Pandas

Pandas DataFrame

- **N행 확인하기**
  - .head(N) 함수
    - 데이터프레임의 상위 N개의 행을 출력
    - N생략하면 5개
  - .tail(N)
    - 하위 N개의 행을 출력

```
df.head()
```

	항목	구분	중구	서구	동구	영도구	부산진구	동래구	남구	북구
0	매립처리량	연간(톤)	3034	2323	1653	1219	17393	9195	8042	10498
1	매립처리량	일일(톤)	8	7	5	3	48	25	22	29
2	소각처리량	연간(톤)	13729	10446	8297	8651	538	17	158	97
3	소각처리량	일일(톤)	38	29	23	24	2	0	0	0
4	재활용처리량	연간(톤)	9082	21301	17968	24616	88453	56526	57921	63879

```
df.head(2)
```

	항목	구분	중구	서구	동구	영도구	부산진구	동래구	남구	북구	해운대
0	매립처리량	연간(톤)	3034	2323	1653	1219	17393	9195	8042	10498	5
1	매립처리량	일일(톤)	8	7	5	3	48	25	22	29	

Pandas DataFrame

- **정보확인하기**
  - .shape 속성 : (행, 열) 크기를 확인하기
    - 행의 개수 : len(데이터프레임), len(.index), .shape[0]
    - 열의 개수 : len(.column), .shape[1]
  - .info()
    - 데이터프레임에 대한 전반적인 정보
    - 데이터프레임을 구성하는 행과 열의 크기, 컬럼명, 컬럼을 구성하는 값의 자료형 등을 출력
  - .count()
    - Null이 아닌 행의 개수 확인
    - Null이 아닌 열의 개수 확인 : .count(axis='columns')
  - .value_counts()
    - 개별 컬럼 내에 각각의 값이 나온 횟수

Pandas DataFrame

- **요약 통계량 확인**
  - **.describe()**
    - 데이터프레임의 열별 요약 통계량
    - sum(),mean(), max(), median()등 개별 함수를 사용가능

- **시리즈 내에 유일한 값 확인**
  - **.unique()**
    - 개별 컬럼 내에 유일한 값을 확인

Pandas DataFrame 슬라이싱

- .loc[행인덱싱, 열인덱싱]

```
#행추출
df2.loc['매립처리량']
```

항목	구분	중구	서구	동구	영도구	부산진구	동래
매립처리량	연간(톤)	3034	2323	1653	1219	17393	91
매립처리량	일일(톤)	8	7	5	3	48	

```
df2.loc[['매립처리량','재활용처리량']]
```

항목	구분	중구	서구	동구	영도구	부산진구
매립처리량	연간(톤)	3034	2323	1653	1219	17393
매립처리량	일일(톤)	8	7	5	3	48
재활용처리량	연간(톤)	9082	21301	17968	24616	88453
재활용처리량	일일(톤)	25	58	49	68	242

```
#열추출
df2.loc[:, '금정구']
```

```
항목
매립처리량        7240
매립처리량          20
소각처리량         222
소각처리량           1
재활용처리량     56153
재활용처리량       154
음식물류발생량   21273
음식물류발생량      58
Name: 금정구, dtype: int64
```

```
df2.loc[:, ['구분','금정구']]
```

항목	구분	금정구
매립처리량	연간(톤)	7240
매립처리량	일일(톤)	20
소각처리량	연간(톤)	222
소각처리량	일일(톤)	1
재활용처리량	연간(톤)	56153
재활용처리량	일일(톤)	154
음식물류발생량	연간(톤)	21273
음식물류발생량	일일(톤)	58

```
df2.loc[['매립처리량','재활용처리량'], ['구분','금정구']]
```

항목	구분	금정구
매립처리량	연간(톤)	7240
매립처리량	일일(톤)	20
재활용처리량	연간(톤)	56153
재활용처리량	일일(톤)	154

Pandas DataFrame 슬라이싱

- .iloc[행순서번호, 열순서번호]

```
#행추출
df2.iloc[:2]
```

항목	구분	중구	서구	동구	영도구	부산진구	동
매립처리량	연간(톤)	3034	2323	1653	1219	17393	9
매립처리량	일일(톤)	8	7	5	3	48	

```
df2.iloc[[0,1,4,5]]
```

항목	구분	중구	서구	동구	영도구	부산진구
매립처리량	연간(톤)	3034	2323	1653	1219	17393
매립처리량	일일(톤)	8	7	5	3	48
재활용처리량	연간(톤)	9082	21301	17968	24616	88453
재활용처리량	일일(톤)	25	58	49	68	242

```
#열추출
df2.iloc[:, 11]
```

```
항목
매립처리량      7240
매립처리량        20
소각처리량       222
소각처리량         1
재활용처리량     56153
재활용처리량       154
음식물류발생량    21273
음식물류발생량       58
Name: 금정구, dtype: int64
```

```
df2.iloc[:, [0,11]]
```

항목	구분	금정구
매립처리량	연간(톤)	7240
매립처리량	일일(톤)	20
소각처리량	연간(톤)	222
소각처리량	일일(톤)	1
재활용처리량	연간(톤)	56153
재활용처리량	일일(톤)	154
음식물류발생량	연간(톤)	21273
음식물류발생량	일일(톤)	58

```
df2.iloc[[0,1,4,5], [0,11]]
```

항목	구분	금정구
매립처리량	연간(톤)	7240
매립처리량	일일(톤)	20
재활용처리량	연간(톤)	56153
재활용처리량	일일(톤)	154

해결문제

- 부산시기온.csv파일을 읽어서 2018년 최고, 최저 기온을 추출하고 통계량을 표시하시오.

