

Pandas



Pandas

- 파이썬에서 사용하는 데이터분석 라이브러리
 - 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있게 되며 보다 안정적으로 대용량의 데이터들을 처리하는데 매우 편리한 도구
 - NumPy를 기반으로 구축되었으며 과학 컴퓨팅 환경 내에서 다른 많은 타사 라이브러리와 잘 통합되도록 설계
 - `import pandas as pd`
- 자료구조
 - Series
 - 1차원 자료구조
 - index로 value를 구분하는 1차원 자료구조
 - DataFrame
 - 2차원 자료구조인 DataFrame는 행과 열이 있는 테이블 데이터



Pandas Series

- 1차원 자료구조

– index로 value를 구분하는 1차원 자료구조

list를 series로 형 변환

```
tempavg = list(dt['평균'])  
print(tempavg)  
print(type(tempavg))
```

```
[4.1, 5.5, 9.0, 15.0, 19.0, 21.3,  
<class 'list'>
```



```
#판다스 series 만들기  
ds = pd.Series(tempavg)  
print(ds)  
print(type(ds))
```

```
0    4.1  
1    5.5  
2    9.0  
3   15.0
```

index와 value 추출

```
#인덱스와 값 추출  
print(ds.index)  
print(ds.keys())  
print(ds.values)
```

```
RangeIndex(start=0, stop=30, step=1)  
RangeIndex(start=0, stop=30, step=1)  
[ 4.1  5.5  9.  15.  19.  21.3 26.1 27.  22.6 18.1 11.2  3.6  1.9  3.9  
 10.  14.5 17.8 21.5 26.7 27.9 21.9 16.4 12.4  5.7  4.5  6.6 10.5 13.4  
 19.3 21.1]
```

index로 행 추출

```
#series 데이터 가져오기  
ds1 = ds[11:16]  
print(ds1)
```

```
11    3.6  
12    1.9  
13    3.9  
14   10.0  
15   14.5  
dtype: float64
```

Pandas Series

index 변경하기

```
dsDay = dt['일시']  
dsDay[:5]
```

```
['2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01', '2017-05-01']
```

```
#series 인덱스 지정하기  
ds = pd.Series(tempavg, index = dsDay)  
print(ds)  
print(type(ds))
```

```
2017-01-01    4.1  
2017-02-01    5.5  
2017-03-01    9.0  
2017-04-01   15.0
```

```
#인덱스와 값 추출  
print(ds.index)  
print(ds.keys())  
print(ds.values)
```

```
Index(['2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01', '2017-05-01',  
      '2017-06-01', '2017-07-01', '2017-08-01', '2017-09-01', '2017-10-01',  
      '2017-11-01', '2017-12-01', '2018-01-01', '2018-02-01', '2018-03-01',  
      '2018-04-01', '2018-05-01', '2018-06-01', '2018-07-01', '2018-08-01',  
      '2018-09-01', '2018-10-01', '2018-11-01', '2018-12-01', '2019-01-01',  
      '2019-02-01', '2019-03-01', '2019-04-01', '2019-05-01', '2019-06-01'],  
      dtype='object')  
Index(['2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01', '2017-05-01',  
      '2017-06-01', '2017-07-01', '2017-08-01', '2017-09-01', '2017-10-01',  
      '2017-11-01', '2017-12-01', '2018-01-01', '2018-02-01', '2018-03-01',  
      '2018-04-01', '2018-05-01', '2018-06-01', '2018-07-01', '2018-08-01',  
      '2018-09-01', '2018-10-01', '2018-11-01', '2018-12-01', '2019-01-01',  
      '2019-02-01', '2019-03-01', '2019-04-01', '2019-05-01', '2019-06-01'],  
      dtype='object')  
[ 4.1  5.5  9.  15.  19.  21.3 26.1 27.  22.6 18.1 11.2  3.6  1.9  3.9  
 10.  14.5 17.8 21.5 26.7 27.9 21.9 16.4 12.4  5.7  4.5  6.6 10.5 13.4  
 19.3 21.1]
```

index로 데이터 가져오기

```
#series 데이터 가져오기  
ds1 = ds['2018-01-01':'2018-12-01']  
print(ds1)
```

```
2018-01-01    1.9  
2018-02-01    3.9  
2018-03-01   10.0  
2018-04-01   14.5  
2018-05-01   17.8  
2018-06-01   21.5  
2018-07-01   26.7  
2018-08-01   27.9  
2018-09-01   21.9  
2018-10-01   16.4  
2018-11-01   12.4  
2018-12-01    5.7  
dtype: float64
```

```
#series 데이터 가져오기  
ds1 = ds[['2018-01-01', '2018-12-01']]  
print(ds1)
```

```
2018-01-01    1.9  
2018-12-01    5.7  
dtype: float64
```

