Big Data

Pandas

Python

Pandas 모듈

- Series Class : 1차원
  - 인덱스(index) + 값(value)
- DataFrame Class : 2차원
  - 표와 같은 형태

Pandas 사용하기

- import pandas as pd

  - Pandas 모듈(라이브러리)를 import하고 앞으로 pd라는 이름으로 부른다.

Series 사용

Series 생성

- obj = pd.Series([4,7,-5,3])

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

인덱스 지정하여 생성하기

- obj2 = pd.Series([4,7,-5,3], index=['d','b','a','c'])

```
d    4
b    7
a   -5
c    3
dtype: int64
```

Series 값 확인

- obj2.values

```
array([ 4,  7, -5,  3], dtype=int64)
```

Series 인덱스 확인

- obj2.index

```
Index(['d', 'b', 'a', 'c'], dtype='object')
```

Series 타입 확인

- obj2.dtype

```
dtype('int64')
```

각 도시의 2015년 인구 데이터를 시리즈로 만들어 보세요.

- s = pd.Seires

```
서울     9904312
부산     3448737
인천     2890451
대구     2466052
dtype: int64
```

Series에 이름 지정

- s.name = “인구”
- s.index.name = “도시”

```
도시
서울    9904312
부산    3448737
인천    2890451
대구    2466052
Name: 인구, dtype: int64
```

Series연산

- s / 1000000

```
도시
서울    9.904312
부산    3.448737
인천    2.890451
대구    2.466052
Name: 인구, dtype: float64
```

Series 인덱싱

- s[1], s["부산"]

```
(3448737, 3448737)
```

- s[3], s["대구"]

```
(2466052, 2466052)
```

- s[[0,3,1]]
- s[["서울", "대구", "부산"]]

```
도시
서울    9904312
대구    2466052
부산    3448737
Name: 인구, dtype: int64
```

Series Boolean 인덱싱

- s>=2500000

```
도시
서울      True
부산      True
인천      True
대구     False
Name: 인구, dtype: bool
```

- s[s>=2500000]

```
도시
서울    9904312
부산    3448737
인천    2890451
Name: 인구, dtype: int64
```

인구수가 250만 이상의 도시

- s[s>=2500000]

```
도시
서울    9904312
부산    3448737
인천    2890451
Name: 인구, dtype: int64
```

인구수가 500만 이하의 도시

- s[s<=5000000]

```
도시
부산    3448737
인천    2890451
대구    2466052
Name: 인구, dtype: int64
```

인구수가 250만 이상 500만 이하의 도시

- s[(s>=2500000) & (s<=5000000)]

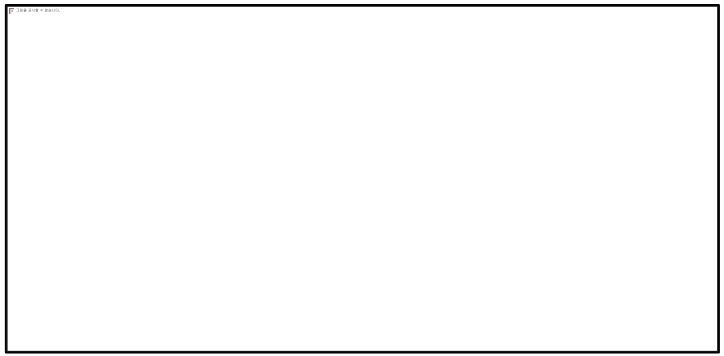
```
도시
부산    3448737
인천    2890451
Name: 인구, dtype: int64
```

Series 슬라이싱

- s[1 : 3]

```
도시
부산    3448737
인천    2890451
Name: 인구, dtype: int64
```

- s[“부산” : “대구”]



딕셔너리 객체로 Series 생성 (2010년 인구수)

- data = {"서울": 9631482, "부산": 3393191,
  "인천": 2632035, "대전": 1490158}

- s2 = pd.Series(data)

```
대전     1490158
부산     3393191
서울     9631482
인천     2632035
dtype: int64
```

2015년도와 2010년도의 인구 증가를 계산

- ds = s - s2

```
대구          NaN
대전          NaN
부산      55546.0
서울     272830.0
인천     258416.0
dtype: float64
```

- ds.notnull()

```
도시
서울      True
부산      True
인천      True
대구     False
대전     False
Name: 인구, dtype: bool
```

- ds[ds.notnull()]

```
도시
서울    9904312.0
부산    3448737.0
인천    2890451.0
Name: 인구, dtype: float64
```

- ds.isnull()

```
도시
서울    False
부산    False
인천    False
대구     True
대전     True
Name: 인구, dtype: bool
```

