', '3,410,925', '2,432,883', '2,954,955', '1,456,121', '1,473,125', '1,145,710', '343,788', '13,265,377', '1,539,521', '1,598,599', '2,120,995', '1,815,112', '1,861,894', '2,658,956', '3,358,828', '670,876'], ['4,345,877', ...

population 모듈의 모듈 함수 get_list()

population.py

```
def get_list(list_data) :
    import csv
    f = open('population_2020.csv', 'r',
encoding='utf-8')
    lines = csv.reader(f)


    header = next(lines)
    list_tmp = []
    for line in lines :
        list_tmp.append(line[:])
for j in range(7) :
        tmp = []
        for i in range(len(list_tmp)) :
            tmp.append(list_tmp[i][j])
        list_data.append(tmp)
```

예제 14-14. 리스트 => 딕셔너리

ex14.ipynb

```
import population as pop


list_data = []
pop.get_list(list_data)
dict_data = {}
keys = ['지역', '총인구수', '세대수', '세대당_인구', '남자_인구수', '여자_인구수', '남여_비율']
pop.get_dict(list_data, keys, dict_data)
print(list_data)
```

예제 14-14. 리스트 => 딕셔너리(계속)

: : 실행 결과

{'지역': ['서울특별시', '부산광역시', '대구광역시', '인천광역시', '광주광역시', '대전광역시', '울산광역시', '세종특별자치시', '경기도', '강원도', '충청북도', '충청남도', '전라북도', '전라남도', '경상북도', '경상남도', '제주특별자치도'], '총인구수': [9736962, 3410925, 2432883, 2954955, 1456121, 1473125, 1145710, 343788, 13265377, 1539521, 1598599, 2120995, 1815112, 1861894, 2658956, 3358828, 670876], '세대수': [4345877, 1502333, 1033349, 1242107, 618503, 637726, 469551, 136629, 5497087, 721003, 723931, 961890, 818452, 873871, 1229265, 1455655, 293932], ...

population 모듈의 모듈 함수 get_dict()

population.py

```
def get_dict(list_data, keys, dict_data) :
    area = get_area(list_data[0])
    dict_data.update({keys[0]:area})
    for i in range(1, 7) :
        if i==3 or i==6 :
            data = del_comma(list_data[i], 'float')
        else :
            data = del_comma(list_data[i], 'integer')
        dict_data.update({keys[i]:data})
```

예제 14-15. 딕셔너리로 DataFrame 객체 생성.

ex14.iphynb

```
mport population as pop
import pandas as pd


list_data = []
list_data에 저장
pop.get_list(list_data)
dict_data = {}
```

```
keys = ['지역', '총인구수', '세대수', '세대당_인
구', '남자_인구수', '여자_인구수', '남여_비율']
pop.get_dict(list_data, keys, dict_data)
frame = pd.DataFrame(dict_data)
print(frame)
```

예제 14-15. 딕셔너리로 DataFrame 객체 생성(계속)

: : 실행 결과

```
지역 총인구수 세대수 세대당_인구 남자_인구수 여자_인구수 남여_비율
0 서울특별시 9736962 4345877 2.24 4745133 4991829 0.95
1 부산광역시 3410925 1502333 2.27 1673266 1737659 0.96
2 대구광역시 2432883 1033349 2.35 1202364 1230519 0.98
3 인천광역시 2954955 1242107 2.38 1481133 1473822 1.00
4 광주광역시 1456121 618503 2.35 720686 735435 0.98
5 대전광역시 1473125 637726 2.31 735791 737334 1.00
6 울산광역시 1145710 469551 2.44 588626 557084 1.06
...
```

예제 14-16. 총 인구순 정렬

ex14.iphynb

```
frame = pd.DataFrame(dict_data)


rank = frame.sort_values(by=['총인구수'], ascending=False)
print(rank)


rank = rank.reset_index(drop=True)
print(rank)
```

예제 14-16. 총 인구순 정렬(계속)

: : 실행 결과

```
지역 총인구수 세대수 세대당_인구 남자_인구수 여자_인구수 남여_비율
8 경기도 13265377 5497087 2.41 6672706 6592671 1.01
0 서울특별시 9736962 4345877 2.24 4745133 4991829 0.95
1 부산광역시 3410925 1502333 2.27 1673266 1737659 0.96
...
지역 총인구수 세대수 세대당_인구 남자_인구수 여자_인구수 남여_비율
0 경기도 13265377 5497087 2.41 6672706 6592671 1.01
1 서울특별시 9736962 4345877 2.24 4745133 4991829 0.95
2 부산광역시 3410925 1502333 2.27 1673266 1737659 0.96
3 경상남도 3358828 1455655 2.31 1690600 1668228 1.01
...
```

예제 14-17. 국내 인구수, 세대수, 남녀 인구수

ex14.iphynb

```
frame = pd.DataFrame(dict_data)
frame2 = frame.iloc[:, [1, 2, 4, 5]]
print(frame2)
sum = frame2.sum(axis=0)
print(sum)
print('-' * 50)
print('국내 전체 인구 통계')
print('-' * 50)
```

```
print('- 총 인구수 : %d명' % sum.iloc[0])
print('- 총 세대수 : %d명' % sum.iloc[1])
print('- 총 남자 인구수 : %d명' % sum.iloc[2])
print('- 총 여자 인구수 : %d명' % sum.iloc[3])
print('-' * 50)
```

예제 14-17. 국내 인구수, 세대수, 남녀 인구수(계속)

: : 실행 결과

```
총인구수 세대수 남자_인구수 여자_인구수
0 9736962 4345877 4745133 4991829
1 3410925 1502333 1673266 1737659
2 2432883 1033349 1202364 1230519
3 2954955 1242107 1481133 1473822
4 1456121 618503 720686 735435
5 1473125 637726 735791 737334
6 1145710 469551 588626 557084
…
```

예제 14-18. 인구 통계 시각화하기

ex14.iphynb

```
frame = pd.DataFrame(dict_data)
index = ['서울', '부산', '대구', '인천', '광주', '대전']
x1 = frame.iloc[:6, 1]
x1 = x1.values.tolist()
x2 = frame.iloc[:6, 2]
x2 = x2.values.tolist()
df = pd.DataFrame({'총인구수':x1, '총세대수':x2}, index=index)
ax = df.plot.bar(rot=0)
```

예제 14-18. 인구 통계 시각화하기(계속)