조건으로 데이터 가져오기

```
ds2 = ds[ds > 20]  
ds2
```

```
2017-06-01   21.3  
2017-07-01   26.1  
2017-08-01   27.0  
2017-09-01   22.6  
2018-06-01   21.5  
2018-07-01   26.7  
2018-08-01   27.9  
2018-09-01   21.9  
2019-06-01   21.1  
dtype: float64
```



Pandas Series 기초 통계

- `mean()` : 평균
- `max()` : 최대
- `min()` : 최소
- `std()` : 표준편차
- `describe()`
 - 요약 통계량 계산

```
print(f'평균 : {ds.mean():0.2f}')  
print(f'최대 : {ds.max():0.2f}')  
print(f'최소 : {ds.min():0.2f}')  
print(f'표준편차 : {ds.std():0.2f}')  
print(f'요약 \n{ds.describe() }')
```

```
평균 : 14.62  
최대 : 27.90  
최소 : 1.90  
표준편차 : 7.99  
요약  
count    30.000000  
mean     14.616667  
std       7.989048  
min       1.900000  
25%       7.200000  
50%      14.750000  
75%      21.250000  
max      27.900000  
dtype: float64
```



series 자주사용되는 메소드

- **replace()**
 - 특정값을 가진 요소 값을 변경
- **equals()**
 - 시리즈와 시리즈가 같은지 비교
- **isin()**
 - 시리즈에 포함된 값이 있는지 확인
- **drop_duplicates()**
 - 중복값이 없는 시리즈 반환
- **append()**
 - 2개이상의 시리즈 연결
- **sort_index()**
 - 인덱스로 정렬
- **sort_values()**
 - 값으로 정렬
- **to_frame()**
 - 시리즈를 데이터프레임으로 변경



series 연산

- 시리즈의 길이가 같아야 하고 인덱스가 같은 것끼리 연산
 - + : 같은 길이의 시리즈와 시리즈 더하기
 - * : 같은 길이의 시리즈와 시리즈 곱하기

```
print(ds1 + 1)
print(ds2 * 3)
```

```
2017-01-01    5.1
2017-02-01    6.5
2017-03-01   10.0
2017-04-01   16.0
2017-05-01   20.0
2017-06-01   22.3
dtype: float64
2018-01-01    5.7
2018-02-01   11.7
2018-03-01   30.0
2018-04-01   43.5
2018-05-01   53.4
2018-06-01   64.5
dtype: float64
```

ds1 + ds2

```
2017-01-01    NaN
2017-02-01    NaN
2017-03-01    NaN
2017-04-01    NaN
2017-05-01    NaN
2017-06-01    NaN
2018-01-01    NaN
2018-02-01    NaN
2018-03-01    NaN
2018-04-01    NaN
2018-05-01    NaN
2018-06-01    NaN
dtype: float64
```

인덱스 변경



```
ds1.index = list(range(len(ds1)))
ds1
```

```
0    4.1
1    5.5
2    9.0
3   15.0
4   19.0
5   21.3
dtype: float64
```

```
ds2.index = list(range(len(ds2)))
ds2
```

```
0    1.9
1    3.9
2   10.0
3   14.5
4   17.8
5   21.5
dtype: float64
```

ds1 + ds2

```
0    6.0
1    9.4
2   19.0
3   29.5
4   36.8
5   42.8
dtype: float64
```

ds1 * ds2

```
0    7.79
1   21.45
2   90.00
3  217.50
4  338.20
5  457.95
dtype: float64
```