	최고	최저
count	12.000000	12.000000
mean	25.158333	5.425000
std	7.292519	10.782741
min	14.000000	-9.900000
25%	21.425000	-2.300000
50%	24.950000	5.450000
75%	30.325000	14.950000
max	36.400000	21.200000

Pandas DataFrame 열 유형

- **데이터프레임.dtypes**
  - 각열의 유형을 알려줌
- **특정 유형으로 변경**
  - 열시리즈.astype()
    - .astype(float64) , .astype(float32) , .astype(float16), .astype(int)

```
df.dtypes
```

```
일시     datetime64[ns]
최고            float64
최저            float64
차이            float64
평균            float64
년도              int64
월                int64
일                int64
dtype: object
```

```
df['차이'] = df['차이'].astype(int)
df.dtypes
```

```
일시     datetime64[ns]
최고            float64
최저            float64
차이              int64
평균            float64
년도              int64
월                int64
일                int64
dtype: object
```

```
df
```

	일시	최고	최저	차이	평균	년도	월	일
0	2018-01-01	14.0	-9.9	23.9	2.05	2018	1	1
1	2018-02-01	14.7	-9.6	24.3	2.55	2018	2	1

```
df
```

	일시	최고	최저	차이	평균	년도	월	일
0	2018-01-01	14.0	-9.9	23	2.05	2018	1	1
1	2018-02-01	14.7	-9.6	24	2.55	2018	2	1

Pandas DataFrame 열 유형

- 특정 유형으로 변경
  - pd.to_numeric()
    - 값을 숫자 유형으로 변경
    - errors 매개 변수
      - errors='ignore' : 무시
      - errors='coerce' : NaN

```
lt = ['1','2','a']
df2 = pd.DataFrame(lt)
df2.columns = ['자료']
df2
```

	자료
0	1
1	2
2	a

```
df2['자료'] = pd.to_numeric(df2['자료'])
df2.dtypes
```

```
---------------------------------------------
ValueError                                Tr
pandas/_libs/lib.pyx in pandas._libs.lib.may

ValueError: Unable to parse string "a"

During handling of the above exception, anot
```

```
df2['자료'] = pd.to_numeric(df2['자료'], errors='ignore')
df2
```

	자료
0	1
1	2
2	a

```
df2.dtypes
```

```
자료    object
dtype: object
```

```
df2['자료'] = pd.to_numeric(df2['자료'], errors='coerce')
df2
```

	자료
0	1.0
1	2.0
2	NaN

```
df2.dtypes
```

```
자료    float64
dtype: object
```

Pandas DataFrame 열 유형

- 특정 유형으로 변경
  - pd.to_datetime(열명)
    - .dt.year : 년도 , .dt.month : 월, .dt.day : 일

```
dfbusan.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 12 to 23
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   일시      12 non-null     object
 1   최고      12 non-null     float64
 2   최저      12 non-null     float64
dtypes: float64(2), object(1)
memory usage: 420.0+ bytes
```

```
dfbusan['일시'] = pd.to_datetime(dfbusan['일시'])
dfbusan.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 12 to 23
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   일시      12 non-null     datetime64[ns]
 1   최고      12 non-null     float64
 2   최저      12 non-null     float64
dtypes: datetime64[ns](1), float64(2)
memory usage: 420.0 bytes
```

Pandas DataFrame 열 추가

```
dfbusan['년도'] = dfbusan['일시'].dt.year
dfbusan['월'] = dfbusan['일시'].dt.month
dfbusan['일'] = dfbusan['일시'].dt.day
dfbusan
```

	일시	최고	최저	년도	월	일
12	2018-01-01	14.0	-9.9	2018	1	1
13	2018-02-01	14.7	-9.6	2018	2	1
14	2018-03-01	22.3	-0.7	2018	3	1
15	2018-04-01	24.3	3.0	2018	4	1

```
dfbusan['차이'] = dfbusan['최고'] - dfbusan['최저']
dfbusan
```

	일시	최고	최저	차이
12	2018-01-01	14.0	-9.9	23.9
13	2018-02-01	14.7	-9.6	24.3
14	2018-03-01	22.3	-0.7	23.0
15	2018-04-01	24.3	3.0	21.3

```
dfbusan['평균'] = dfbusan[['최고','최저']].mean(axis=1)
dfbusan
```