- ds[ds.isnull()]

```
도시
대구   NaN
대전   NaN
Name: 인구, dtype: float64
```

2015년도와 2010년도의 인구 증가율(%)를 계산

- rs = (s – s2)/s2*100
- rs[rs.notnull()]

```
부산    1.636984
서울    2.832690
인천    9.818107
dtype: float64
```

Series 데이터 갱신, 추가, 삭제

- rs[“부산”] = 1.6
- rs[“대구”] = 1.41
- del rs[“서울”]
- rs[rs.notnull()]

```
대구      1.410000
부산      1.600000
인천      9.818107
dtype: float64
```

DataFrame 사용

DataFrame을 만드는 방법

- data = {
  "2015": [9904312, 3448737, 2890451, 2466052],
  “2010": [9631482, 3393191, 2632035, 2431774]
  }
- df = pd.DataFrame(data)

	2010	2015
0	9631482	9904312
1	3393191	3448737
2	2632035	2890451
3	2431774	2466052

DataFrame 인덱스 수정

- df.index = [“서울”, “부산”, “인천“, “대구“]

	2010	2015
서울	9631482	9904312
부산	3393191	3448737
인천	2632035	2890451
대구	2431774	2466052

DataFrame 인덱스 지정하여 생성

- data = [
  [9904312, 3448737, 2890451, 2466052],
  [9631482, 3393191, 2632035, 2431774]
  ]
- ind = [“2015”, “2010”]
- col = [“서울”, “부산“, “인천“, “대구“]
- df2 = pd.DataFrame(data, index = ind, columns = col)

	서울	부산	인천	대구
2015	9904312	3448737	2890451	2466052
2010	9631482	3393191	2632035	2431774

df2.T

	2015	2010
서울	9904312	9631482
부산	3448737	3393191
인천	2890451	2632035
대구	2466052	2431774

DataFrame을 이용하여 아래와 같은 결과를 구성하시오.

	홍길동	김사또	임꺽정
키	175.3	180.2	178.6
몸무게	66.2	78.9	55.1
나이	27.0	49.0	35.0

DataFrame 값 확인

- df.values

```
array([[9904312, 9631482],
       [3448737, 3393191],
       [2890451, 2632035],
       [2466052, 2431774]], dtype=int64)
```

DataFrame 인덱스 확인

- df.index

```
Index(['서울', '부산', '인천', '대구'], dtype='object')
```

DataFrame 컬럼 확인

- df.columns

```
Index(['2015', '2010'], dtype='object')
```

DataFrame 열 인덱스

- df["2015"]

```
서울      9904312
부산      3448737
인천      2890451
대구      2466052
Name: 2015, dtype: int64
```

- df[["2010"]]

	2015
서울	9904312
부산	3448737
인천	2890451
대구	2466052

- df[["2010", "2015"]]

	2010	2015
서울	9631482	9904312
부산	3393191	3448737
인천	2632035	2890451
대구	2431774	2466052

DataFrame 열 인덱스

- df[["2015","2010"]]

	2015	2010
서울	9904312	9631482
부산	3448737	3393191
인천	2890451	2632035
대구	2466052	2431774

- df[["2010", "2015"]]

	2010	2015
서울	9631482	9904312
부산	3393191	3448737
인천	2632035	2890451
대구	2431774	2466052

"2005"라는 컬럼명으로 2005년 인구수 대입

- df ["2005"] = [9762546, 3512547, 2517680, 2456016]

	2015	2010	2005
서울	9904312	9631482	9762546
부산	3448737	3393191	3512547
인천	2890451	2632035	2517680
대구	2466052	2431774	2456016

DataFrame 행 인덱싱

- df[0:1]

	2015	2010	2005
서울	9904312	9631482	9762546

DataFrame 행 인덱싱

- df[“서울”:”인천”]

	2015	2010	2005
서울	9904312	9631482	9762546
부산	3448737	3393191	3512547
인천	2890451	2632035	2517680

loc(), iloc() 함수

- loc()

  - 실제 인덱스를 사용하여 행을 가지고 올 때 사용

  > df.loc[행,열]

loc(), iloc() 함수

- loc()

  - 실제 인덱스를 사용하여 행을 가지고 올 때 사용

  > df.loc[“서울”]

```
2015    9904312.0
2010    9631482.0
2005    9762546.0
Name: 서울, dtype: float64
```

loc(), iloc() 함수

- loc()

  - 실제 인덱스를 사용하여 행을 가지고 올 때 사용

  > df.loc[“서울”:”부산”,”2015”:”2010”]

	2015	2010
서울	9904312	9631482
부산	3448737	3393191

loc(), iloc() 함수

- iloc()

  - numpy의 array인덱싱 방식으로 행을 가지고 올 때 사용

  >df.iloc[3]

```
2015    2890451.0
2010    2632035.0
2005    2517680.0
Name: 인천, dtype: float64
```