Pandas DataFrame

- 2차원 자료구조인 DataFrame는 행과 열이 있는 테이블 데이터

딕셔너리를 DataFrame으로 형 변환

```
print(dt)
```

```
{'일시': ['2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01'],
```



```
#딕셔너리를 데이터 프레임으로 변경  
#키가 열이 되고 값이 행이 됨  
df = pd.DataFrame(dt)  
df
```

데이터 프레임 정보 보기

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 30 entries, 0 to 29  
Data columns (total 4 columns):  
#   Column  Non-Null Count  Dtype  
---  -  
0   일시    30 non-null      object  
1   평균    30 non-null      float64  
2   최고    30 non-null      float64  
3   최저    30 non-null      float64  
dtypes: float64(3), object(1)  
memory usage: 1.1+ KB
```

	일시	평균	최고	최저
0	2017-01-01	4.1	15.1	-7.7
1	2017-02-01	5.5	17.7	-5.7
2	2017-03-01	9.0	18.1	-2.3
3	2017-04-01	15.0	23.2	5.0



Pandas DataFrame

열 인덱스

```
print(df.columns)
```

```
Index(['일시', '평균', '최고', '최저'], dtype='object')
```

행 인덱스

```
print(df.index)
```

```
RangeIndex(start=0, stop=30, step=1)
```

#행 인덱스 변경

```
df = df.set_index('일시')  
df
```

	일시	평균	최고	최저
	2017-01-01	4.1	15.1	-7.7
	2017-02-01	5.5	17.7	-5.7

#행 인덱스를 자료형으로 변경

```
df = df.reset_index()  
df
```

	일시	평균	최고	최저
0	2017-01-01	4.1	15.1	-7.7
1	2017-02-01	5.5	17.7	-5.7
2	2017-03-01	9.0	18.1	-2.3

Pandas DataFrame

열 추출

```
#열가져오기
dfavg = df['평균']
print(type(davg))
print(davg)
```

```
<class 'pandas.core.series.Series'>
0    4.1
1    5.5
2    9.0
3   15.0
```

```
#열가져오기
dfminmax = df[['일시', '최고', '최저']]
print(type(dfminmax))
dfminmax
```

```
<class 'pandas.core.frame.DataFrame'>
```

	일시	최고	최저
0	2017-01-01	15.1	-7.7
1	2017-02-01	17.7	-5.7
2	2017-03-01	18.1	-2.3

행 추출

```
#행 데이터 가져오기
df29 = df.loc[3:5]
df29
```

	일시	평균	최고	최저
3	2017-04-01	15.0	23.2	5.0
4	2017-05-01	19.0	28.2	12.8
5	2017-06-01	21.3	30.4	15.4

```
#행 데이터 가져오기
df29 = df.loc[[3,5]]
df29
```

	일시	평균	최고	최저
3	2017-04-01	15.0	23.2	5.0
5	2017-06-01	21.3	30.4	15.4



Pandas 외부데이터 가져오기

- csv 읽기

- `df = pd.read_csv('파일명')`
- 한글 파일명을 사용할 경우 :
`df = pd.read_csv('파일명', engine='python')`

- csv 쓰기

- `df.to_csv('파일명')`

```
import pandas as pd
```

```
#외부 데이터 가져오기
```

```
df = pd.read_csv('부산시기온.csv', engine='python')  
df
```

	일시	평균	최고	최저
0	2017-01-01	4.1	15.1	-7.7
1	2017-02-01	5.5	17.7	-5.7
2	2017-03-01	9.0	18.1	-2.3
3	2017-04-01	15.0	23.2	5.0
4	2017-05-01	19.0	28.2	12.8
5	2017-06-01	21.3	30.4	15.4

```
#데이터 생성하기
```

```
subset = df.iloc[0:2, 0:2]  
subset
```

	일시	평균
0	2017-01-01	4.1
1	2017-02-01	5.5

```
subset.to_csv('부분기온.csv')
```

해결문제

- 부산쓰레기발생2017년.csv 파일에서 '금정구'의 연간 매립처리량과 소각처리량, 음식물류발생량 정보를 추출하시오.
 - 한글 인코딩 : euc-kr , cp949

연간(톤):매립처리량	7240.0
연간(톤):소각처리량	222.0
연간(톤):재활용처리량	56153.0
연간(톤):음식물류발생량	21273.0
dtype: float64	