	일시	최고	최저	차이	평균
12	2018-01-01	14.0	-9.9	23.9	2.05
13	2018-02-01	14.7	-9.6	24.3	2.55
14	2018-03-01	22.3	-0.7	23.0	10.80
15	2018-04-01	24.3	3.0	21.3	13.65

Pandas DataFrame 행 추가

```
df['일시'] = df['일시'].astype(str)
df.dtypes
```

```
일시     object
최고    float64
최저    float64
차이    float64
평균    float64
dtype: object
```

```
df = df.set_index('일시')
df
```

일시	최고	최저	차이	평균
2018-01-01	14.0	-9.9	23.9	2.05
2018-02-01	14.7	-9.6	24.3	2.55
2018-03-01	22.3	-0.7	23.0	10.80

```
df.loc['평균'] = df.mean()
df
```

일시	최고	최저	차이	평균
2018-01-01	14.000000	-9.900	23.900000	2.050000
2018-02-01	14.700000	-9.600	24.300000	2.550000
2018-03-01	22.300000	-0.700	23.000000	10.800000
2018-04-01	24.300000	3.000	21.300000	13.650000
2018-05-01	26.600000	9.800	16.800000	18.200000
2018-06-01	31.300000	15.100	16.200000	23.200000
2018-07-01	35.400000	17.800	17.600000	26.600000
2018-08-01	36.400000	21.200	15.200000	28.800000
2018-09-01	30.000000	14.900	15.100000	22.450000
2018-10-01	25.600000	7.900	17.700000	16.750000
2018-11-01	22.500000	2.700	19.800000	12.600000
2018-12-01	18.800000	-7.100	25.900000	5.850000
평균	25.158333	5.425	19.733333	15.291667

Pandas DataFrame 행/열 삭제

- .drop(행인덱스, axis=0)
- .drop(열인덱스, axis=1)

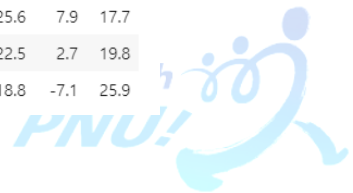
일시	최고	최저	차이	평균
2018-01-01	14.000000	-9.900	23.900000	2.050000
2018-02-01	14.700000	-9.600	24.300000	2.550000
2018-03-01	22.300000	-0.700	23.000000	10.800000
2018-04-01	24.300000	3.000	21.300000	13.650000
2018-05-01	26.600000	9.800	16.800000	18.200000
2018-06-01	31.300000	15.100	16.200000	23.200000
2018-07-01	35.400000	17.800	17.600000	26.600000
2018-08-01	36.400000	21.200	15.200000	28.800000
2018-09-01	30.000000	14.900	15.100000	22.450000
2018-10-01	25.600000	7.900	17.700000	16.750000
2018-11-01	22.500000	2.700	19.800000	12.600000
2018-12-01	18.800000	-7.100	25.900000	5.850000
평균	25.158333	5.425	19.733333	15.291667

```
df = df.drop('평균', axis=0)
df
```

일시	최고	최저	차이	평균
2018-01-01	14.0	-9.9	23.9	2.05
2018-02-01	14.7	-9.6	24.3	2.55
2018-03-01	22.3	-0.7	23.0	10.80
2018-04-01	24.3	3.0	21.3	13.65
2018-05-01	26.6	9.8	16.8	18.20
2018-06-01	31.3	15.1	16.2	23.20
2018-07-01	35.4	17.8	17.6	26.60
2018-08-01	36.4	21.2	15.2	28.80
2018-09-01	30.0	14.9	15.1	22.45
2018-10-01	25.6	7.9	17.7	16.75
2018-11-01	22.5	2.7	19.8	12.60
2018-12-01	18.8	-7.1	25.9	5.85

```
df = df.drop('평균', axis=1)
df
```

일시	최고	최저	차이
2018-01-01	14.0	-9.9	23.9
2018-02-01	14.7	-9.6	24.3
2018-03-01	22.3	-0.7	23.0
2018-04-01	24.3	3.0	21.3
2018-05-01	26.6	9.8	16.8
2018-06-01	31.3	15.1	16.2
2018-07-01	35.4	17.8	17.6
2018-08-01	36.4	21.2	15.2
2018-09-01	30.0	14.9	15.1
2018-10-01	25.6	7.9	17.7
2018-11-01	22.5	2.7	19.8
2018-12-01	18.8	-7.1	25.9



해결문제

- mv1.csv 파일을 읽어서 2019년 자료를 추출하여 평점 평균을 구하시오.

```
df2019.tail()
```

일자	영화명	순위	평점
2019-07-31	마이펫의 이중생활 2	47	7.900000
2019-06-19	롱 리브 더 킹: 목포 영웅	48	7.600000
2019-12-11	쥬만지: 넥스트 레벨	49	5.900000
2019-06-26	존 윅 3: 파라벨룸	50	6.900000
평균			7.481633