Pandas Boolean 인덱싱

- df ["2010"] >=2500000

```
서울      True
부산      True
인천      True
대구     False
Name: 2010, dtype: bool
```

csv파일 불러오기

```
population_number = pd.read_csv("population_number.csv",encoding="euc-kr")
```

	도시	지역	2015	2010	2005	2000
0	서울	수도권	9904312	9631482.0	9762546.0	9853972
1	부산	경상권	3448737	NaN	NaN	3655437
2	인천	수도권	2890451	2632035.0	NaN	2466338
3	대구	경상권	2466052	2431774.0	2456016.0	2473990

csv파일 불러오기

```
pd.read_csv("population_number.csv",index_col="도시", encoding="euc-kr",)
```

도시	지역	2015	2010	2005	2000
서울	수도권	9904312	9631482.0	9762546.0	9853972
부산	경상권	3448737	NaN	NaN	3655437
인천	수도권	2890451	2632035.0	NaN	2466338
대구	경상권	2466052	2431774.0	2456016.0	2473990

count 함수

- 데이터 개수를 셀 수 있다.

```
population_number.count()
```

```
도시          4
지역          4
2015      3
2010      2
2005      4
2000      4
dtype: int64
```

Python

value_counts 함수

- 값이 숫자, 문자열, 카테고리 값인 경우에 각각의 값이 나온 횟수를 셀 수 있다.

➢ s2 = pd.Series(np.random.randint(6, size=100))

➢ s2.tail()

```
Out[18]: 95    5
         96    5
         97    5
         98    3
         99    4
         dtype: int32
```

➢ s2.value_counts()

```
Out[19]: 4    25
         3    18
         5    17
         0    16
         1    15
         2     9
         dtype: int64
```

```
population_number["2015"].value_counts()
```

```
2466052    1
2890451    1
3448737    1
9904312    1
Name: 2015, dtype: int64
```

```
population_number["2010"].value_counts()
```

```
9631482.0    1
2431774.0    1
2632035.0    1
Name: 2010, dtype: int64
```

정렬

- sort_index 함수 : 인덱스 값을 기준으로 정렬한다.
- sort_values 함수 : 데이터 값을 기준으로 정렬한다.

정렬

```
population_number["2010"].sort_values()
```

```
도시
대구     2431774.0
인천     2632035.0
서울     9631482.0
부산           NaN
```

정렬

```
population_number["2010"].sort_values(ascending=False)
```

```
도시
서울     9631482.0
인천     2632035.0
대구     2431774.0
부산           NaN
Name: 2010, dtype: float64
```

score.csv파일 불러오기

과목	1반	2반	3반	4반
수학	45	44	73	39
영어	76	92	45	69
국어	47	92	45	69
사회	92	81	85	40
과학	11	79	47	26

학급별 총계

```
score.sum()
```

```
1반     271
2반     388
3반     295
4반     243
dtype: int64
```

학급별 순위

```
score.sum().sort_values()
```

```
4반     243
1반     271
3반     295
2반     388
dtype: int64
```

과목별 총계

```
score.sum(axis=1)
```

```
과목
수학    201
영어    282
국어    253
사회    298
과학    163
dtype: int64
```

과목별 합계 DataFrame에 추가하기

```
score["합계"]=score.sum(axis=1)
```

과목	1반	2반	3반	4반	합계
수학	45	44	73	39	201
영어	76	92	45	69	282
국어	47	92	45	69	253
사회	92	81	85	40	298
과학	11	79	47	26	163

연습문제 3-1)

과목별 평균을 계산하여 column 추가하기

과목	1반	2반	3반	4반	합계	평균
수학	45	44	73	39	201	
영어	76	92	45	69	282	
국어	47	92	45	69	253	
사회	92	81	85	40	298	
과학	11	79	47	26	163	

연습문제 1)

과목별 평균을 계산하여 column 추가하기

```
score["평균"] = score.loc[ : , :"4반"].mean(axis=1)
```

과목	1반	2반	3반	4반	합계	평균
수학	45	44	73	39	201	50.25
영어	76	92	45	69	282	70.50
국어	47	92	45	69	253	63.25
사회	92	81	85	40	298	74.50
과학	11	79	47	26	163	40.75

연습문제 2)

반평균을 계산 하여 row 추가 하기

```
score.loc["반평균"] =
```

과목	1반	2반	3반	4반	합계	평균
수학	45.0	44.0	73.0	39.0	201.0	50.25
영어	76.0	92.0	45.0	69.0	282.0	70.50
국어	47.0	92.0	45.0	69.0	253.0	63.25
사회	92.0	81.0	85.0	40.0	298.0	74.50
과학	11.0	79.0	47.0	26.0	163.0	40.75
반평균						